Enabling Safe Autonomous Driving in Real-World City Traffic Using Multiple Criteria Decision Making

Andrei Furda and Ljubo Vlacic

Intelligent Control Systems Laboratory (ICSL),
Institute of Integrated and Intelligent Systems, Griffith University,
Brisbane, QLD 4111, Australia, E-mails: andrei.furda@griffithuni.edu.au
and l.vlacic@griffith.edu.au

***Abstract*–**This paper addresses the topic of real-time decision making for autonomous city vehicles, i.e., the autonomous vehicles' ability to make appropriate driving decisions in city road traffic situations. The paper explains the overall controls system architecture, the decision making task decomposition, and focuses on how Multiple Criteria Decision Making (MCDM) is used in the process of selecting the most appropriate driving maneuver from the set of feasible ones. Experimental tests show that MCDM is suitable for this new application area.

I. Introduction

Autonomous city vehicles capable of driving safely through urban traffic and sharing the roads with other traffic participants, have been a vision for many years (Kolodko and Vlacic, 2003; Li and Tang, 2009).

One of the research topics which are crucial for enabling autonomous vehicles to cope with urban traffic conditions is their ability to make safe and appropriate driving decisions in any traffic situation.

The solution to this high-level vehicle decision making and control problem depends on a variety of related research areas and topics, such as (Figure 1):

- *Sensor* technology for on-board sensor systems, which are able to deliver accurate and reliable information about the vehicle's environment in any light, weather, and road conditions.
- *Perception* algorithms which use information obtained from on-board sensor systems and enable the reliable recognition of relevant traffic features, such as traffic signs, road markings, obstacles, pedestrians, vehicles, etc.
- *Localization* technology is crucial for path planning and decision making. The currently used localization methods based on GPS and DGPS, and Inertial Navigation Systems (INS) sensors does not offer the required level of reliability and accuracy in urban areas, where GPS reception between high buildings is unreliable.
- *Communication* technology which enables reliable exchange of information between autonomous vehicles (vehicle-to-

vehicle, V2V), but also between the road infrastructure and autonomous vehicles (vehicles-to-infrastructure, V2I). Communication enables cooperation between vehicles, and is relevant for improving the efficiency and safety of autonomous vehicles.
- *High-Level Vehicle Control* tasks, such as real-time decision making, and the execution of driving maneuvers.
- *Low-Level Vehicle Control* includes the control of actuators for steering angle, accelerator, brakes, gearbox, etc.
- System Reliability, Safety, and Fault Tolerance: Since human safety will depend on them, autonomous vehicles will not be accepted unless they are safer than today's human driven cars.

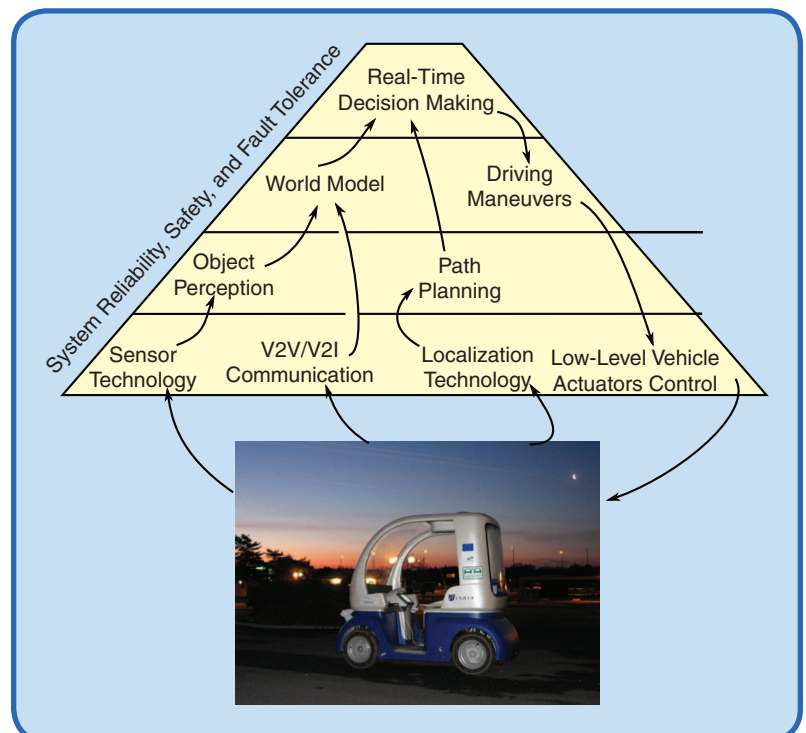This article deals with the high-level vehicle control tasks and addresses the question on enabling driverless

FIG 1 Overview of research topics of autonomous city vehicles and the scope of real-time decision making.

city vehicles to decide about the most appropriate driving maneuver to perform under the given road traffic circumstances, using Multiple Criteria based Decision Making techniques and methods.

The following section gives an overview about the autonomous vehicle's control software architecture.

II. The Autonomous Vehicle Decision Making and Control System

The driverless vehicle decision making and control system consists of the following functional subsystems (Figure 2):

■ Perception Subsystem,
■ Real-Time Decision Making and Driving Maneuver Control,
■ Driving Maneuvers,
■ Vehicle Interface.

The purpose of the Perception Subsystem is to collect available information about the vehicle's road traffic environment obtained from either on-board sensors (Munz et al., 2010), or from external sensors (Li and Jia, 2009; Zhao et al., 2009), to manage and process it, and to provide it in an adequate form to the Real-Time Decision Making and Driving Maneuver Control, and Driving Maneuver subsystems. The Perception Subsystem's components are (Figure 3):

■ A Priori Information: software components providing information which is available before the vehicle begins its journey (typically provided as a data file, such as

the RNDF (route network definition) file in the DARPA Urban Challenge (DARPA, 2006)).

■ Sensor Components: software and hardware components providing information obtained from on-board sensors.

■ Communication: software and hardware components providing information obtained through communication with other vehicles or infrastructure (e.g., traffic management centre).

■ World Model: software component which collects information from subsystems, maintains an up-to-date view of the vehicle's environment, actively notifies other subsystems about relevant events in the traffic environment, and provides access to all its information to other software components through an API (Application Programming Interface) (Figure 3).

Based on the information provided by the Perception Subsystem, the Real-Time Decision Making and Driving Maneuver Control subsystem makes driving decisions. This software subsystem decides about the activation and the execution of the most appropriate driving maneuver for any given traffic situation.

The Driving Maneuvers subsystem contains a set of closed-loop control algorithms, each able to maneuver the vehicle in a specific traffic situation (e.g., road following, overtaking, crossing intersections, etc.). The driving maneuvers direct their output to the Vehicle Interface subsystem.

The Low-Level Vehicle Control subsystem contains hardware and software components, which control the vehicle's speed, steering angle, and other actuators (e.g., transmission).

A. World Model and Route Planner—Information Basis for Decision Making

The main purpose of the World Model is to provide information input for the real-time decision making subsystem. The World Model merges the various types of information and provides at any given time an accurate and up-to-date representation of the driverless vehicle's traffic environment.

The following are the World Model's most relevant functional requirements:

■ Stores a priori information, such as roads, intersections, and traffic signs (a priori known).

