Chapter 1
Overview of Systems Engineering

Juan R. Pimentel
Consultant

1.1. Introduction

Automated vehicles (AVs), also called autonomous or self-driving vehicles, have the potential to reduce accidents, help with the environment, reduce congestion, help the elderly and other disadvantaged populations, and produce other societal benefits [1-5]. However, the touted advantages of autonomous vehicles and those including latest ADAS features are turning out difficult to sell to the public than many manufacturers and tier 1's have anticipated. Much of the early euphoria of self-driving vehicles is diminishing in the wake of some recent accidents involving automated vehicles with varying degrees of automation. A recent online marketplace for buying and selling cars, found 69% of respondents are scared of autonomous automobiles. It also found that these people found technology in cars helpful (58%) but only 12% said ADAS and infotainment features were "must have". The survey asked more than 1,000 respondents from across the US geographically and across age groups, although it should be noted the biggest group was 60+ years-old[1]. Accidents involving self-driving vehicles are inevitable; as it is the case with other industries, accidents have happened and will happen no matter the efforts made to avoid them. The National Transportation Safety Board (NTSB) has issued a report[2] on a Tesla accident on May 7, 2016 and two preliminary reports on a Uber accident[3] on March 18, 2018 and a Tesla accident[4] on March 23, 2018. After analyzing these reports, what is disturbing are the details associated with these accidents which indicates that as an industry, we may need to go back to safety 101[5].

Regarding the aforementioned accidents, it would not be so bad if the safety systems of the vehicles in question were designed and functioning properly according to their stated automation level. In the case of the Tesla accident in Florida, the vehicle failed to activate the forward collision warning (FCW) system and automatic emergency braking (AEB). In the case of the Uber accident, emergency braking maneuvers were not enabled while the vehicle was under computer control, to reduce the potential for erratic vehicle behavior and the system relied on the vehicle operator for safety. In the Tesla accident in California, the vehicle failed to detect a damaged crash attenuator and hit it at a speed of about 71 mph.

After recent incidents and mishaps involving automated vehicles such as those described above, it is clear that there is much room for improvement not only by manufacturers but also by government regulations, researchers, the general public and other stakeholders. Over the past few months, the media has been full of headlines such as "How Safe Is Driverless Car Technology, Really?", "Autonomous Cars: How Safe Is Safe Enough?" and "How safe should we expect self-driving cars to be?" In addition, some industry analysts and safety experts are offering advice to tech and automotive companies to re-consider their safety programs. There is also some agreement that "the self-driving car industry's reputation has suffered a setback," and the question is how to fix it[5]. It appears that automated vehicle companies are much more stringent when using semiconductor devices and EDA tools demanding that they conform to ISO 26262 than using the same yard stick for their own safety critical designs.

[1]http://analysis.tu-auto.com/autonomous-car/shifting-public-acceptance-autonomous-tech?NL=TU-001&Issue=TU-001_20180723_TU-001_235&sfvc4enews=42&cl=article_2_2
[2] https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1702.pdf
[3] https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf
[4] https://ntsb.gov/investigations/AccidentReports/Reports/HWY18FH011-preliminary.pdf
[5]https://www.eetimes.com/document.asp?doc_id=1333446&_mc=RSS_EET_EDT&utm_source=newsletter&utm_campaign=link&utm_medium=EETimesWeekInReview-20180721

So what is there to do? Safety is not new, at least for the last 60 years it has been successfully applied in several industries such as nuclear, avionics, process control, automotive, and others. What is unique and special about the safety of self-driving vehicles? What should be the emphasis for a more effective automated vehicle safety program? What are the roles of governments, standards, testing, verification, validation, and sound safety engineering efforts? Addressing issues regarding autonomous vehicle safety is challenging [7]. Currently, as an industry, we just do not fully understand the nature of self-driving vehicle safety and how to design safe automated vehicles. For example, there is little discussion on ways to estimate, analyze, compute, or measure the level of safety of an automated vehicle design or automated vehicles. We need to begin by fully characterizing it and this book series is an effort in this direction.

Some manufacturers such as Waymo cite their recent milestone of 8 million miles driven on public roads as a measure of the safety achieved by their self-driving vehicles[6]. However, it is not clear how a certain number of millions of miles driven contributes to the safety level of self-driving vehicles. Some industry analysts believe that policy makers and city officials overseeing infrastructure will be the most important players in reshaping the self-driving vehicle safety landscape. For example, NHTSA has issued a voluntary guidance whose purpose is to help designers of automated driving systems (ADSs) analyze, identify, and resolve safety considerations prior to deployment using their own, industry, and other best practices[7]. It outlines 12 safety elements, which the Agency believes represent the consensus across the industry, that are generally considered to be the most salient design aspects to consider and address when developing, testing, and deploying ADSs on public roadways. Within each safety design element, entities are encouraged to consider and document their use of industry standards, best practices, company policies, or other methods they have employed to provide for increased system safety in real-world conditions. The 12 safety design elements apply to both ADS original equipment and to replacement equipment or updates (including software updates/ upgrades) to ADSs. However the NHTSA guidance is not specific enough to help manufacturers designing effective safety mechanisms to reduce risk.

1.2 Characterizing the Safety of Automated Vehicles

How different is the concept or notion of safety in self-driving vehicles when compared to that used in other industries such as aviation, process control, and automotive? While the fundamental concepts are the same, the safety of self-driving vehicles has specific attributes that are different or not present in the safety of other industries. In this section we briefly discuss these attributes. When compared to the safety of traditional industries such as avionics, process control and automotive, there are specific attributes pertaining to the safety of self-driving vehicles that we discuss next [23].

