Standardization of Generic Architecture for Autonomous Driving: A Reality Check

C. Guettier, B. Bradai, F. Hochart, P. Resende, J. Yelloz and A. Garnault

Abstract Autonomous driving requirements regarding safety, redundancy and performance lead to an increasing number of on-board heterogeneous equipments (sensors, actuators and processing units). Systems with the highest levels of autonomy must handle a large number of real-life situations, some of them being quite challenging. Interactions between sensors, perception, planning, control and actuators are fundamental to ensure operational efficiency while providing both proactive safety and timeliness execution. To address these constraints, appropriate design patterns are required to implement complex and robust decisional architectures. As a matter of fact, current system vehicle standards do not address these design neither patterns nor their underlying complexity. Based on various experiences in defense and civilian domains, we will present a critical review of ongoing standard developments and propose a generic design pattern to qualify autonomous vehicles architectures.

Keywords Autonomous architecture · Layered planning · Standardisation · Safety

Glossary

ADAS	Advanced Driver Assistance System
ADL	Architecture Description Language
AUTOSAR	Automotrive Open System Architecture
DDS	Data Distribution Service
GVA	Generic Vehicle Architecture
ISO26262	Road Vehicles—Functional Safety
JAUS	Joint Architecture for Unmanned System
PLEVID	Platform Level Extended Video Standard

C. Guettier (✉) · F. Hochart · J. Yelloz
SAFRAN—Sagem, 100 Avenue de Paris, Massy, France
e-mail: christophe.guettier@sagem.com

B. Bradai · P. Resende · A. Garnault
Valeo Driving Assistance, 34, Rue Saint André, 93012 Bobigny, France

J. Langheim (ed.), *Energy Consumption and Autonomous Driving*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-19818-7_7

ROS	Robot Operating System
UGV	Unmanned Ground Vehicle
US NREC	US National Robotic Engineering Center

1 Introduction

An autonomous car, also known as a driverless car, self-driving car, or robotic car is an autonomous road vehicle capable of fulfilling the transportation capabilities of a traditional car with minimal human input [1]. As an autonomous vehicle, it is capable of sensing its environment, and to perform navigation, guidance and control. Autonomous cars technology can improve our life with reducing both car accidents and traffic congestion. It can also discharge the driver from the monotonous daily driving, during which he can perform more productive tasks or just relax. From market point of view and end user acceptance, after automated parking manoeuvres and autonomous emergency braking, the next ADAS (Advanced Driver Assistance System) functions go one step further toward the automated driving vehicle (see Fig. 1).

Similar functionalities are expected for first responders, civilian security and defense autonomous vehicles. In those areas, Unmanned Ground Vehicles (UGV) must fulfill complex missions in spite of an unknown environment, resource constraints and with limited human interventions.

These autonomous systems must strongly adapt to their environment, whatever the circumstances are. The architecture design has to deal with perception, data fusion, mission planning and control. It must also scale up according to time frames and environment complexity. Minimizing human intervention in the system loop exacerbates the classical requirements of safety and availability. Also, such systems need close loop interleaving between embedded operation, situation awareness, planning and decision with finer grain real time control and command.

Various architectures have been developed by research teams in the framework of the DARPA Grand Challenges (2004–2007) [3] and in many European funded projects like HAVE-It [4] and V-Charge [5]. In Defense and Aerospace, many relevant architectures have been also proposed by NASA (see Remote Agent

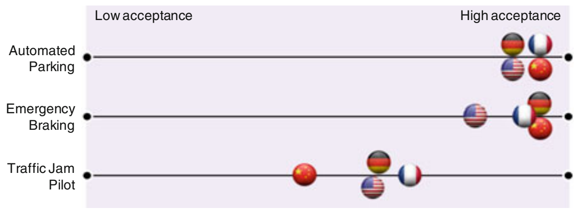


Fig. 1 End user acceptance of new functions

Architecture [9]) or ESA [10], and the US NREC, and through the UGV Demo I/II/III.

After presenting a classical decisional architecture in Sect. 2, the paper proposes to review existing standards in both civilian and defense automation in Sect. 3. The existing standards regarding land vehicles automation may be fundamentally divided in two types: architecture driven (AUTOSAR, GVA, JAUS, ROS) or functional safety requirements (ISO26262, or its counterpart in aerospace, the DO178). The impact of a decisional architecture for autonomous driven is discussed throughout a critical analysis in Sect. 4.