■ Stores information provided by sensors, such as obstacles, traffic lanes, and perceived traffic signs.
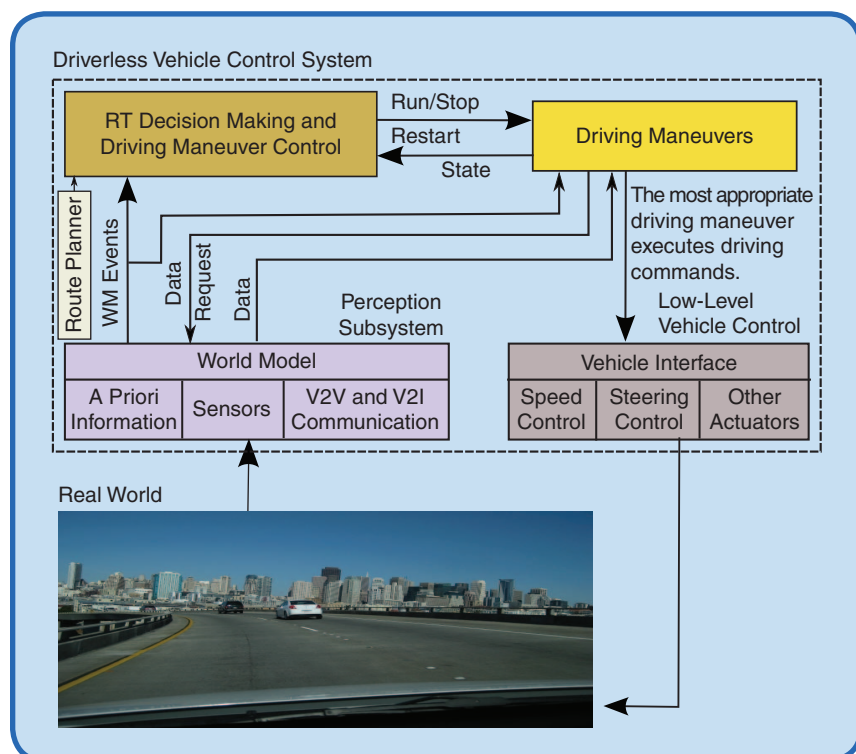

FIG 2 Simplified view of the driverless vehicle decision making and control software architecture and the flow of data.

- Stores information obtained through communication with other vehicles or a traffic management center.
- Cyclicly merges and updates the a priori information with the information obtained continuously from sensor and communication components.
- Calculates cyclicly the relationships between the stored entities. For example, if sufficient data is available, the World Model determines the positions of the current road, the current traffic lane, obstacles on the current road, the type of obstacles on the current lane/road, and distances to obstacles.
- Notifies other subsystems of relevant events in the traffic environment through an asynchronous mechanism.
- Provides other subsystems complete access to all stored information by replying to synchronous data requests (Figure 3).

The World Model API consists of two entities, each implementing a different concept:
- World Model Events,
- Object-oriented data structure.

A World Model Event represents a discrete event which signals the availability of certain information, that certain conditions are suddenly met, or the occurrence of a certain event happening in the real world. The principle is along the line of Discrete Event Systems (Cassandras and Lafortune, 2008). The state of a *World Model Event* is modeled as a boolean variable.

World Model Events are used to notify other software components about the state of certain predefined conditions, which are relevant for the execution of driving maneuvers. The specification of each driving maneuver requires that, in order to be safely performed, certain conditions have to be met, or certain information has to be available. Such conditions can be, for instance, related to traffic situations (e.g., "pedestrian *n* meters in front of the vehicle"), but might as well be related to the availability of sensor information (e.g., "GPS localization available").

The World Model Events enable a flexible way to define what information is relevant for decision making, and provide an easy mechanism for quick information exchange between the World Model and the Real-Time Decision Making subsystem.

The state of all defined World Model Events is provided as a *k*-tuple:

$$WM_{\text{events}} = (w_1, w_2, \ldots, w_k) : w_l \in \{true, \ false\}, \quad (1)$$

where each element $w_l (l = 1, 2, \ldots, k)$ represents an event.

The *k*-tuple WM_{events} is updated cyclically, and other (as observer registered) software components are actively notified about the occurrence of certain conditions as defined for each event. A timestamp assigned to the event tuple may be used to ensure that the update operation is performed within the required time intervals.

Besides World Model Events, the World Model provides other software components full access to all its data through an object-oriented data structure. Further details about the developed World Model have been published in (Furda and Vlacic, 2010).

The Route Planner: In many situations, the planned route has a significant impact on decision making. For instance overtaking a slower vehicle just before a planned turn may not be adequate. Therefore, a route planner provides the decision making module with information about the prospective planned route, well in advance (e.g., 50–150 m) before required turns and changes of direction.

Without the loss of generality, we require that the provided route planner information is specified as an element of the set

$$D_{\text{route}} = \{\, forward\_straight, \ forward\_right, \\ forward\_left, \ turn\_around\}, \quad (2)$$

where each element indicates the future travel direction. The direction *forward_straight* indicates that the planned route is to follow the current road, *forward_right* and *forward_left* indicate a turn in the near future, while *turn_around* indicates that the next destination lies behind and a U-turn is necessary.
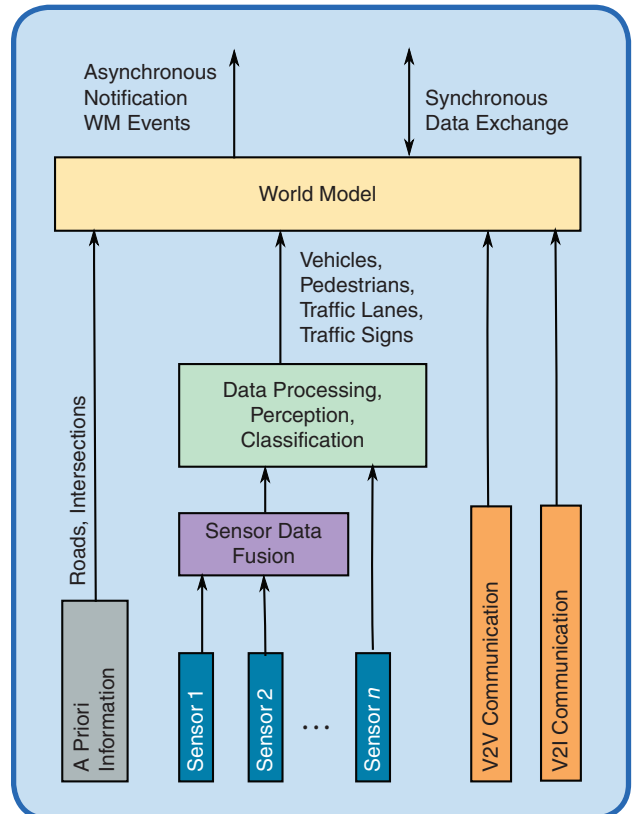

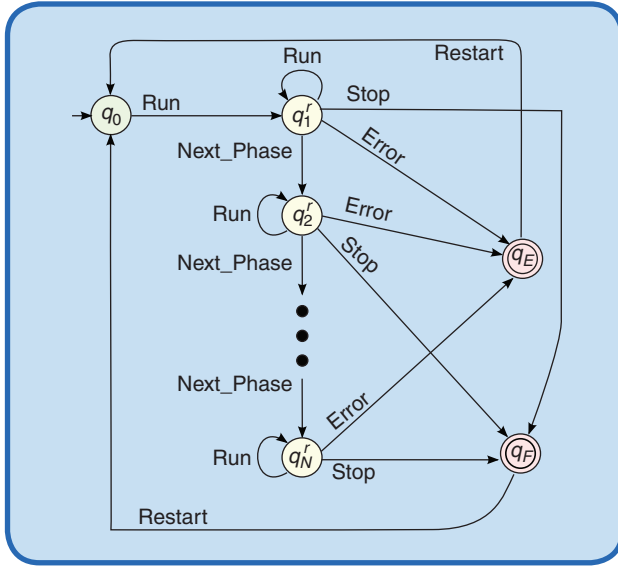
FIG 3 World Model Input and Output.