1.2.1. Performance degradation.

Traditional safety is based on faults and failures of mostly hardware components, and this is referred to as the reliability approach to safety [9]. In contrast, accidents involving self-driving vehicles might happen even if no hardware device fails, but rather a performance degradation of some of its functions or intended functionality occurs. Addressing safety issues for these situations is referred to as SOTIF, and it is a fairly new concept as applied to the safety of self-driving vehicles [10]. Thus, some failures are due to performance degradation of self-driving vehicle components, typically involving higher levels of processing or higher levels of automation, e.g., service failures. This definition of failure goes beyond that which is defined in the standard ISO 26262; however, it is compatible with other safety frameworks such as [8], STPA [11], [12], [13], [14], [15], or other real-time distributed systems [16]. One example of this understanding of the concept of safety is the failure of a vehicle detection system where the perception system provides missed detections (i.e., false negatives) or spurious detections (i.e., false positives). This could happen because the processing of environmental data is highly complex and the object detection function is subject to errors and impairments, particularly in bad weather or in night conditions when the

[6] https://www.theverge.com/2018/7/20/17595968/waymo-self-driving-cars-8-million-miles-testing
[7] https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf

visibility is poor. Either one of these failures could be catastrophic and could result in an accident or harm. Another example is a RAdio Detection And Ranging (RADAR) system correctly detecting objects only when the objects are moving, thus missing static objects because of limits on its performance. Thus, failure occurs in a degraded performance scenario.

1.2.2. Focus on software.

It is well known that the amount of software in a vehicle continues a rapidly increasing trend that started with the development of by-wire systems. Much of the functionality of a self-driving vehicle is implemented in software, and thus it is important to view the perception system as a set of software servers each providing services to the rest of the system. Therefore, one can refer to these various functionalities as a vehicle detection server, a pedestrian detection server, a road detection server, etc. The software in self-driving vehicles is much larger in size and scope compared to traditional industries; thus, there should be a focus on the safety of the software. As noted, a failure can occur if the software services deviate from the correct services and this could lead to safety hazards and safety risks. Ultimately, the overall safety of a self-driving vehicle will be dictated by the safety of its software [9], [12].

1.2.3. Non-deterministic perception system.

In the absence of hardware faults, the perception systems of traditional industries are mostly deterministic in nature. For example, sensing the intake manifold pressure or engine speed in automotive systems is deterministic4. In contrast, the perception systems of self-driving vehicles are non-deterministic, leading to a high level of false positives and false negatives when their performance deteriorates to the point that service failures cannot be avoided. The non-deterministic aspect of the perception system stems from the fact that one never knows when its performance will deteriorate to the point where failures begin to appear in the services delivered by the system. Thus, the services provided by the perception system are subject to random failures; e.g., when the weather deteriorates or when the system makes detection errors.

1.2.4. Perception system complexity.

Sensing elementary physical phenomena such as temperature or pressure is relatively simple, involving just some deterministic sensors, some electronics, and communications. In contrast, sensing or detecting man-made entities or constructs such as another vehicle, a road boundary, a city street, or a street intersection is complex because of the lack of structure of what is being sensed or perceived. The implication of the complexity of the perception system is that it is prone to errors, which degrades the performance or safety of the overall vehicle.

1.2.5. Overall system complexity.

In addition to its perception system, a self-driving vehicle also includes localization and mapping, planning and control, and actuation resulting in a highly complex system. One of the main issues with system complexity is that it makes testing for safety challenging, particularly if machine learning techniques are used, as it makes the design opaque to humans. This makes tracing the design and the test plans to the requirements problematic, since there is no human-understandable design that can be used for verification and testing [17]. In addition, it is known that when the system is complex, the system safety is affected by interacting complexity and tight coupling [9]. Another aspect of system complexity is that the autonomous vehicle operates in a complex external environment, and there are safety hazards due to events outside the domain of the autonomous vehicle, e.g., from other vehicles (whether self-driving or not). Thus, the safety attributes of a self-driving vehicle are significantly different from those in other industries such as avionics, process control, and automotive.

In addition to its attributes, what are the various types of safety that encompass the overall safety of self-driving vehicles? As noted, the safety of self-driving vehicles is complex and differs from that of other industries such as avionics, process control, and automotive. On the one hand, there are safety commonalities such as the safety that involves component failures, which is the subject of so-called *functional safety*, and the safety involving components whose failure rates are well understood because they are proven in use, i.e., in actual operation. On the other hand, there are two types of safety that are not prevalent in the avionics, process control, and automotive industries, and these include *SOTIF* and *multi-agent safety*. Thus, the types of safety that characterize the safety of self-driving vehicles include (1) Traditional functional safety as defined by ISO 26262, (2) SOTIF, and (3) multi-agent safety [23]. Feth et al also emphasize that safety assurance is a concern because established safety engineering standards and methodologies are currently not sufficient [22]. They also conclude that there are three types of safety that characterize the safety of self-driving vehicles, (1) Traditional functional safety, (2) SOTIF which they assume are due to *functional insufficiencies*, and (3) multi-agent safety related to safe driving behaviors which are abstracted from technological challenges of situation awareness. Furthermore, they elaborate the fundamental safety engineering steps that are necessary to create safe vehicle of higher automation levels while mapping these steps to the guidance presently available in existing (e.g., ISO26262) and upcoming (e.g., ISO PAS 21448 [26]) standards. Functional safety is a well understood area which is guided by a number of international standards such as IEC 61508 [18], IEC 61511 [19], and ISO 26262 [6], and there is a large number of papers and publications on this topic. However, it is noted that ISO 26262 does not cover automated vehicles, thus its application should be done with great care. In the following, we characterize the safety category of SOTIF.

1.3 **Role of Systems Engineering**

Many systems and products are very complex, very large, or both; certainly one example is an automated vehicle with a high degree of safety, thus the complexity of designing and developing an autonomous vehicle with a high degree of safety. Systems engineering (SE) is a methodology that includes a set of processes, techniques, and procedures to design systems that are very complex and/or very large. SE is based on the concept of a system lifecycle composed of a number of phases such as concept, product development, and production and operation. The concept phase includes many sub-tasks such as item definition, initiation of safety lifecycle, hazard analysis and risk assessment, and functional safety concept (FSC). The product development phase can be performed at a system, hardware, and software levels. Finally, the production and operation (or utilization) phase includes production, operation, service (maintenance, repair, etc.), and decommissioning.