2 Layered Decision Making

2.1 Motivation

Automated driving is a global target, gathering different functional levels (see Fig. 2), that should be introduced step by step, according to the current legal framework, market and technology maturity [2].

Currently the most advanced automated driving level allowed by legal framework is Level 2. It includes temporary longitudinal and lateral control of the vehicle based on sensors and software logic. Although the vehicle is driven automatically the driver shall permanently supervise the good operation of the system and be in position to take immediate control if required [2].

Higher level of automation (Level 3 of automation or Conditional Automation) is currently being developed in order to further improve road safety, and driving comfort. For instance, in some situations, the automation can help the driving in demanding tasks like in a traffic jam or in monotonous tasks like in a long journey drive in a motorway. These functions are highly expected by end user.

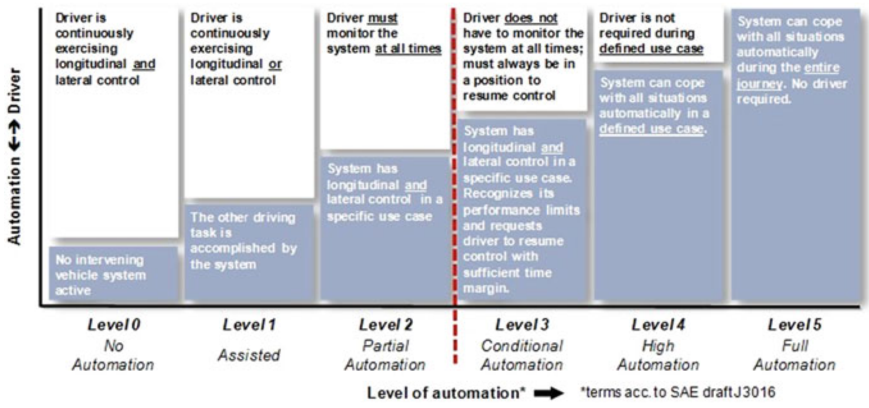

Automation ←→ Driver
Driver is continuously exercising longitudinal and lateral control
No intervening vehicle system active
Driver is continuously exercising longitudinal or lateral control
The other driving task is accomplished by the system
Driver must monitor the system at all times
System has longitudinal and lateral control in a specific use case
Driver does not have to monitor the system at all times; must always be in a position to resume control
System has longitudinal and lateral control in a specific use case. Recognizes its performance limits and requests driver to resume control with sufficient time margin.
Driver is not required during defined use case
System can cope with all situations automatically in a defined use case.
System can cope with all situations automatically during the entire journey. No driver required.
Level 0 No Automation
Level 1 Assisted
Level 2 Partial Automation
Level 3 Conditional Automation
Level 4 High Automation
Level 5 Full Automation
Level of automation*
*terms acc. to SAE draft J3016

Fig. 2 Automated driving levels

For the highest levels of Automation (Levels 4/5 or High/Full Automation), vehicle management needs to tackle different levels of situational awareness. At one extreme, localisation and real-time mapping of the proximal environment is required for fine manoeuvre. At the other extreme, the global traffic conditions are required to assess mission or journey feasibility. This enables a complete awareness of the situation that can be provided to other decision making and control components.

Most of the technologies are available to construct a situational awareness either for the proximal environment or for the global mission. Indeed, the vehicle and driver have access today to networking resources that enhance the knowledge about events (e.g. road blocks) that may affect the global mission.

2.2 Layered Planning Architecture

Due to the different problem scales, multiple planning levels are necessary to guarantee a consistent autonomous behaviour. The upper level of the planning deals with long term environment conditions and mission objectives. The lower level manages short term contingencies and local driving objectives (See [10, 11] for applications in aerospace autonomy). An intermediate level deals with routes and navigability. In this approach, each level has the ability to solve a planning problem at a given scale.