FIG 4 A driving maneuver finite automaton with multiple Run states. The number of Run states q_i^r, $i = 1, 2, \ldots N$ equals the number of driving maneuver phases.

B. Driving Maneuvers—The Decision Alternatives

Driving maneuvers are closed-loop control algorithms, each capable of maneuvering the driverless vehicle over a time period or distance. All driving maneuvers are structured in a common way. Their operational behaviors are modeled as deterministic finite automata (Hopcroft et al., 2007; Cassandras and Lafortune, 2008) (Figure 4):

- a *start state* q_0 is the waiting or idle state, in which the automaton is waiting for the Run signal.
- a set of *Run states* $Q^{\text{run}} = \{q_1^r, q_2^r, \ldots, q_n^r\} \subset Q$, each representing a phase of a driving maneuver, perform the maneuvering of the vehicle. Each of them includes checking of necessary preconditions, such as the availability of required information and safety conditions. As long as the defined preconditions are met, the Run states execute closed-loop control algorithms. Otherwise, if certain preconditions are not met, the *Error* symbol is generated, and the automaton changes into the error state q_E.
- two final states $\{q_F, q_E\} = F$. The final state q_F (finished) represents a successful completion of the driving maneuver, while the error state q_E (error) signals that the driving maneuver has been aborted due to an error or some other reason.
- a set of input symbols Σ, which consists of at least the symbols: *Run*, *Stop*, *Restart*, *Error*, and the state transition function $\delta : Q \times \Sigma \to Q$.

Furthermore, each driving maneuver offers a set of *execution alternatives*, which are specified by discrete parameters. Parameters are reference values (set points) for the closed-loop control algorithms implemented in the Run states Q^{run}. For instance, an overtaking maneuver requires the specification of parameters which define how close and

how fast the driverless vehicle should approach the front vehicle before changing lanes.

III. Real-Time Decision Making

Based on the elaborated prerequisites, the decision making task can be specified as follows:

- a set $M_{\text{all}} = \{m_1, m_2, \ldots m_n\}$, $n \in \mathbb{N}^1$, of all available driving maneuvers which can be performed by the driverless vehicle,
- a k-tuple $(w_1, w_2, \ldots, w_k) \in W_{\text{events}}$ of World Model Events, $w_l \in \{0, 1\}$, $l = 1, 2, \ldots, k$
- a route planner direction indication $d_i \in D_{\text{route}}$.

The general task of decision making in this context is to identify the most appropriate driving maneuver $m_{\text{most\_appr.}} \in M_{\text{all}}$, which leads to a driverless vehicle driving behavior conforming to the specification.

A. Decomposition into Subtasks

The general decision making task is decomposed into the following two consecutive stages:

1) Decision regarding *feasible, safety-critical driving maneuvers* subject to World Model Events and route planner indication. A driving maneuver is defined as feasible if it can be safely performed in a specific traffic situation, and is conforming to the road traffic rules. In order to ensure safety, it is assumed that the autonomous vehicle will always obey the traffic rules, which however might need to be adapted for such vehicles in the future. In any traffic situation, there can be multiple feasible driving maneuvers (e.g., overtaking a stopped vehicle or waiting for it to continue driving).

2) Decision regarding the most appropriate driving maneuver. This stage selects and starts the execution of one single driving maneuver, which is selected as the most appropriate for the specific traffic situation. Since only those driving maneuvers are considered in this stage which have been selected as feasible (and therefore safe), this stage is not safety-critical because it does not include any decision making attributes which affect safety.

The decomposition into two stages with different objectives leads to subtasks with manageable complexity, enabling the verification and testing of each stage in particular. While the main focus of the first stage is to determine which driving maneuvers are safe and conform to traffic rules, the second, non safety critical decision stage focuses on improving comfort and efficiency.

The entire decision making process (i.e., the two stages) is executed cyclically. Once a driving maneuver has been activated, it remains active within the given cycle, and it will be executed as long as its execution is feasible, until it finishes in one of its two final states q_F or q_E. If a driving

---

[1]$\mathbb{N}$ = set of natural numbers.

maneuver becomes infeasible during its execution, the decision making subsystem is able to abort its execution, and activate another driving maneuver instead.

B. Decision Stage 1: Feasible, Safety-Critical Driving Maneuvers

This section presents a modeling approach for the first stage of the decision making task, the decision regarding the set of feasible driving maneuvers.

The goal of this stage is to decide about the feasible driving maneuvers, i.e., the subset of all driving maneuvers, which can be performed without putting any traffic participants at risk.

The following aspects are relevant for the decision making on feasible driving maneuvers:

- Information about the vehicle's environment, which is provided in the form of World Model Events.
- Knowledge about traffic rules and compliance with them.
- Information about the planned travel direction, which is assumed to be provided by the route planner.

Figure 5 shows the processing steps for the selection of feasible driving maneuvers. Each driving maneuver requires information about the traffic environment, which is provided by the World Model in the form of events. Therefore, occurring World Model Events define which driving maneuvers are operational (i.e., which can be performed). In order to comply to traffic rules, additional restrictions of driving maneuvers are required. The knowledge about traffic rules is embedded in the first step of this stage (DMU1A).

As a third aspect, the planned traveling route plays a further role in reducing the number of candidate driving maneuvers (DMU1B in Figure 5). Driving maneuvers which lead the vehicle into the wrong direction, or maneuver it inadequately with respect to the planned route, are omitted from the set of feasible driving maneuvers. For instance overtaking a slower vehicle while the route planner indicates a necessary U-turn might not be adequate.

Consequently, the large number of factors to be considered in this decision stage requires a model which enables the design and analysis of a highly complex operational behavior. As Petri nets are a suitable modeling method for this purpose (Peterson, 1981), we use Petri nets to model this decision stage.

The decision making unit shown in figure 5 is modeled as a Petri net (Figure 6) consisting of two subnets. Each subnet models the decision making units DMU1A and DMU1B, respectively.
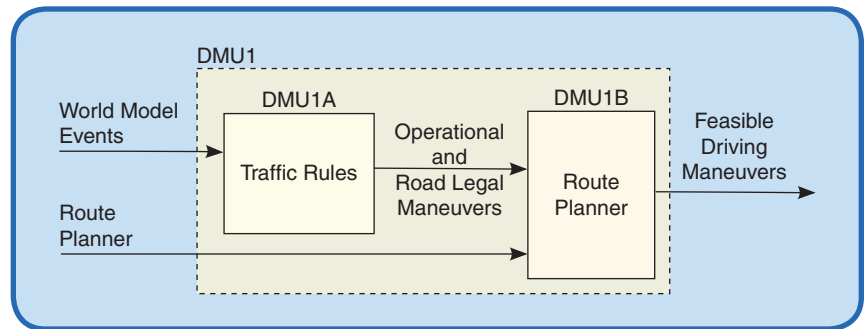

FIG 5 The decision making unit for the selection of feasible driving maneuvers.