The number and names of system lifecycle phases are not unique. Other phases identified with a system lifecycle include requirements phase, preliminary and detailed design phase, construction and production phase, and the operation phase. The requirements phase is an important phase that takes place at the beginning of a project that involves top level requirements at all functional levels. Some requirements are further developed or extended into "derived requirements". This process is repeated appropriately until reaching the lowest level possible. There are two types of requirements, functional and non-functional. Requirements need to be carefully selected in order to ensure that they make sense in the context of the final outcome of the project and conveyed to all the team members. Missing out on a requirement or misapplying one could spell disaster for a project. A functional requirement specifies what the system should do, i.e., it describes a particular behavior or function of the system when certain conditions are met, for example: "Send email when a new customer signs up" or "Open a new account." A functional requirement for an everyday object like a cup would be: "ability to contain tea or coffee without leaking. "Examples of some functional requirements include: Carry people on public streets and roadways, carry people through air space provide external interfaces, define business rules, perform transaction corrections, perform adjustments and cancellations, provide administrative functions, provide authentication, use authorization levels, perform audit tracking, etc. A non-functional requirement specifies how the system performs a certain function, i.e., it describes how a system should behave and what limits there are on its functionality; it specifies the system's quality attributes or characteristics, for example: "Modified data in a database should be updated for all users accessing it within 2 seconds." A non-functional requirement for the cup mentioned previously would be: "contain hot liquid without heating up to more than 45 ° C." Important categories of non-functional requirements include availability, safety, security, and reliability. The preliminary and detailed design phase includes intermediate and final designs with enough details and specifics for its implementation by other parties. It is based on requirements and involves assumptions, calculations, measurements, simulations, etc. The outcomes of this phase are produced using an assortment of tools in the categories of drawing, simulation, system engineering, etc. The construction and production phase is that where the system is built up using blueprints of the detailed design phase. When the construction is done in series, e.g., in a manufacturing plant, then it is termed production. Testing is an important activity to be done during construction. There are many types of tests such as unit test, system test, interoperability test, stress test, system test, etc. The operation phase starts when the system or product is put in service and ends when the system is taken out of service or decommissioned. It includes the following activities: operation, maintenance, troubleshooting, repair, etc.

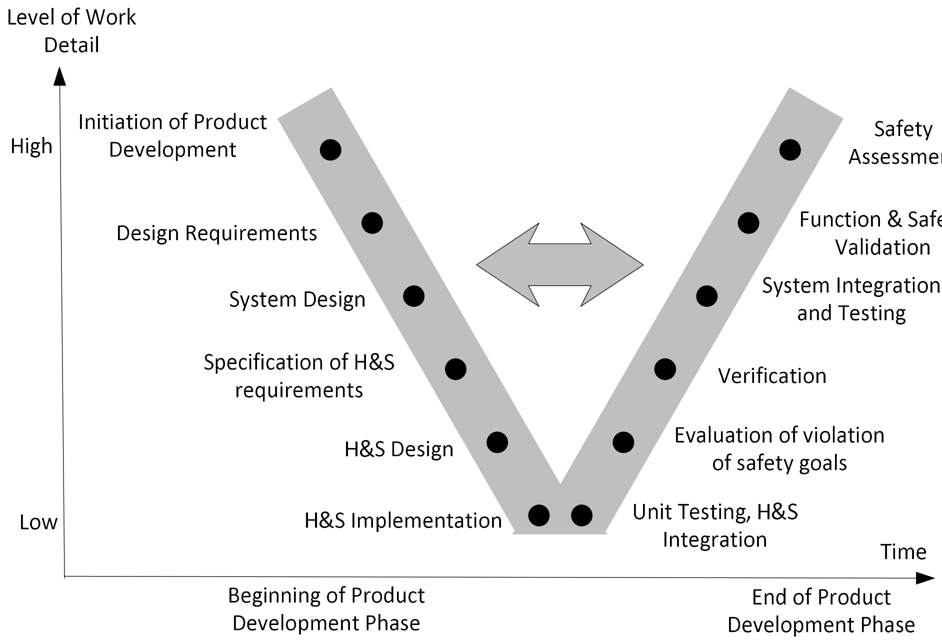
Level of Work Detail
High
Low
Initiation of Product Development
Design Requirements
System Design
Specification of H&S requirements
H&S Design
H&S Implementation
Unit Testing, H&S Integration
Evaluation of violation of safety goals
Verification
System Integration and Testing
Function & Safety Validation
Safety Assessment
Time
Beginning of Product Development Phase
End of Product Development Phase

Fig. 1. System Engineering V-model listing main tasks on the left and right sides of the V.

Systems engineering also uses the so-called V model to specify and describe key tasks while designing a complex or large system. These key tasks include requirements, design, implementation, test, verification, and validation which are performed at specific times and in a *V* shape depicted in Fig. 1. The lifetime phases (or tasks) of requirements, design, and implementation are performed on the left side of the V while the tasks involving testing, verification, and validation are performed on the right side of the V. The reason for depicting these activities on a V (rather than in a linear fashion) is that while working on the requirements specifications, the corresponding specifications for validation are developed concurrently and correspond to the same vertical level. While the completion of the requirements specification task is made very early in the development lifecycle of the product or project, the completion of the validation task is the last one. On the left side of the V, decomposition and definition activities resolve the system architecture and create the details of the design. Integration and verification flow up and to the right as successively higher levels of subsystems are verified, culminating at the system level. Verification ensures the system was built right (meets requirements, standards, etc.), whereas validation ensures the right system was built (meets customers' needs). Ascending the right side of the V is the process of integration and verification and at each level, there is a direct correspondence between activities on the left and right sides of the V. This correspondence is deliberate - the method of verification must be determined as the requirements are developed and documented at each level. This minimizes the chances that requirements are specified in a way that cannot be measured or verified.