3 Standardization Efforts

3.1 Generic Vehicle Architecture (GVA)

GVA is an initiative from the UK MoD to standardize the architecture of military vehicles in order to reduce the total cost of ownership of a vehicle [12]. It includes maintenance, vehicle upgrades and evolutions, in order to take into account the vehicle heterogeneity. The key main principles or properties of the architecture are the following:

- Architecture is open, and can use of third-party software
- Scalability: considering the number and variety of equipment's connected to the vehicle bus
- Modularity: new sensors or even new services shall be easily added
- Reduction of integrations costs

The scope of the GVA is to standardize electrical; mechanical; human-machine and software interfaces. The overall architecture is structured around an independent communication bus for meta-information. The modularity of the architecture is based on a Service Oriented Architecture (SOA) approach. The architecture consists of several communication buses off different levels:

- 802.3 Ethernet bus for best effort traffic
- Time trigger Ethernet for deterministic traffic,
- MILCAN bus
- Energy bus
- Software bus based on the Data Distribution Service middleware (DDS) and the GVA data model (Land data model)
- Video bus based the PLEVID standard (video over IP)

End users equipment can also be connected through USB.

The middleware DDS is based on the Publish/Subscribe paradigm. Compared to others middleware, there is no point of failure and the transport layer is highly configurable. The weakness of DDS is that it covers only the transport layer.

There is no dedicated document to describe Safety requirements, but this class of properties is addressed in the system engineering method. Nevertheless, Health and Usage Monitoring of automation System techniques contribute to the safety of the overall system.

3.2 JAUS Standardisation

The Joint Architecture for Unmanned Systems (JAUS) standardisation process comprises a set of standards focusing on both system and operational independence of the robot [8]. JAUS comprises different sets of models, architecture designs, requirements and interfaces at vehicle, mission and systems levels. JAUS can be seen as a layered standard:

- Capabilities from the mission/user: JAUS SAE AIR 5665 provides an Architecture Framework for Unmanned Systems,
- Application level interfaces: SAE AS 5710 and SAE AS 6009 define those interfaces with unmanned system components.
- The JAUS Service Interface Definition Language (JSIDL, SAE AS 5684).
- The Link Layer level, JAUS Transport Specification (SAE AS 5669) defines transmission protocol over standard networks.
- Other JAUS standardisation processes define ontologies for architecture design (JAUS Service Interface Definition Language) and components as well as reference architecture (SAE Aerospace Information Report AIR5315—Generic Open Architecture GOA).

In JAUS, both safety and decision making requirements are not fully addressed and it is not possible to guarantee the vehicle autonomy behaviour. The JAUS standardisation process is not very active.

3.3 AUTOSAR

The increasing evolution of the vehicle applications leads automotive systems to a high level of complexity. In order to manage this issue, a development partnership of automotive manufacturers, suppliers and tool vendors has been organised to develop the Automotive Open System Architecture [6]. AUTOSAR objective is to create and establish an open and standardized automotive software architecture whose main goals are:

- Scalability to different vehicle and platform variants
- Transferability of software throughout network
- Support of different functional domains
- Increased use of COTS products
- Integration of different modules from multiple suppliers
- Support of applicable automotive international standards and state-of-the-art technologies
- Safety mechanisms considerations as well as dependability

To improve cost-efficiency and reusability, AUTOSAR separates application software from the associated hardware.

To achieve the technical goals of modularity, scalability, transferability and re-usability of functions, AUTOSAR provides a common software infrastructure for automotive systems of all vehicle domains based on standardized interfaces for the different layers as shown in the Fig. 3.

AUTOSAR basic concepts are:

- Standardization of the interfaces, application software components description, basic software and hardware abstraction layers
- Description of system and processing constraints
- Virtual functional bus which contains all communication mechanism and interfaces
- Functional mapping over processing resources and generation of their associated runtime environment and basic software configuration

AUTOSAR supports ISO 26262 standard and offers various safety mechanisms such as:

- End to end protection to ensure integrity of data transmitted through communication links
- Memory partitioning to prevent software components from interferences between them
- Basic software module defense behaviour to prevent from unauthorized service calls
- On-line Program flow monitoring to check correct software execution sequences
- Timing determinism/protection
- Dual microcontroller architecture