The structure of the Petri net modeling DMU1 (Figure 6) is as follows. The input to the Petri net consists of two sets of input places. The first set of input places represents World Model Events; each World Model Event is represented by one single input place. The second set of input places represents the route planner indication; each route planner direction indication is represented by one input place. The output of the Petri net modeling DMU1 represents driving maneuvers. There is one output place of the Petri net for each available driving maneuver.

In each execution cycle the World Model and the Route Planner mark the Petri net's input places. A token is placed by the World Model into each place which represents a World Model Event which has the value *true*. Similarly, the Route Planner marks the input places which correspond to the planned travel direction.

The purpose of subnet DMU1A is to select those driving maneuvers which are operational (i.e., possible due to sufficient information) and which conform to traffic rules. Therefore, the knowledge and application of traffic rules is embedded in the execution structure of the subnet DMU1A, which marks only those places which represent operational driving maneuvers. The result of DMU1A is passed to the second subnet DMU1B.
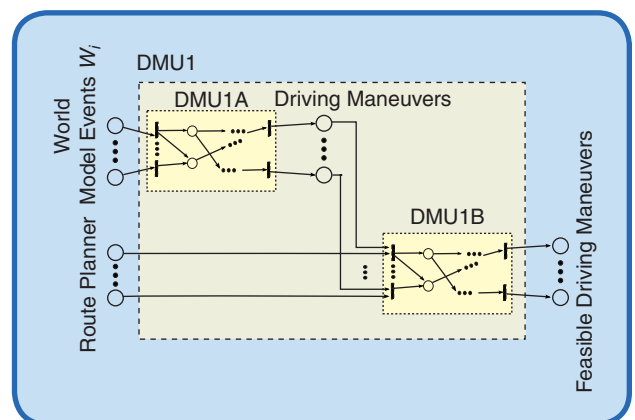

FIG 6 The decision making unit for the selection of feasible driving maneuvers and its two steps, each modeled by the subnets DMU1A and DMU1B, respectively.

The purpose of the subnet DMU1B is to filter out those driving maneuvers which were determined as operational by DMU1A, however which do not lead the vehicle into the direction indicated by the route planner. Therefore, the subnet DMU1B receives both inputs from the route planner and inputs from the subnet DMU1A. After its execution, DMU1B places a token into each Petri net output place which represents a driving maneuver which is both operational and according to the route planner indication.

After the execution of the complete Petri net DMU1, only those Petri net output places are marked, which represent feasible driving maneuvers (i.e., operational and according to the route planner). After each decision making cycle, all marked places of the Petri net are cleared, and a new decision making cycle begins.

In the current prototype implementation, since this is the safest option, in the unlikely event that there is no feasible driving maneuver, the vehicle stops, and waits for a maneuver to become feasible. However, with a complete system specification, which defines how the vehicle should respond to any traffic situation, and a complete list of required driving maneuvers, this will be avoided.

Besides its scalability, the main benefit of the Petri net model is that it allows to model, analyze, and verify the correctness of a very complex operational behavior of the first, safety-critical decision making stage, including a large number of World Model events (Petri net input places), which represent real-world events in the vehicle's traffic environment. This allows the decision making subsystem to deal with very complex real-world traffic situations. Furthermore, since the Petri net structure is decoupled from the control software implementation, and loaded from an external XML file, only its execution is implemented in the vehicle control software source code. Consequently, changes of the decision making operational behavior (in the Petri net XML file) do not require changes of the source code, and this in turn minimizes the possibility to introduce new software errors.

C. Decision Stage 2: Selecting the Most Appropriate Driving Maneuver Using MCDM

The goal of the second decision making stage is to select and execute the most appropriate alternative from those driving maneuvers which have been determined to be feasible in the current traffic situation.

Each of the feasible driving maneuvers offers multiple *execution alternatives*, which can be selected through discrete driving maneuver parameters. For instance, the overtaking maneuver could be performed at low or high speed, in distant or close proximity to the front vehicle, and on the right or left hand side. In order to select the most appropriate driving maneuver, and for it the most appropriate execution alternative, we apply Multiple Criteria Decision Making (MCDM) as follows.

Objectives: we define a hierarchy of objectives starting from a main, most general driving objective, which is then further successively broken down into more specific and therefore more operational objectives on lower hierarchy levels. Eventually, the bottom level of the objective hierarchy contains only objectives obj_j which are fully operational and which are measurable through their attributes.

The most general objective for autonomous driving is to safely reach the specified destination. More precisely, this objective is broken down into a lower hierarchy level containing more specific objectives, which specify how to achieve the objective of the higher level.

Thus, we define the following example objective hierarchy consisting of four ($k = 4$) level 2 objectives:

- Drive to destination safely $=: obj^{\text{Level1}}$
  - Stay within road boundaries $=: obj_1^{\text{Level2}}$
    keep distance to right boundary $:= attr_1$
    keep distance to left boundary $:= attr_2$
  - Keep safety distances $=: obj_2^{\text{Level2}}$
    keep distance to front vehicle $:= attr_3$
    keep distance to moving obstacles $:= attr_4$
    keep distance to static obstacles $:= attr_5$
  - Do not collide $=: obj_3^{\text{Level2}}$
    keep minimum distance to obstacles $:= attr_6$
    drive around obstacles $:= attr_7$
    avoid sudden braking $:= attr_8$
    avoid quick lane changes $:= attr_9$
  - Minimize waiting time $=: obj_4^{\text{Level2}}$
    maintain minimum speed $:= attr_{10}$
    avoid stops $:= attr_{11}$

Attributes: a set of measurable attributes

$$\{attr_1, attr_2, \ldots, attr_p\}, \quad p \in \mathbb{N} \tag{3}$$

is assigned to each objective on the lowest hierarchy level (in our example $p = 11$). An attribute is a property of a specific objective. In order to define various levels of importance, weights may be assigned to each attribute.

Alternatives: in the context of our application, decision alternatives correspond to the execution of driving maneuvers. Therefore, in a first step, we regard each element of the set of driving maneuvers $\{M^1, M^2, \ldots, M^n\}(n \in \mathbb{N})$ to be an element of the set of alternatives A:

$$A = \{M^1, M^2, \ldots, M^n\} \tag{4}$$

However, each driving maneuver $M^m (1 \leq m \leq n)$ offers one or multiple *execution alternatives* by specifying discrete parameter values (e.g., fast/slow, close/far, etc.). The driving maneuver parameters correspond in MCDM terms to decision variables, where each alternative is represented by a decision variable vector.

We obtain:

$$M^1 = \{M_1^1, M_2^1, \ldots, M_j^1\}$$
$$M^2 = \{M_1^2, M_2^2, \ldots, M_k^2\}$$
$$\vdots$$
$$M^n = \{M_1^n, M_2^n, \ldots, M_l^n\} \qquad (5)$$

where n denotes the number of driving maneuvers, and j, k, l the number of execution alternatives for the maneuvers M^1, M^2, and M^n respectively.

