Lot1 : Plateforme d'intégration système

L1.3 : System Architecture

Programme	FUI23
Référence	L1.3
Version	0.1
Date	04 / 12 / 2018
Porteur	UPPA
Auteur(s)	TBD
Contributeurs(s)	RT, Renault, INRIA, UPPA, IFFSTAR, Exoskill, Institut Pascal, UTC, Easymile

Financé par

bpifrance ÎledeFrance La Région Auvergne-Rhône-Alpes La Région Occitanie

Pôles de labellisation

aerospace valley mov'eo Imagine mobility ViaMéca

Authors

Name	Entity	Email
Arunkumar Ramaswamy	Renault	arunkumar.ramaswamy@renault.com
Ernesto Exposito	UPPA	Ernesto.exposito@univ-pau.fr
Sven Salomon	EasyMile	sven.salomon@easymile.com

Document History

Version	Update Date	Subject Changes	Author
0.1		Initial draft	

Table of Contents

Table of Figures

1 Introduction

Autonomous vehicles are complex systems, their deployment involves multiple stakeholders. A systematic system engineering approach is needed to deploy operational level-4 autonomous vehicles. This document presents the results from the 'System Need Analysis' and the 'Logical Architecture' phase of the Arcadia methodology. The purpose of System needs Analysis (SA) is to define the contribution expected of the system to users' needs, as they are described in the previous Operational Analysis (OA). The Logical Architecture (LA), which in our case can be called as 'Software Architecture', implements the big decisions of the solution, in terms of principles of construction, and ways to fulfill the expectations of stakeholders; it is then formalized by means of a decomposition into abstract components, namely principles of behavior and interface diagrams, etc.