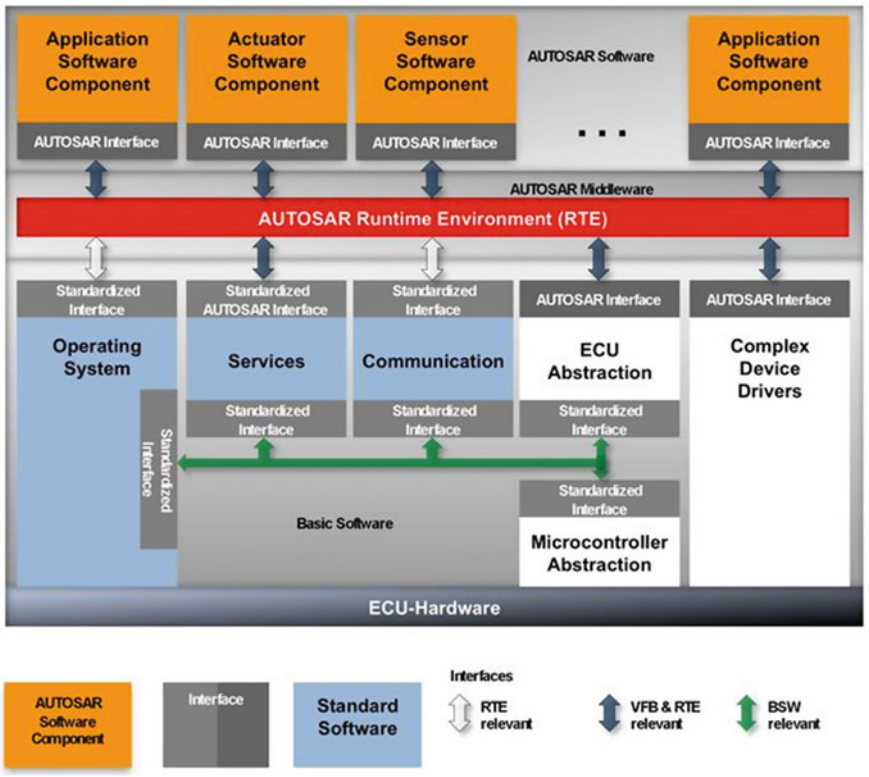


Fig. 3 AUTOSAR architecture

AUTOSAR combines the interest of an architecture description language (ADL) with partition-based execution, such as the ARINC 653 in aeronautic.

3.4 ROS

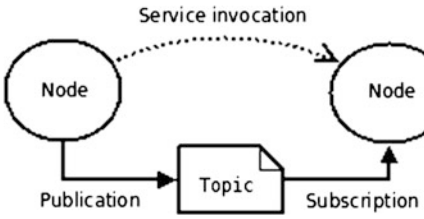
The Robot Operating System (ROS) is an open source distributed robotics platform designed to accelerate robotics research and development, including commercial application development. ROS is a high-quality, actively maintained, well-documented software platform intended to support the academic and industrial robotics communities. ROS includes reusable components that implement a variety of low- and high-level functionality, such as base navigation, mapping, visual odometry, arm planning and control, data visualization, object recognition, and task-level execution. ROS supports a number of research robots and common robot simulators.

All ROS software is released under an Open Source license, and the great majority of it is licensed under a BSD-style license that allows users and companies to build applications on top without licensing constraints.

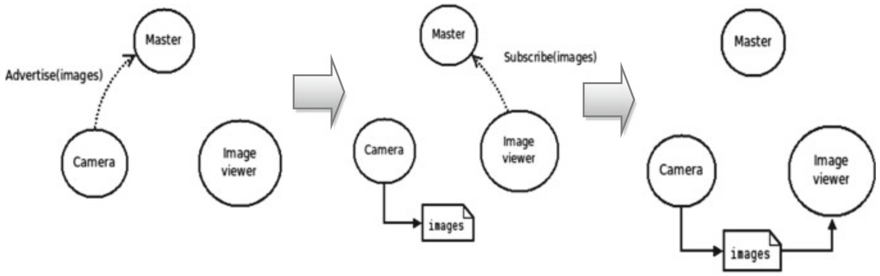
ROS has three levels of concepts: the Filesystem level, the Computation Graph level, and the Community level.

The Filesystem level concepts are ROS resources that you encounter on disk, such as Packages, Manifests, Stacks, Stack Manifests, Message Types and Service Types.

The Computation Graph is the peer-to-peer network of ROS processes that are processing data together. The basic Computation Graph concepts of ROS are Nodes, Master, Parameter Server, Messages, Services, Topics, and Bags, all of which provide data to the Graph in different ways. These concepts are implemented in the ros_comm stack.