Therefore, the set of alternatives A contains all execution alternatives of all n driving maneuvers:

$$A = \bigcup_{m=1}^{n} M^m$$
$$= \{M_1^1, M_2^1, \ldots, M_j^1, M_1^2, \ldots, M_k^2, \ldots, M_1^n, \ldots M_l^n\} \qquad (6)$$

For the sake of readability, we denote all alternatives as:

$$A = \{a_1, a_2, \ldots, a_q\}, \ (q = j + k + \cdots + l) \qquad (7)$$

Utility Functions: utility functions $f_1(a_r), \ldots, f_p(a_r)$ specify the level of achievement of an objective by an alternative $a_r \in A \, (r \in [1, q])$ with respect to each of the p attributes.

For each attribute $attr_i \, (i \in [1, p])$, we define a utility function $f_{attr_i} = f_i$:

$$f_i : A \to [0, 1] \qquad (8)$$

Consequently, defining utility functions f_i for all alternatives $a_1, a_2, \ldots, a_q$ and all attributes $attr_i \, (i \in [1, p])$ results in the following decision matrix:

$$
\begin{array}{ccccc}
a_r & attr_1 & attr_2 & \cdots & attr_p \\
a_1 & f_1(a_1) & f_2(a_1) & \cdots & f_p(a_1) \\
a_2 & f_1(a_2) & f_2(a_2) & \cdots & f_p(a_2) \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
a_q & f_1(a_q) & f_2(a_q) & \cdots & f_p(a_q)
\end{array}
$$

The remaining task is to calculate a best solution among the feasible alternatives. A variety of MCDM methods can be applied in order to solve this problem, such as dominance methods, satisficing methods, sequential elimination methods, or scoring methods (Yoon and Hwang, 1995).

In the following example we choose a widely used scoring method, the Simple Additive Weighting Method, in which the value $V(a_r)$ of an alternative a_r is calculated by multiplying the utility function values with the attribute weights and then totaling the products over all attributes (see equation 11) (Yoon and Hwang, 1995). The alternative with the highest value is then chosen. Besides being easy to calculate, the Simple Additive Weighting Method allows to indicate the level of importance of certain attributes using weights.

Instead of using the Simple Additive Weighting Method, other methods can also be applied, in order to seek a Pareto optimal (noninferior) solution, i.e., a solution where no other alternative will improve one attribute without degrading at least another attribute (Chankong and Haimes, 1983). This comparison has to therefore be performed in the context of finding safety-critical solutions. Also, having in mind the decision making algorithm's hard real-time requirements which in turn are safety-crucial, the benefits of finding a Pareto optimal solution will be achieved if and only if the necessary computing power is available.

D. Example for Decision Stage 2

In this example we assume the traffic situation where a driverless vehicle decides about passing a stopped vehicle. Without oncoming traffic, the first decision making stage determined the following two driving maneuvers as *feasible:* Passing the stopped vehicle, or Stop&Go (i.e., waiting behind the temporarily stopped vehicle).

For the sake of simplicity, we assume that only the following few execution alternatives for the two driving maneuvers are possible:
- Passing maneuver M^1:
  - $a_1 := speed$ = slow, *lateral distance* = small
  - $a_2 := speed$ = slow, *lateral distance* = large
  - $a_3 := speed$ = fast, *lateral distance* = small
  - $a_4 := speed$ = fast, *lateral distance* = large
- Stop and Go maneuver M^2:
  - $a_5 := distance\ to\ front\ vehicle$ = small
  - $a_6 := distance\ to\ front\ vehicle$ = large

Consequently, the set of feasible alternatives is:

$$A = \{a_1, a_2, \ldots, a_6\} \qquad (9)$$

The utility functions $f_i(A)$ evaluate the achievement level of each attribute i for each of the 6 alternatives. In order to allow comparisons between the levels of achievement of different objectives, the values of the utility functions f_i are scaled to a common measurement scale, the interval of real numbers between 0 and 1. We define:

$$f_i \in [0, 1] \subset \mathbb{R}, \qquad (10)$$

where the value 1 denotes the optimal achievement of an objective, while 0 denotes that the objective is not achieved at all.

We define the utility functions as follows. Each of the 6 alternatives are rated regarding on how well they fulfill the driving objectives on the lowest hierarchy level. We rate the alternatives on a scale from 0 to 1, where:
- 1 denotes optimal fulfillment of the objective,
- 0.75 denotes good fulfillment,
- 0.5 denotes indifference,

- 0.25 denotes bad fulfillment,
- 0 denotes unsatisfactory fulfillment.

In our example, the utility function values are assigned based on heuristics reflecting the preferences of a human driver, as listed in Table 1.

For calculating the best solution, we choose in this example the Simple Additive Weighting Method (Yoon and Hwang, 1995). We define the value of an alternative a_r as follows:

$$V(a_r) := \sum_{j=1}^{p} w_j f_j(a_r), \tag{11}$$

where p denotes the number of attributes.

Each attribute is assigned a weight w_j, which reflects its importance. For autonomous driving, the importance of various objectives changes depending on the road conditions. For example, on a wide boulevard at higher speed, the attribute "$attr_8$: avoid sudden braking" is more important than the attribute "$attr_1$: keep distance to right boundary." However, in a residential area, the opposite might be the case. Consequently, instead of defining invariable attribute weights, this method offers the possibility to adapt the attribute weights, and therefore the decision preferences, according to the current traffic environment. In our example, we define the attribute weights as listed in Table 1.

Using the utility functions and attribute weights as listed in Table 1, we calculate the value of each alternative:

$$
\begin{aligned}
V(a_1) &= \sum_{j=1}^{11} w_j f_j(a_1) \\
&= 1 * 1 + 1 * 0.5 + 2 * 0.5 + 1 * 0.5 + 1 * 0.25 \\
&\quad + 1 * 0.25 + 1 * 1 + 3 * 0.5 + 2 * 0.75 + 2 * 0.75 \\
&\quad + 2 * 1 = 11.0
\end{aligned}
$$
$$V(a_2) = 12.25; V(a_3) = 12.0; V(a_4) = 13.25;$$
$$V(a_5) = 4.5; V(a_6) = 8.0 \tag{12}$$

The highest value $\max_{1 \le i \le 0} V(a_r) = 13.25$ is achieved by alternative a_4 (passing at fast speed with a large lateral distance to the stopped vehicle). Therefore, this alternative is chosen for execution. Since there is no vehicle oncoming, and the road is sufficiently wide, a human driver would very likely choose the same alternative.

E. Ensuring Real-Time Performance

One of the most important aspects for the decision making process is its ability to perform in real-time, i.e. to deliver correct results within specified time limits. However, in this application, the real-time requirements depend on a variety of factors, such as the vehicle's speed, the surrounding environment, etc. At high vehicle speed, the decision making subsystem, and all other safety-relevant components, need to react faster than at slow vehicle speed. On the other side, even at low speed, while driving in an urban area with pedestrians nearby, a quicker reaction time is needed, in order to avoid collisions.

Since the goal is to achieve a driving performance similar to a human driver, however with improved safety, it can be estimated that the overall guaranteed response time of the entire autonomous vehicle control system needs to be at least as short as the typical reaction time of a human driver. For the calculation of the autonomous vehicle's control system response time, all hardware and software subsystems need to be included. Therefore, since it is a critical subsystem of the autonomous vehicle's control software, the decision making subsystem's real-time performance needs to be ensured.