Although SE can be applied to any endeavor (e.g., physical and social sciences, engineering, and management, etc.), in this section we discuss its application to engineering and more specifically, designing a safe automated vehicle. One of the most fundamental ideas of SE is the following. At each level, starting at highest level of abstraction and proceeding to lowest, identify main functions using input/output relationships and use functional decomposition techniques to generate sub-functions at lower levels of abstraction. This decomposition process is repeated until 4 to 8 levels (typically) deep are generated. At each level sufficient details is added to clarify composition, relationships, requirements, behaviors, interfaces, parameters, etc. This process is illustrated in Figs 2 (a) and (b) where a system or sub-subsystem is decomposed into constituent components until a sub-system with readily available components can be implemented.

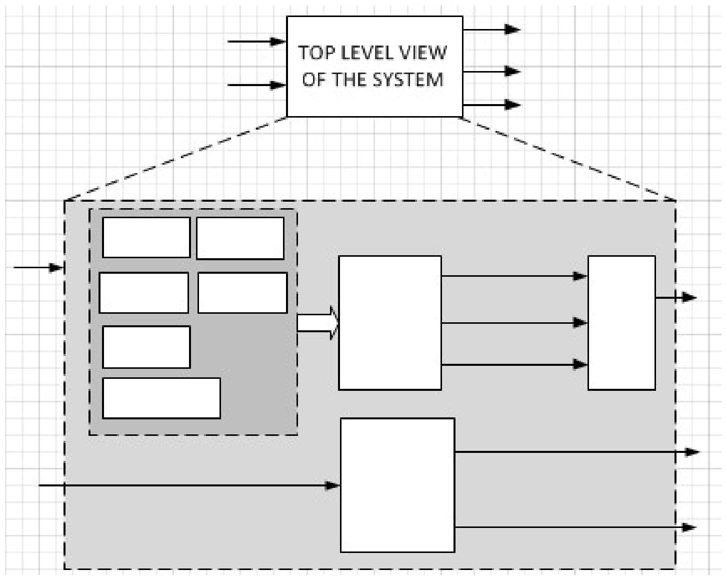

Figure 2 (a). Top level block diagram of a system.

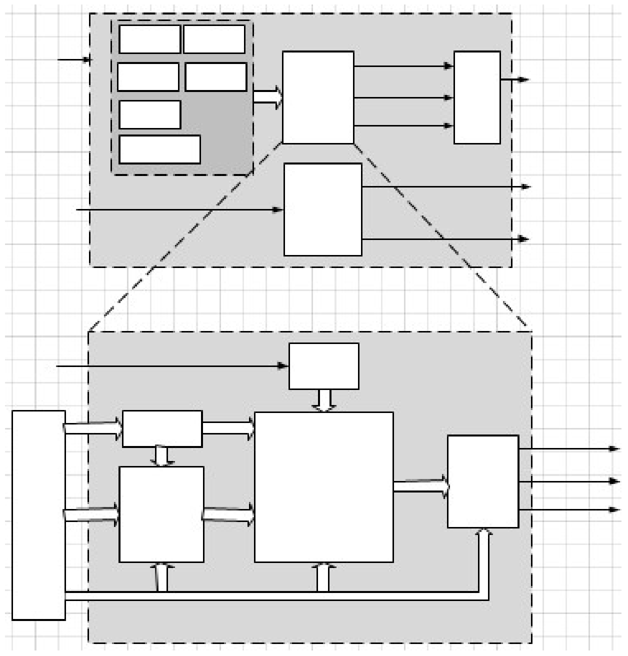

Figure 2 (b). Functional decomposition of a component into its constituent sub-components.

Designing a vehicle with a high level of safety is complex, thus the importance of using SE principles. Indeed, the ISO 26262 standard which has been specifically developed for automotive safety uses the V model as the underlying framework. This decision minimizes unnecessary rework, errors in requirements development and cascading. This will force some testing and verification activities to occur at various levels of integration early in the process. More specifically, the ISO 26262 standard uses the V model at the system level (Section 4), at the hardware level (Section 5) and at the software level (Section 6).

1.3.1 Model Based Systems Engineering (MBSE)

Model-based systems engineering is a systems engineering methodology that focuses on creating and exploiting domain models as the primary means of information exchange between engineers, rather than on document-based information exchange. These models fall into many categories including functional, behavioral, structural, operational, component, performance, safety, and many others. The benefits that are attributed to using an MBSE approach include shared understanding of system requirements and design, assisting in managing complex system development, improving design quality, supporting early and on-going verification and validation to reduce risk, providing value through life cycle, and enhancing knowledge capture. One effective way to benefit from MBSE is through specific modeling languages for systems engineering which have incorporated many domain models in an intrinsic fashion as is the case with SysML which is summarized next.

1.3.2 The Systems Modeling Language (SysML)

SysML is a general-purpose modeling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities. SysML is becoming important in the execution of many safety tasks associated with the V-model of system engineering. There are four major categories of SysML diagrams, also known as the "four pillars of SysML", structure, behavior, requirements, and parametrics. The structure category basically allows one to describe what the system is including its main constituent components. While this is important to get started, the structure category is static as it does not describe how the system works or behaves which is the function of the behavior category. Thus the behavior category describes the dynamic aspect of the system. As noted, all design work begin with a set of requirements describing what one can expect from the system. The requirements category of SysML enables the specification of these requirements. A system typically consists of many sub-systems with a number of interfaces. The parametrics category defines the nature of these interfaces including quantification beyond qualitativeness, e.g., formulas, equations, units, etc.

In most cases, just nine SysML diagrams are used which belong to the four categories described previously. The structure category includes three diagrams: block definition diagram (*bdd*), internal block diagram (*ibd*), and package diagram (*pkg*). Four diagrams belong to the behavior category: sequence diagram (sq), state Machine diagram (*stm*), activity diagram (*act*), and use case diagram (*uc*). The requirements and parametrics categories each include just one diagram, the requirement diagram (*req*), and parametric diagram (par) respectively. In the following, we briefly summarize some of these SysML diagrams.

The SysML block definition diagram (bdd) represents system elements called blocks, and their composition, classification and navigation. The following figure illustrates the *bdd* of a perception system of an automated vehicle.