Example of an “images” topic publisher/subscriber procedure using the ROS Master registration.



The ROS Community Level concepts are ROS resources that enable separate communities to exchange software and knowledge. These resources include Distributions, Repositories, ROS Wiki, Bug Ticket System, Mailing Lists, ROS Answers, and Blog.

Open topics like the support for fail-operational execution with dependable communication and firm real-time execution, model-driven development, quality management, safety qualification, cross-platform portability, and joint industrial

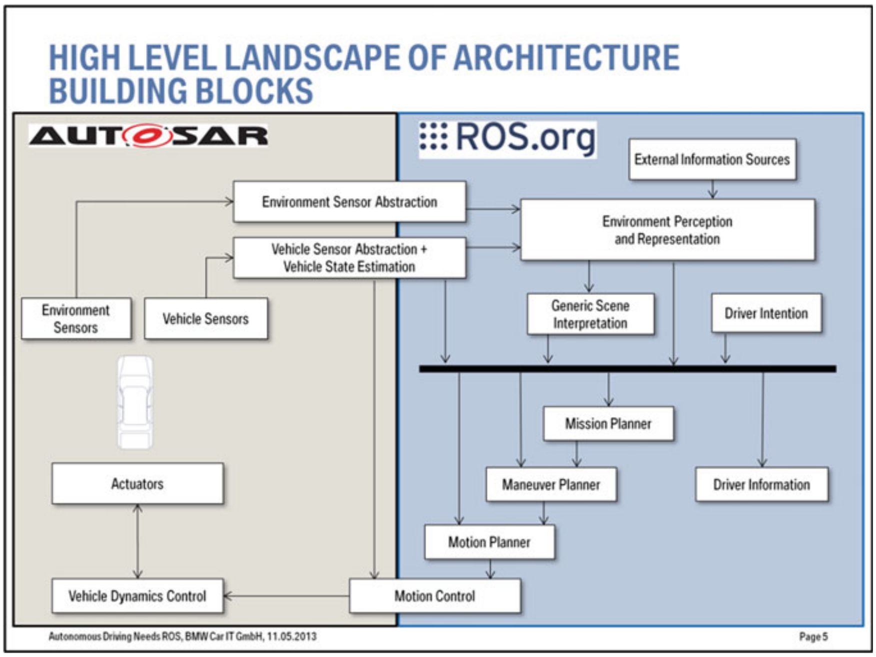

HIGH LEVEL LANDSCAPE OF ARCHITECTURE BUILDING BLOCKS
AUTOSAR
ROS.org
External Information Sources
Environment Sensor Abstraction
Environment Perception and Representation
Vehicle Sensor Abstraction + Vehicle State Estimation
Environment Sensors
Vehicle Sensors
Generic Scene Interpretation
Driver Intention
Mission Planner
Actuators
Maneuver Planner
Driver Information
Motion Planner
Vehicle Dynamics Control
Motion Control
Autonomous Driving Needs ROS, BMW Car IT GmbH, 11.05.2013
Page 5

Fig. 4 BMW architectural building blocks for autonomous driving

roadmap for development are already being addressed by the large ROS community, ROS for Products and Industrial ROS.

The following Fig. 4 by BMW Car IT GmbH shows the architectural building blocks for autonomous driving and how they could be distributed among the two frameworks, AUTOSAR and ROS.

4 Critical Analysis

4.1 Towards a Decisional Architecture Standard?

Several architectures have been specially designed and implemented to control autonomous road vehicles. A huge variety of functional decisional components has been proposed and demonstrated. Many of these architectures are often associated to a given development platform, lack maintenance and have little or no contribution to existing standards.

Indeed very few architectures have reached an industrial stage and no common design patterns for both functions and interfaces were established as an undeniable standard.