The developed decision making approach facilitates the analysis, simulation, and testing of real-time performance in the following ways:

- Decision Stage 1: The Petri net, which models the first, safety-critical stage of the decision making process can be analyzed, simulated, and tested independently from the rest of the autonomous vehicle's control system components. This also allows to measure the execution

Table 1. Heuristic definition of utility functions $f_j : A \to [0, 1]$ for the six alternatives $a_1, \ldots, a_6$ and 11 attributes $attr_1, \ldots, attr_{11}$. Weights indicate the level of importance. The column $V(a_r)$ lists the calculated values for each alternative a_r.

a_r	$attr_1$	$attr_2$	$attr_3$	$attr_4$	$attr_5$	$attr_6$	$attr_7$	$attr_8$	$attr_9$	$attr_{10}$	$attr_{11}$	$V(a_r)$
a_1	1	0.5	0.5	0.5	0.25	0.25	1	0.5	0.75	0.75	1	11
a_2	1	0.25	0.5	0.5	1	1	1	0.5	0.75	0.75	1	12.25
a_3	1	0.5	0.5	0.5	0.25	0.25	1	1	0.25	1	1	12
a_4	1	0.25	0.5	0.5	1	1	1	1	0.25	1	1	**13.25**
a_5	0.5	0.5	0.25	0.5	0.25	0.25	0	0	1	0	0	4.5
a_6	0.5	0.5	1	0.5	0.75	1	0	0.25	1	0	0	8
Weight	$w_1 = 1$	$w_2 = 1$	$w_3 = 2$	$w_4 = 1$	$w_5 = 1$	$w_6 = 1$	$w_7 = 1$	$w_8 = 3$	$w_9 = 2$	$w_{10} = 2$	$w_{11} = 2$	
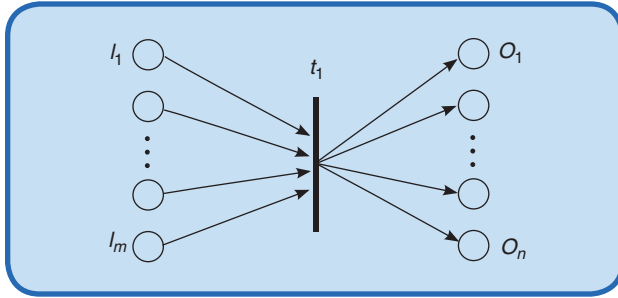
FIG 7 Petri Net with a single transition with multiple inputs and multiple outputs.

#Inputs	#Outputs	Exec. [ms]	Reset [ms]
100	20	0.4	1
1,000	20	3	2
5,000	20	16	2
10,000	20	31	20
100	100	0.5	1
1,000	100	4	1
5,000	100	16	2
10,000	100	32	31
100	1,000	2	0
1,000	1,000	5	0
5,000	1,000	18	8
10,000	1,000	34	36

times for the Petri net execution on the actual computing system, including the worst-case scenarios.

- Decision Stage 2: Although the second decision making stage does not include safety-critical attributes, its real-time performance ability does have an effect on the entire decision making process. Therefore, the number of MCDM objectives, the choice of the MCDM method, and for instance the calculation cost for seeking Pareto optimal solutions, need to be in line with real-time requirements. For this reason, the Simple Additive Weighting Method has been chosen in the implementation, due to its low calculation costs.

F. Real-Time Performance Measurements

In order to assess the real-time performance of the first, Petri Net based decision making stage, the Petri net implementation has been tested and its execution performance has been measured independently from the vehicle control software. For this purpose, two different Petri Net structures have been created, which reflect the building blocks of a complex decision making net:

- A Petri Net with a single transition with multiple inputs and multiple outputs (Figure 7), and;
- A Petri net with multiple transitions with single inputs and single outputs (Figure 8).

Single Transition with Multiple Inputs/Multiple Outputs. The first measured Petri Net structure consists of a single transition with a multiple inputs and multiple outputs (Figure 7). In order to assess the execution time of this structure, a large number of input places and output places has been created, the input places have been marked, and the execution time for the execution of the Petri Net (i.e., the firing of the single transition) has been measured.

Table 2 shows the average execution times with respect to the number of input and output places, as well as the time required to remove all Petri Net markings after the execution of the entire Petri Net (Reset). The removal of remaining markings after each execution cycle is required, in order to reset the Petri Net to its original state before the execution of a new decision making cycle.

Multiple Transitions with Single Input and Single Output. The second measured Petri Net structure consists of

multiple transitions, each with a single input and single output (Figure 8). In order to assess the execution time of this structure, a large number of transitions has been created, each connected by a single input and single output place, the Petri Net's input place has been marked, and the execution time for the execution of the Petri Net (i.e., the firing of all transitions) has been measured.

Table 3 shows the average execution times with respect to the number transitions. Since the execution times for the removal of remaining markings was below timing measurement limits (less than 1 ms) for the listed number of places, these values have been omitted.

The significantly longer and rapidly increasing execution times for a Petri Net with multiple transitions (Table 3) is due to the fact that in each Petri Net execution cycle all transitions need to be checked whether they are enabled or not. Enabled transitions fire by removing a marking from their input place and placing it into their output place, which in turn enables other following transitions to fire. The search for enabled transitions and their firing
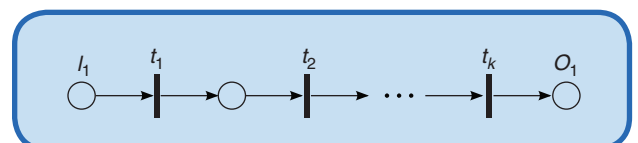

FIG 8 Petri Net with multiple transitions with a single input and a single output.

Table 3. Measured processing time in [*ms*] for the execution of a Petri Net with multiple transitions, each with a single input and single output. The measured execution times are average values from 1,000–10,000 executions (CPU: Intel T5450, 1.66 GHz).	
#Transitions	Exec. Time [ms]
10	5
20	7
50	38
100	78
200	296
500	1,600
1,000	6,300
2,000	25,000
5,000	160,000

is executed until there are no more enabled transitions, which results in increased processing times.

The measured execution time values listed in tables 2 and 3 can be used to estimate the maximum required processing times for a decision making Petri Net in a worst case scenario.

For example, the execution for a Petri Net consisting of 200 transitions with each 1000 input and 100 output places, will require on this specific hardware in a worst case (Table 2): $200 * 4$ ms = 800 ms.

Additionally, the time required to reset the net (i.e., to clear all markings) is around 1 ms for each transition. Therefore, the execution time for the entire Petri Net on this specific CPU is: 800 ms $+ 1$ ms $* 200 = 1000$ ms

Although the implemented prototype system does not include all required aspects for an autonomous vehicle to be fully operational in real-world traffic, the conducted tests attest that the developed approach is suitable to fulfill real-time requirements. So far, the developed software has been successfully tested on a Windows Vista notebook

FIG 9 On-road decision making experiment with oncoming traffic.

PC with an Intel T5450 CPU at 1.66 GHz with satisfactory results for the low vehicle speed of around 1 m/s. On this low-end notebook CPU and general purpose (i.e., non-real time) operating system, the decision making process was executed at 1–2 Hz, concurrently to all other vehicle control tasks on the same CPU. During these tests, the CPU load was relatively low (30–40%), which indicates that the decision making process, if executed on a dedicated CPU, under a real-time operating system, is able fulfill the real-time requirements for much higher vehicle speeds.