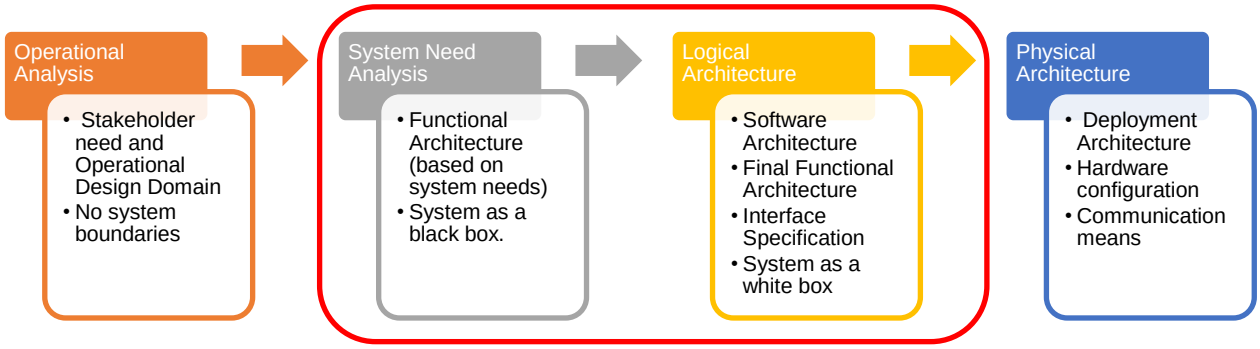

Figure 1 - Arcadia System Engineering Phases

2 Functional Architecture

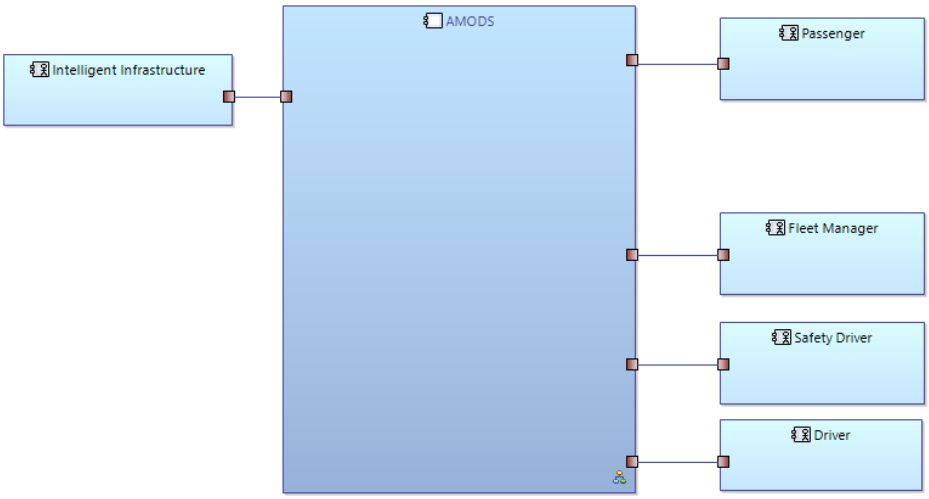

Figure 2 - AMODS system with external actors

The main decision that is made in the SA phase is defining the boundary of the system and identification of external actors. Figure 2 shows the Autonomous Mobility On-Demand System (AMODS) with external actors. In this phase, the analysis is intended to define the essential characteristics necessary for the fulfillment of each operational capability from OA phase, to study different alternative orientations likely to satisfy these required capabilities. At the SA phase, the system functions are decomposed as shown in Figure 3Figure 2 - AMODS system with external actors.

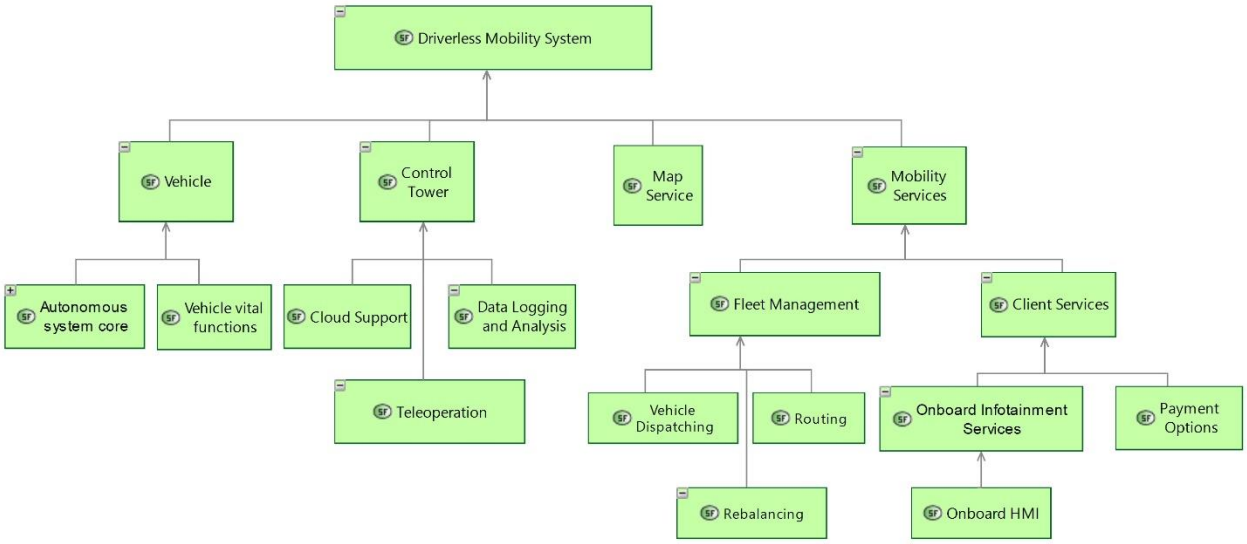

Figure 3 - AMODS System Functional Decomposition Diagram

3 Logical Architecture

Figure 4 illustrates the interaction between different vehicle providers and service providers. The AD vehicles communicate to the Ride Sourcing Provider (RSP) which does the management of fleet. The mobile application interacts with RSP for sending the mission for interacting with the passenger.