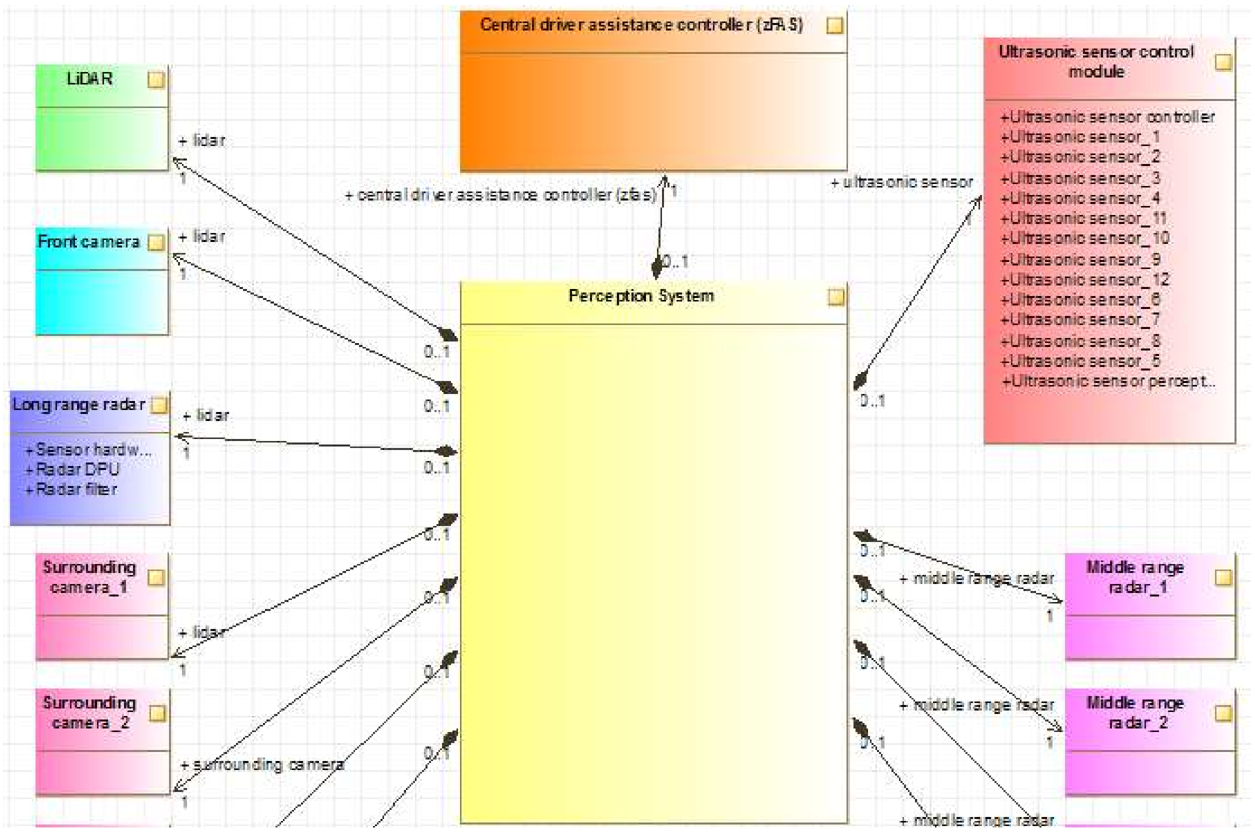

Figure 3. The block definition diagram (bdd) of SysML.

The SysML internal block diagram (ibd) represents interconnection and interfaces between the parts of a block including external ports of the block
. The following figure illustrates the ibd of a perception system of an automated vehicle.

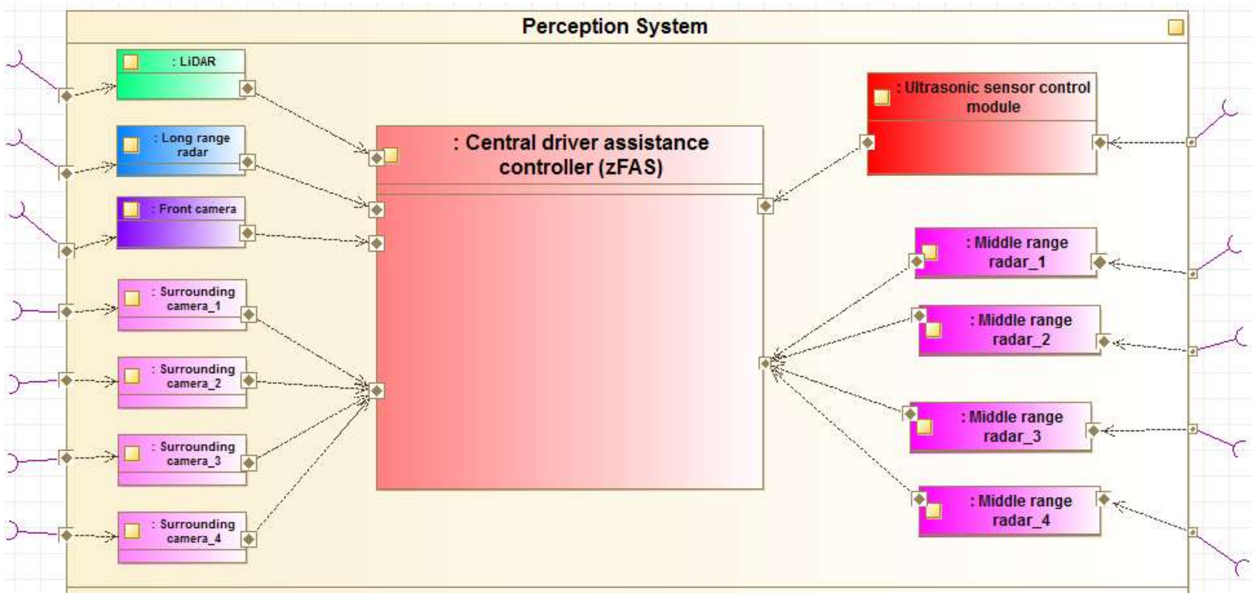

Figure 4. The internal block diagram (ibd) of SysML.