G. Error Recovery

Due to quickly and unexpectedly changing traffic conditions, in some situations, the decision making subsystem may need to abort the execution of certain driving maneuvers (such as overtaking), and switch to the execution of driving maneuvers specifically developed for error recovery. However, since the structure of error recovery driving maneuvers is identical to normal driving maneuvers, the process of error recovery does not require any changes of the decision making approach.

One of the challenges for the near future, and a so far not addressed question, is the development of a complete and detailed system specification for autonomous driving in urban traffic conditions, which foresees and includes how to deal with such unexpected traffic conditions.

IV. Test Results

The Decision Stage 1 outcomes have already been demonstrated in both 3-D simulation (Boisse et al., 2007) and in on-road experiments with a Cycab vehicle (Furda and Vlacic, 2009), where the decision about passing a stopped vehicle was repeatedly made correctly, with and without oncoming traffic (Figure 9).

The second stage of the decision making process has been evaluated in the 3-D simulation. We have conducted experimental tests for a variety of urban road traffic situations, however in the scope of this paper, only four situations are discussed. The following 11 attributes $attr_1, \ldots, attr_{11}$ have been used in the experimental tests (also addressed in subsection 3.3):

- keep distance to right boundary $:= attr_1$
- keep distance to left boundary $:= attr_2$
- keep distance to front vehicle $:= attr_3$
- keep distance to moving obstacles $:= attr_4$
- keep distance to static obstacles $:= attr_5$
- keep minimum distance to obstacles $:= attr_6$
- drive around obstacles $:= attr_7$
- avoid sudden braking $:= attr_8$
- avoid quick lane changes $:= attr_9$
- maintain minimum speed $:= attr_{10}$
- avoid stops $:= attr_{11}$

The list of the considered driving maneuver alternatives is as follows:

- a1 = GPS_Point2Point fast
- a2 = GPS_Point2Point medium speed
- a3 = GPS_Point2Point slow
- a4 = Intersection crossing fast
- a5 = Intersection crossing slow
- a6 = Pass slow/small distance
- a7 = Pass slow/large distance
- a8 = Pass fast/small distance
- a9 = Pass fast/large distance
- a10 = Platooning large distance
- a11 = Platooning small distance
- a12 = Emergency stop

In order to test the sensitivity of the developed Decision Stage 2 method, three different attribute weight distribution sets have been applied for each of these 11 attributes, as shown in Figure 10. The Weight Set 1 was chosen in such a way that the highest weight factors of 5 is assigned to both attributes 1 and 11, while the weight factor of 2.5 is assigned to attribute 6; the Weight Set 2 assigns increasing weight factors in steps of 0.5, while the Weight Set 3 assigns decreasing weight factors in steps of 0.5.

The outcomes of the first decision making stage are presented in Table 4, Column 2, as follows:
- Situation 1: the autonomous vehicle is following a road, relatively far from the coming intersection. In this situation, the first, Petri net based decision making stage has determined the driving maneuver for following a road using GPS coordinates, and its three execution alternatives (i.e., fast, medium speed, slow), as feasible.
- Situation 2: the autonomous vehicle is approaching an intersection, with a road crossing pedestrian. In addition to the road following maneuver, the first decision making determines the intersection crossing maneuver as feasible.
- Situation 3: the autonomous vehicle is following another vehicle. The first decision making stage determines the driving maneuvers "Pass" and "Platooning" as feasible.
- Situation 4: a static obstacle (a tree) is in front of the autonomous vehicle. The only feasible driving maneuver is the emergency stop.

The second, MCDM-based decision making stage calculates the values $V(a_i)$ for each of the feasible alternatives using Equation 11, which provided in this case the following results (Table 4, column 3):
- Situation 1: the use of the Weight Set 1 and Weight Set 2 result in the alternative $a1$ to be most appropriate (i.e., maximum value), while the Weight Set 3 results in the alternative $a3$ to be the most appropriate.
- Situation 2: the alternative $a4$ is the most appropriate for weight sets 1 and 2, while alternative $a3$ becomes the most appropriate if the Weight Set 3 is applied.
- Situation 3: the use of the Weight Set 1 results in the decision to execute alternative $a9$, while alternatives $a7$ and $a10$ are chosen for weight sets 2 and 3 respectively.
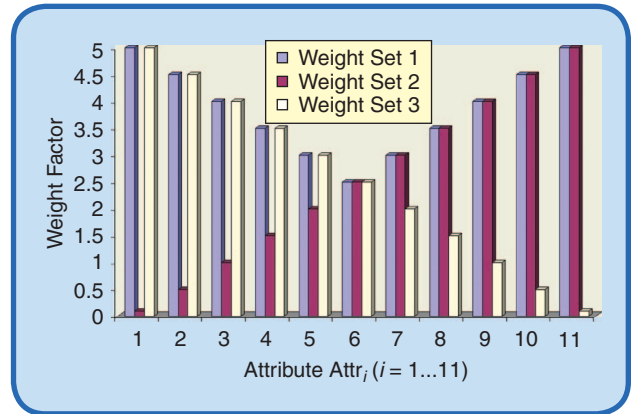

FIG 10 MCDM attribute weight distributions applied to the 11 attributes.

- Situation 4: the only feasible driving maneuver is the emergency stop. In our implementation the emergency stop is not evaluated by the MCDM based stage, but is instead immediately executed whenever it is determined to be the only feasible alternative.
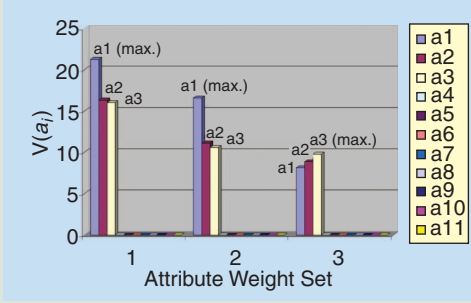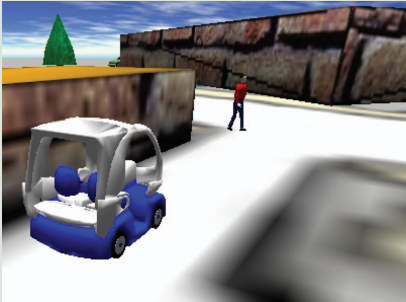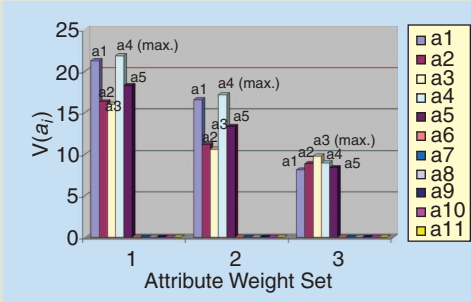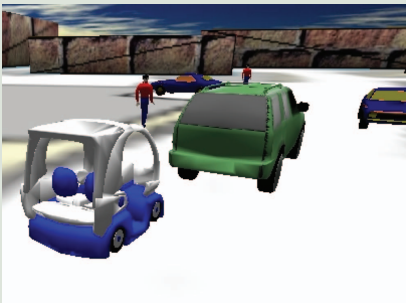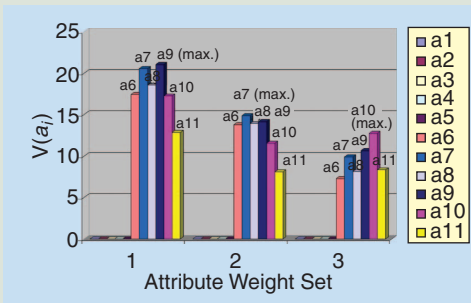
As expected, whenever the World Model fails to provide accurate information, such as for instance information regarding oncoming vehicles (e.g., Table 4, Situation 3), the first decision making stage may make the wrong decision about the feasible driving maneuvers. This error is then further passed on to the second stage, and may result in inappropriate or even unsafe driving decisions. Therefore, since it is mainly responsible for safety aspects, the first decision making stage has a crucial impact on the entire decision result.