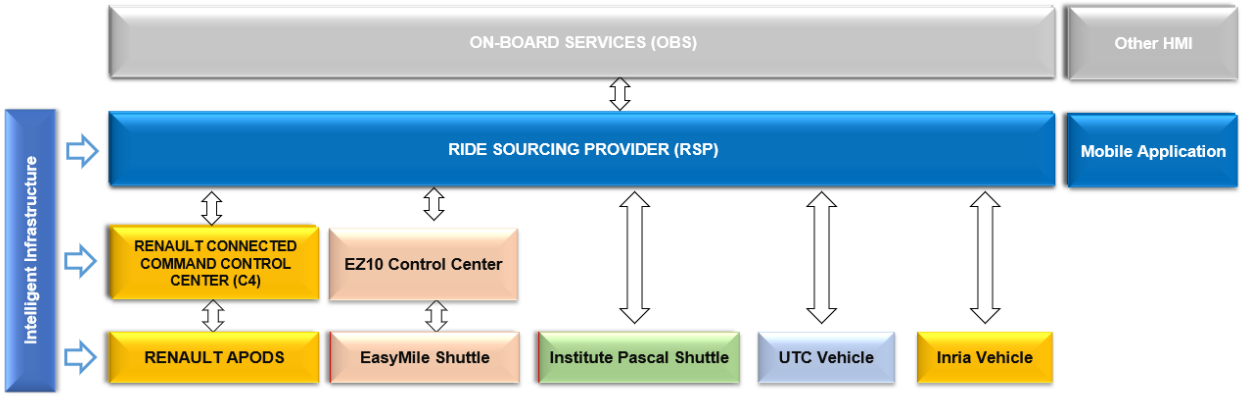

Figure 4 - Tornado system overview

3.1 Interface Diagram

The system discussed in previous section is shown as formal interface diagram in Figure 5. The infrastructure communication with the vehicle is defined in *V2X_Interface*. Currently only the data structure for traffic light communication is defined as shown in Figure 6.