4.2 System Properties and Problems

The vehicle middleware has to guarantee the following properties:

- The safety property corresponds to the expectation that a system does not, under defined conditions, lead to a state in which human life or the environment is endangered
- The liveness property ensures that at some time the execution of a given tasks starts, progresses or ends
- The timeliness property adds timing requirements to the delivery of messages (latencies, delays, deadlines)

Today, most middleware, including DDS, do not address all these properties in a certifiable manner. Other drawback is that middleware's focus only on one layer (transport or just over the transport layer), the links with the lower layers are not specified (for instance MAC Layer or Service Admission). The list of problems to be resolved and for which it is necessary to prove the behaviour according to properties includes:

- Concurrency control algorithms, designed so as to run asynchronously over any number of processors; in spite of occurrence of partial failures at run-time.
- Synchronization of replicated processes whenever some failure occurs.
- Real-time process scheduling, where tasks that are subject to activation models and worst case execution time must fulfill deadlines.

These problems are amplified by decision making tasks (situation awareness, planning), which involve asynchronous data accesses and for which execution time are difficult to predict.

Not too surprisingly, current standardization of the state-of-the-art regarding these issues is limited. In most of autonomous architectures, these problems are related and addressed at the application layer and no generic design patterns have been proposed.

Concerning the ISO26262, the certification process can impose these properties. In general, those properties are defined by the client, which is validated using compliance tests. Some solutions are already proposed by providers, in general following AUTOSAR architecture standards.

4.3 Algorithm Complexity, Completeness and Determinism

One of the major problems is to bound execution time of the various tasks running in the systems. This is required for schedulability analysis such with ADL, or in some case to prove other system properties. Introducing situation awareness and planning algorithms lead to Non-Polynomial complexity. The designer must face the following paradigm:

Table 1 Standards comparison

	Type	Difficulties
GVA /DDS	Architecture driven	Autonomy/safety
JAUS	Architecture driven	Decision making/safety; deprecated
AUTOSAR	Architecture driven	Decision making
ROS	Middleware	Safety/portability
ISO 26262	Requirements driven	Autonomous systems complexity

- Non complete polynomial heuristic may be proposed, but the system may not provide a solution corresponding to the operational situation.
- Complete algorithms are proposed, but since they process tree search, it is difficult to provide deterministic delays (although, in some case, the operational semantic of the algorithm can be deterministic).

These problems have never been addressed in any standards able to deal with autonomous systems.

4.4 A New Systems Engineering Domain?

Each standard has its own scope and drawbacks and evolves in its own direction. No standard completely covers the domain. For an architecture driven approach, the standardization process must consider all relevant design patterns from the state of art in autonomous systems, for example the ones quoted in the paper. For a requirement driven approach; Safety Integrity Level (SIL) should include the formulation of new properties as well as assumptions that are specific to decision making tasks.

Current standardisation approaches can be positioned in the Table 1.

5 Conclusion

This paper has provided an overview of the ability of current standards to take into account decision making architectures such as layered planning for autonomous systems (levels 4 and 5 of automation).

It seems that autonomous systems must integrate a distributed executive that would contain some of the algorithms and protocols which solve problems briefly introduced in the above. This should yield to new practices to guarantee safety of autonomous functionalities.

Acknowledgments The authors acknowledge Safran and Valeo and their management for their full support in conducting this work.

References

1. http://en.wikipedia.org/wiki/Autonomous_car
2. Groult X, Picron V, Vejarano C, Barth H (2014) Active safety for the mass market. SIA VISION 2014
3. Seetharaman G, Lakhotia A, Blasch E (2006) Unmanned vehicles come of age: the darpa grand challenge. Computer 39(12):26–29
4. http://www.haveit-eu.org/
5. http://www.v-charge.eu/
6. http://www.autosar.org/
7. Heinecke H et al (2006) AUTOSAR—Current results and preparations for exploitation. In: Euroforum conference May 3, 2006
8. Rowe S, Wagner R (2008) An introduction to the joint architecture for unmanned systems (JAUS). Cybernet technical reports
9. http://ti.arc.nasa.gov/tech/asr/planning-and-scheduling/remote-agent/experiment/
10. Bornschlegl E, Guettier C, Poncet J-C (2000) Automatic planning for autonomous spacecraft constellations. In: Proceedings of the 2nd international NASA workshop on planning and scheduling for space, San Francisco, California, March 2000
11. Guettier C, Poncet J-C (2001) Multi-levels planning for spacecraft autonomy. In: Proceedings of the international symposium on artificial intelligence, robotics and automation, Montreal, Canada, June 2001
12. UK Ministry of Defence, "Generic Vehicle Architecture", NATO Defence Standard 23–09, Issue 1 Publication Date: 20 August 2010