The SysML package diagram (pkg) represents the organization of a model in terms of packages that contain model elements. Thus this diagram is used for model structuring rather than system structuring. The SysML sequence diagram (sd) represents behavior in terms of a sequence of messages exchanged between parts. The following figure illustrates the sd of a radar sensor sub-system pertaining to the perception system of an automated vehicle.

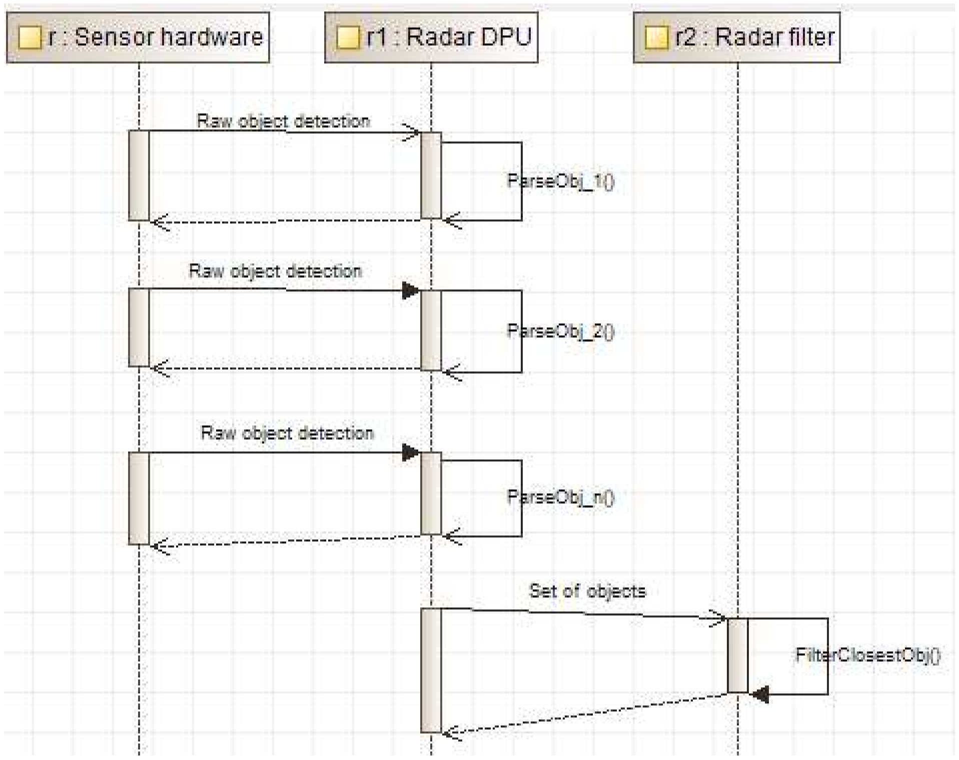

Figure 5. The sequence diagram (sd) of SysML.

The SysML state machine diagram (stm) represents behavior of an entity in terms of its transitions between states triggered by events. The following figure illustrates the stm of an antilock braking system (ABS) of an automobile.

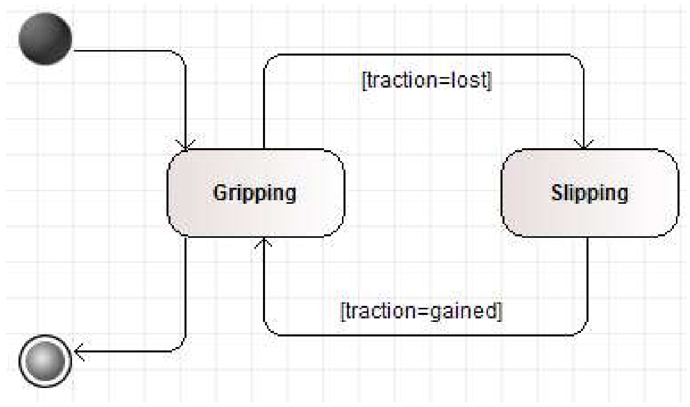

Figure 6. The state transition diagram (std) of SysML.

The SysML activity diagram (act) represents behavior in terms of the ordering of actions based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs. The following figure illustrates the act of an antilock braking system (ABS) of an automobile.

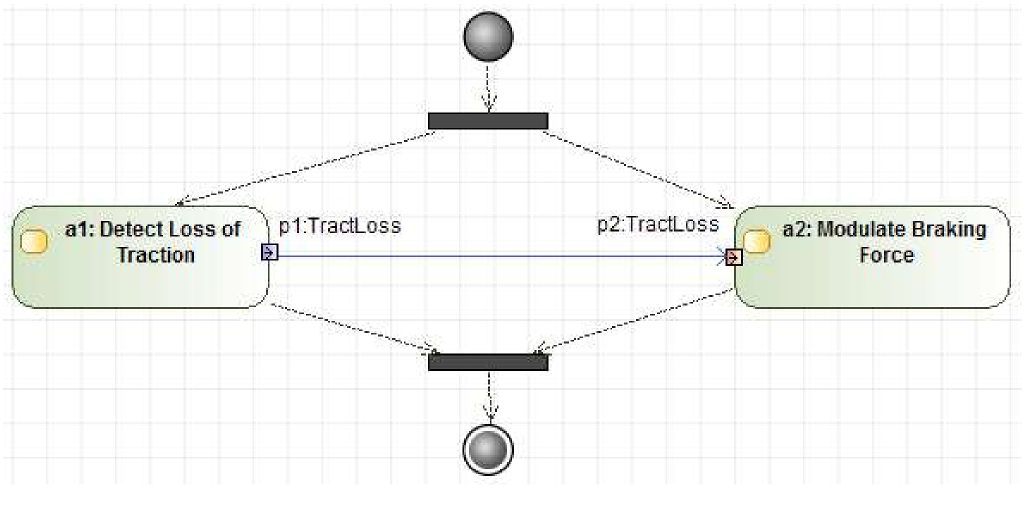

Figure 7. The activity diagram (act) of SysML.