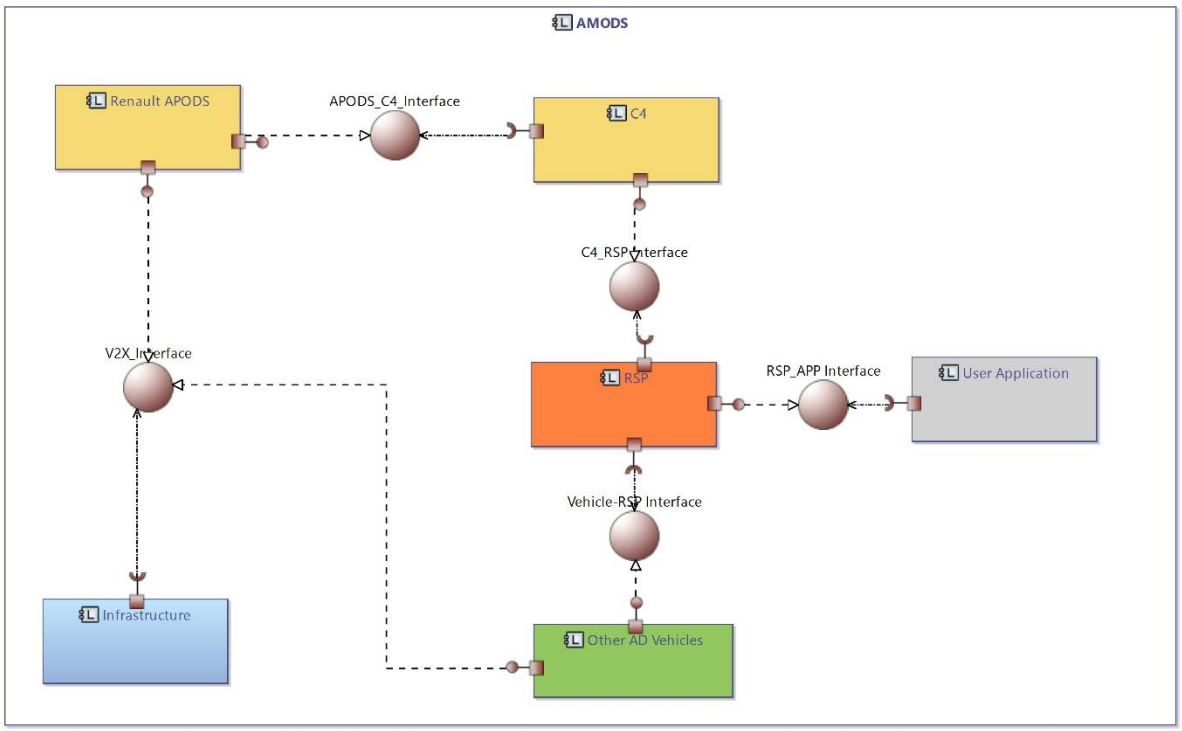

Figure 5 - AMODS System Internal Interface Diagram

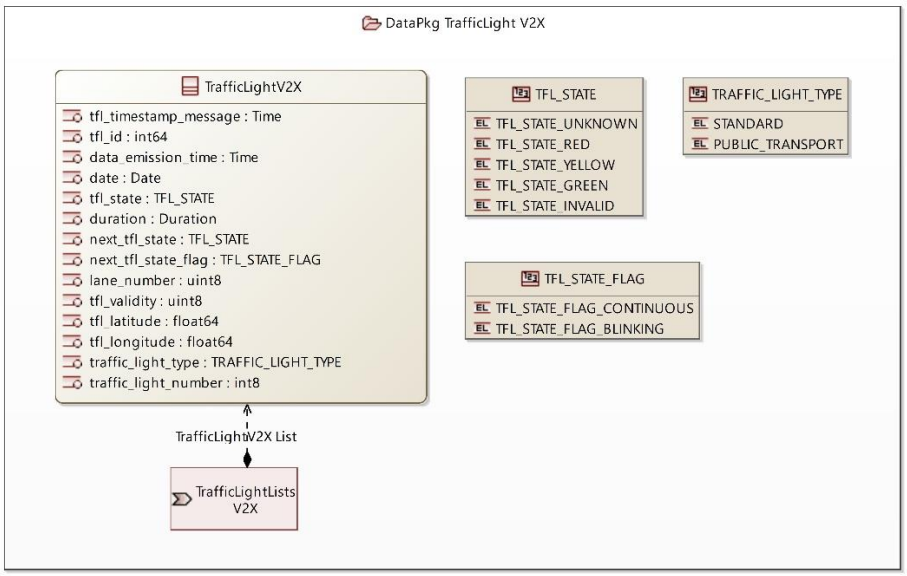

Figure 6 - Data structure of V2X Traffic Light data

The interface between RSP and Vehicle is defined by *C4_RSP Interface* (for Renault vehicle) and *Vehicle_RSP Interface* for other AD vehicle providers. The structure of C4_RSP Interface is shown in Table 1. The main function is *c4MissionSet* for sending the mission to the vehicle and *c4VehicleTrackingGet* for tracking the status of the vehicle. A very detailed specification of this interface is provided as a separate document [1-3]. In order to facilitate the development of fleet management software without access to real AD vehicle, a cloud-based vehicle simulator is also developed [4].

Table 1 - Renault C4 - EasyMile RSP Interface

Function name	Parameters	Description
c4StopListGet	RETURN - stopList [1..*] : Stop EXCEPTION - errorMsg [1..1] : ERROR_MESSAGE	To get the details regarding stop locations servicable by the vehicle
c4FleetTrackingGet	RETURN - vehicleStatusList [1..*] : VehicleStatus EXCEPTION - errorMsg [1..1] : ERROR_MESSAGE	To get the status of all the vehicles in the fleet.
c4VehicleTrackingGet	IN - vehicleID [1..1] : String RETURN - vehicleStatus [1..1] : VehicleStatus EXCEPTION - errorMsg [1..1] : ERROR_MESSAGE	To get the status of a specific vehicle in the fleet.
c4MissionSet	IN - vehicleID [1..1] : String IN - stopID [1..1] : String IN - location [1..1] : Position RETURN - missionAcknowledgement [1..1] : MISSION_ACKNOWLEDGEMENT EXCEPTION - errorMsg [1..1] : ERROR_MESSAGE	To send a mission to the vehicle. stopID is mandatory. location is optional and is reserved for future use cases.

c4TripSet	IN - vehicleID [1..1] : String IN - tripID [1..1] : String RETURN - acknowledgement [1..1] : TRIP_ACKNOWLEDGEMENT EXCEPTION - erroMsg [1..1] : ERROR_MESSAGE	To send a trip to the vehicle. (To be implemented in later stage)

The interface between RSP and Mobile application is defined in Figure 7.