On the other side, the MCDM results in Table 4 show that even when the attributes and utility functions of the second, MCDM based decision making stage have not been specified appropriately, for instance by assigning weights too high to irrelevant attributes, the resulting driving decisions are still safe. For example, applying the first set of attribute weights in Situation 1 results in the decision to approach the intersection at fast speed (i.e., alternative 1), while applying the third attribute weight set results in the more appropriate decision to approach the intersection at a lower speed (i.e., alternative 3).

Consequently, the developed decision making approach delivers correct decision results under the following conditions:
- accurate and sufficient information is provided by the World Model in real-time, especially regarding the MCDM attributes;
- the Petri Net based logic of the first decision making stage is defined according to a complete specification (i.e., a specification which defines the decision logic for all urban traffic conditions);
- the MCDM attributes and their weights are specified according to their importance as judged by the transport system experts.

	DM Stage 1 Results (Feasible)	DM Stage 2 Results (MCDM)

Situation 1: Following Road

a1 = GPS_Point2Point fast
a2 = GPS_Point2Point med. speed
a3 = GPS_Point2Point slow

Situation 2: Intersection

a1 = GPS_Point2Point fast
a2 = GPS_Point2Point med. speed
a3 = GPS_Point2Point slow
a4 = Intersection crossing fast
a5 = Intersection crossing slow

Situation 3: Following Vehicle

a6 = Pass slow/small distance
a7 = Pass slow/large distance
a8 = Pass fast/small distance
a9 = Pass fast/large distance
a10 = Platooning large distance
a11 = Platooning small distance

Situation 4: Static Obstacle

a12 = Emergency Stop
(not considered in MCDM evaluation)

Only Emergency Stop as feasible alternative.
⇒ Decision: Emergency Stop

A. Future Work

While the prototype implementation and the presented evaluation results demonstrate that the developed decision making approach is applicable and suitable, additional work is necessary in order to advance its development towards commercial real-world applications.

In order to obtain unconditional reliable decision making results, the following aspects need to be addressed by transportation system experts:

- The minimal set of traffic environment information provided by the World Model needs to be refined in the context of making safe driving decisions.
- The currently developed set of driving objectives needs to be expanded to unquestionably reflect the system specification.
- Additional research is required for the development of a complete set of driving maneuvers for urban traffic, in order to enable real-time decision making and autonomous driving in any situation.

V. Conclusion

This paper has addressed and presented a solution for the task of Real-Time Decision Making for autonomous city vehicles using Petri nets and MCDM. The decision making task has been divided into two consecutive stages. While the first decision making stage is safety-critical and focuses on selecting the feasible and safe driving maneuvers, the second decision making stage focuses on non safety-critical driving objectives, such as improving comfort and efficiency. We have demonstrated a solution for the first decision making stage based on Petri nets, and we have designed and developed the MCDM model, which is applied in the second decision making stage.

The application of MCDM methods for the second decision making stage enables the consideration of a large number of driving objectives, including possible conflicting ones, and leads to a powerful and flexible solution for non-simplified urban traffic conditions. Furthermore, compared to so far existing solutions, which were however intended only for simplified traffic conditions, the application of MCDM in this new research area offers a variety of benefits with respect to the problem specification, decision flexibility, and scalability.

Acknowledgments

About the Authors
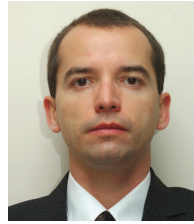
Andrei Furda received his Dipl.-Inform. degree in computer science from the University of Karlsruhe (TH), Germany, in 2004. Since 2006, he has been working towards his Ph.D. degree at the Intelligent Control Systems Laboratory (ICSL), Griffith University, Australia, under the supervision of Prof. Ljubo Vlacic.

Ljubo Vlacic is Professor and Director, Intelligent Control Systems Laboratory, Griffith University, Brisbane. In recognition of his contributions to advancement of control systems and their applications Professor Vlacic was named the 2003 Queensland Professional Engineer of the Year by the Queensland Division of the Institution of Engineers Australia; and also awarded: (i) the 2004 Sir Lionel Hooke Award, by the Australian Council of the Institution of Engineering and Technology—IET; (ii) the 2004 IEEE Achievement Medal (by the IEEE Knowledge Board); and a number of appreciation awards for notable services.

References

[1] S. Boisse, R. Benenson, L. Bouraoui, M. Parent, and L. Vlacic, "Cybernetic transportation systems design and development: Simulation software," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA'2007)*, Rome, Italy, 2007.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems,* 2nd ed. New York: Springer-Verlag, 2008.

[3] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making.* New York: Elsevier, 1985.

[4] DARPA. (2006). Urban challenge rules [Online]. Available: http://www.darpa.mil/grandchallenge/docs/ Urban_Challenge_Rules_121106.pdf

[5] A. Furda and L. Vlacic, "Towards increased road safety: Real-time decision making for driverless city vehicles," in *Proc. 2009 IEEE Int. Conf. Systems, Man, and Cybernetics*, San Antonio, TX, 2009.

[6] A. Furda and L. Vlacic, "An object-oriented design of a world model for driverless city vehicles," in *Proc. 2010 IEEE Intelligent Vehicles Symposium (IV 2010)*, San Diego, CA, 2010.

[7] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation,* 3rd ed. Pearson Education, Inc., 2007.

[8] J. Kolodko and L. Vlacic, "Cooperative autonomous driving at the intelligent control systems laboratory," *IEEE Intell. Syst.*, vol. 18, no. 4, pp. 8–11, 2003.

[9] L. Li and S. Tang, "Intelligent transportation systems in China," *IEEE Intell. Transport. Syst. Mag.*, vol. 1, no. 2, 2009.

[10] R. Li and L. Jia, "On the layout of fixed urban traffic detectors: An application study," *IEEE Intell. Transport. Syst. Mag.,* vol. 1, no. 2, 2009.

[11] M. Munz, M. Mahlisch, and K. Dietmayer, "Generic centralized multi sensor data fusion based on probabilistic sensor and environment models for driver assistance systems," *IEEE Intell. Transport. Syst. Mag.,* vol. 2, no. 1, pp. 6–17, 2010.

[12] J. L. Peterson, *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

[13] K. P. Yoon and C. L. Hwang, *Multiple Attribute Decision Making.* Newbury Park, CA: Sage, 1995.

[14] H. Zhao, J. Cui, H. Zha, K. Katabira, X. Shao, and R. Shibasaki, "Sensing an intersection using a network of laser scanners and video cameras," *IEEE Intell. Transport. Syst. Mag.*, vol. 1, no. 2, 2009.

ITS