The SysML use-case diagram (uc) represents functionality in terms of how a system or other entity is used by external entities (i.e., actors) to accomplish a set of goals. The SysML requirements diagram (req) represents text-based requirements and their relationships with other requirements, design elements, and test cases to support requirements traceability. The following figure illustrates the req of a radar sensor sub-system pertaining to the perception system of an automated vehicle.

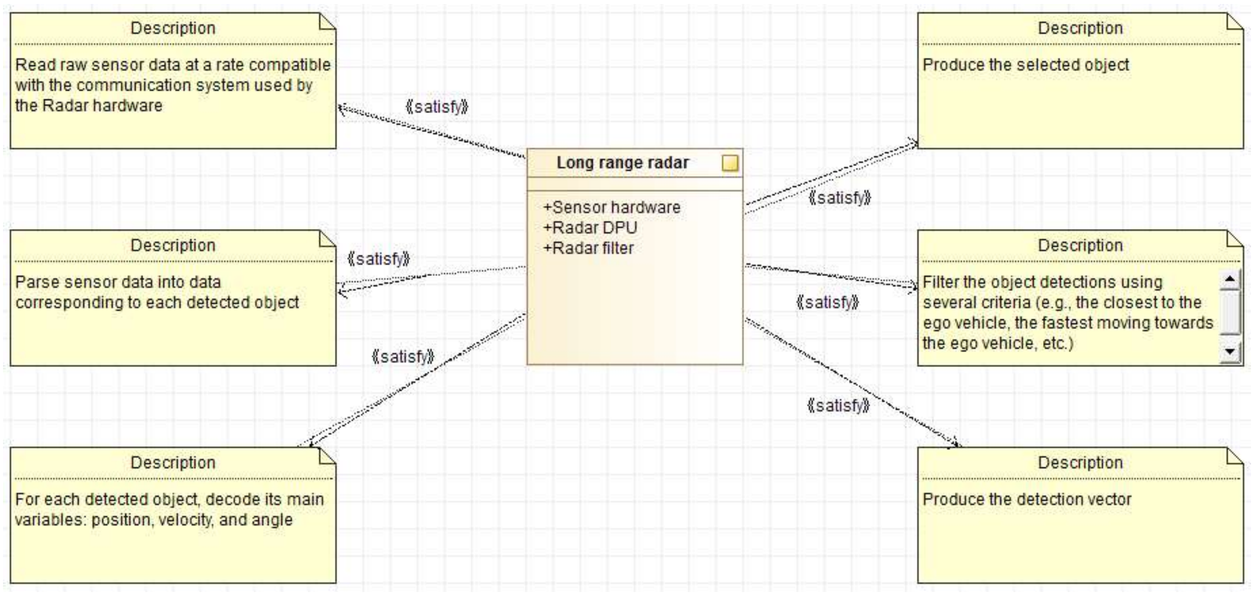
Description
Read raw sensor data at a rate compatible with the communication system used by the Radar hardware
Description
Parse sensor data into data corresponding to each detected object
Description
For each detected object, decode its main variables: position, velocity, and angle
Long range radar
+Sensor hardware
+Radar DPU
+Radar filter
«satisfy»
Description
Produce the selected object
Description
Filter the object detections using several criteria (e.g., the closest to the ego vehicle, the fastest moving towards the ego vehicle, etc.)
Description
Produce the detection vector

Figure 8. The requirements diagram (req) of SysML.

The SysML parametric diagram (*par*) represents constraints on property values, such as *F* = *m*a*, used to support engineering analysis. The following figure illustrates the *par* of an antilock braking system (ABS) of an automobile.

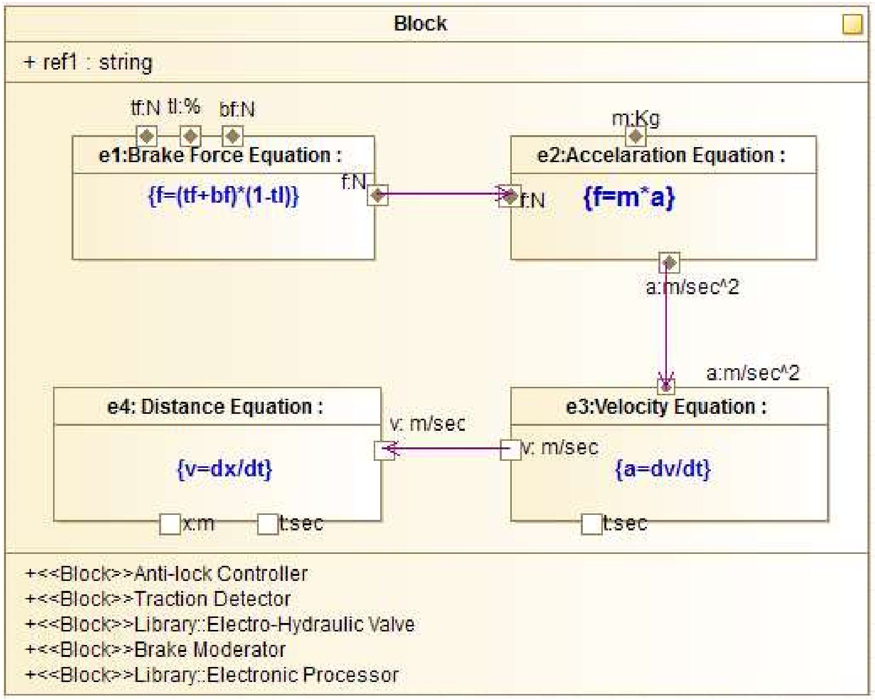
Block
+ ref1 : string
tf:N tl:% bf:N
e1:Brake Force Equation :
{f=(tf+bf)*(1-tl)}
f:N
m:Kg
e2:Accelaration Equation :
f:N
{f=m*a}
a:m/sec^2
a:m/sec^2
e3:Velocity Equation :
v: m/sec
{a=dv/dt}
t:sec
e4: Distance Equation :
v: m/sec
{v=dx/dt}
x:m
t:sec
+<<Block>>Anti-lock Controller
+<<Block>>Traction Detector
+<<Block>>Library::Electro-Hydraulic Valve
+<<Block>>Brake Moderator
+<<Block>>Library::Electronic Processor

Figure 9. The parametric diagram (par) of SysML.