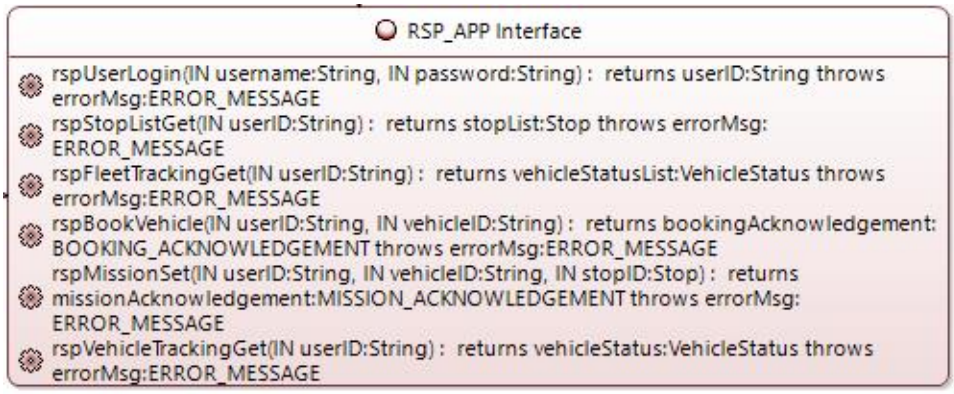

Figure 7 - Example Interface between RSP and Mobile Application

3.2 State Diagram

Figure 8 shows the state machine diagram that must be followed by each AD vehicle provider. It ensures the coherency of mission management by the fleet management provider.

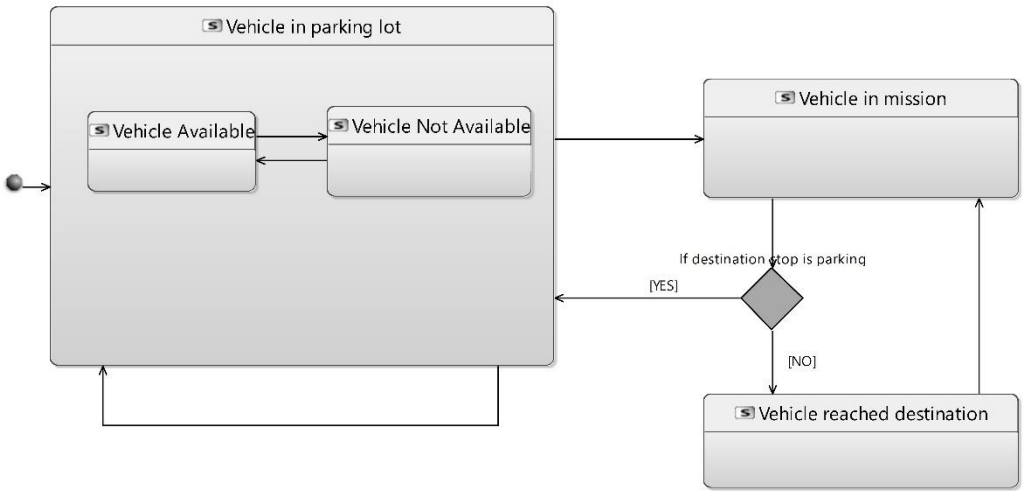

Figure 8 - Vehicle state diagram (Fleet Management-Centric view)

For the case of Renault vehicle, there fleet management provider have no direct communication with the vehicle but to C4 system. Hence this state machine is maintained by the C4 System.

The state machine diagram shown in Figure 9 details different vehicle states maintained by C4. The *availabilityStatus* indicate if the vehicle can accept a mission or not. The vehicle can stop at predefined stop locations only. We assume that the vehicle can be parked (or stopped for longer periods) only at stop locations where parking is capable. In normal conditions, the *availabilityStatus* shall be UNAVAILABLE while the vehicle is in parking location due to the technical constraints of the vehicle.

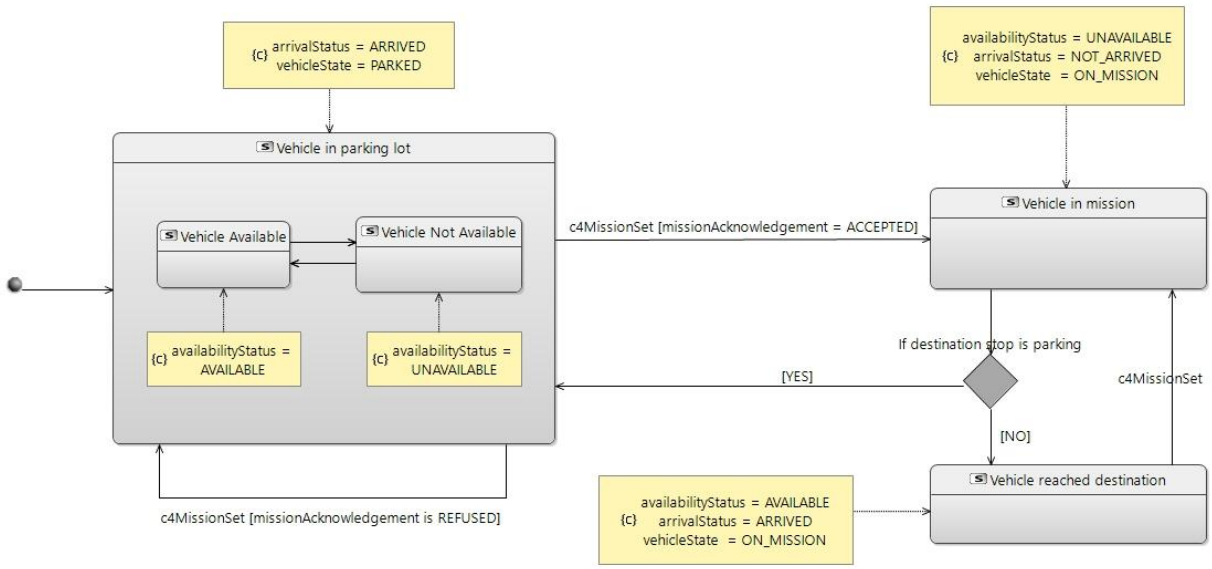

Figure 9 - Vehicle state maintained by C4 for Renault Vehicle

When the vehicle accepts a mission while in parking location, the *vehicleState* is changed from PARKED to ON_MISSION. If the destination stop is *parkingCapable*, the state variables after reaching the destination will be *arrivalStatus* = ARRIVED, *vehicleState* = PARKED. If the destination is not *parkingCapable*, the state variables after reaching the destination will be *availabilityStatus* = AVAILABLE, *arrivalStatus* = ARRIVED, *vehicleState* = ON_MISSION. It is to be noted that *vehicleState* is still ON_MISSION because an AD Vehicle could not be left 'unserviced' while it is in a stop location where parking is not possible. It is the responsibility of RSP to make sure the AD vehicle is parked at a parking capable location when the service of AD vehicle is no longer required. Hence the term 'mission' is with respect to the vehicle not with the provider. The *vehicleState* is changed to PARKED from ON_MISSION when it reaches the parking capable location. However, it is up to the RSP whether to send the vehicle to the parking location or to another a stop depending on the requirement. Therefore, ON_MISSION state cannot be bound to a specific trip or passenger.

4 Conclusion

The analysis or an architecture definition cannot be performed once, in a linear fashion. Most of the time, reflection will be gradually constructed, including increasingly more detailed reflection states (iterative approach), and/or a gradual extension of concepts to be considered, starting with

the most important and critical, then expanding to the full required specification. In this deliverable a very high-level of architecture definition is provided in order to consolidate the system functions and standardize the interfaces.

5 Appendix

1. Renault C4 – RSP Interface Specification Document: https://cloud.tornado-mobility.com/index.php/s/tNO82PKcFUOBvKp
2. Renault C4 Webservice Release Notes: https://cloud.tornado-mobility.com/index.php/s/uVNtfcj7KJNmS8t
3. Renault C4 API Reference: https://cloud.tornado-mobility.com/index.php/s/B2RyDz4wtZC1V0F
4. Renault Cloud Simulator: https://cloud.tornado-mobility.com/index.php/s/RNX99ToVbFfU5hL