REFERENCES

1. S. Thrun et al., “Stanley: The robot that won the DARPA grand challenge,” Journal of Field Robotics, vol. 23, no. 9, pp. 661–692, 2006.
2. C. Urmson et al., “Autonomous driving in urban environments: Boss and the urban challenge,” Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.
3. H. Cheng, Autonomous Intelligent Vehicles: Theory, Algorithms, and Implementation. Springer London, 2011.
4. F.-Y. Wang et al., “IVS O5: New developments and research trends for intelligent vehicles,” IEEE Intelligent Systems, vol. 20, no. 4, pp. 10–14, 2005.
5. H. Cheng et al., “Interactive road situation analysis for driver assistance and safety warning systems: Framework and algorithms,” IEEE Transactions on Intelligent Transportation Systems, vol. 8, no. 1, pp. 157–166, 2007.
6. International Organization for Standardization, “Road Vehicles – Functional Safety,” ISO Standard 26262, 2011.
7. P. Koopman and M. Wagner, “Autonomous vehicle safety: An interdisciplinary challenge,” IEEE Intelligent Transportation Systems Magazine, vol. 9, no. 1, pp. 90–96, 2017.
8. A. Avizienis et al., “Basic concepts and taxonomy of dependable and secure computing,” IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11–33, 2004.
9. N. G. Leveson, Safeware: System Safety and Computers. Addison- Wesley, 1995.
10. W. Wendorff, “Quantitative SOTIF analysis for highly automated driving systems,” in Safetronic.2017, Conference Proceedings, Stuttgart, Germany, 2017.
11. J. Thomas et al., “An integrated approach to requirements development and hazard analysis,” in SAE Technical Paper No. 2015-01-0274.
12. N. G. Leveson, Engineering a Safer World : Systems Thinking Applied to Safety. MIT Press, 2012.
13. W. Young and N. G. Leveson, “An integrated approach to safety and security based on systems theory,” Communications of the Association for Computing Machinery (ACM), vol. 57, no. 2, pp. 31–35, 2014.
14. A. Abdulkhaleq, S. Wagner, and N. Leveson, “A comprehensive safety engineering approach for software-intensive systems based on STPA,” in 3rd European STAMP Workshop, Conference Proceedings, Amsterdam, The Netherlands, 2015.
15. A. Abdulkhaleq et al., “A systematic approach based on STPA for developing a dependable architecture for fully automated driving vehicles,” in 4th European STAMP Workshop, Conference Proceedings, Zurich, Switzerland, 2017.
16. H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers, 1997.
17. P. Koopman and M. Wagner, “Toward a framework for highly automated vehicle safety validation,” in SAE Technical Paper No. 2018-01-1071.
18. International Electrotechnical Commission, “Functional Safety of Electrical / Electronic / Programmable Electronic Safety-Related Systems,” IEC Standard 61508, 2010.
19. International Electrotechnical Commission, “Functional Safety – Safety Instrumented Systems for the Process Industry Sector,” IEC Standard 61511, 2018.
20. S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” Computing Research Repository (CoRR), vol. arXiv:1708.06374 [cs.RO], 2017. [Online]. Available:http://arxiv.org/abs/1708.06374
21. L. Gauerhof, P. Munk, and S. Burton, “Structuring Validation Targets of a Machine Learning Function Applied to Autonomous Driving,” B. Gallina et al. (Eds.): SAFECOMP 2018, LNCS 11093, pp. 45–58, 2018.

22. Patrik Feth, Rasmus Adler, Takeshi Fukuda, Tasuku Ishigooka, Satoshi Otsuka, Daniel Schneider, Denis Uecker, and Kentaro Yoshimura, "Multi-aspect Safety Engineering for Highly Automated Driving Looking Beyond Functional Safety and Established Standards and Methodologies," B. Gallina et al. (Eds.): SAFECOMP 2018, LNCS 11093, pp. 59–72, 2018.
23. J. Pimentel and J. Bastiaan, Characterizing the Safety of Self-Driving Vehicles: A Fault Containment Protocol for Functionality Involving Vehicle Detection, 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES), September 12-14, 2018, Madrid, Spain.
24. J. Pimentel, J. Bastiaan, and M.Zadeh, Numerical Evaluation of the Safety of Self-Driving Vehicles: Functionality Involving Vehicle Detection, 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES), September 12-14, 2018, Madrid, Spain.
25. C. Schorn, A. Guntoro, and G. Ascheid, "Efficient On-Line Error Detection and Mitigation for Deep Neural Network Accelerators," B. Gallina et al. (Eds.): SAFECOMP 2018, LNCS 11093, pp. 205–219, 2018.
26. International Organization for Standardization, ISO/WD PAS 21448, "Road Vehicles - Safety of the Intended Functionality," ISO Working Draft, 2013.
27. S. Burton, L. Gauerhof, and C. Heinzemann, "Making the case for safety of machine learning in highly automated driving," in International Conference on Computer Safety, Reliability, and Security (SAFECOMP), Conference Proceedings, Trento, Italy, 2017.
28. R. Salay, R. Queiroz, and K. Czarnecki, "An analysis of ISO 26262: Machine learning and safety in automotive software," in SAE Technical Paper No. 2018-01-1075.
29. A. Geiger et al., "Vision meets robotics: The KITTI dataset," International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013.
30. X. Bi et al., "A new method of target detection based on autonomous radar and camera data fusion," in SAE Technical Paper No. 2017-01-1977.
31. M. Realpe, B. Vintimilla, and L. Vlacic, "Towards fault tolerant perception for autonomous vehicles: Local fusion," in 7th IEEE International Conference on Robotics, Automation and Mechatronics (RAM), Conference Proceedings, Angkor Wat, Cambodia, 2015.