Abstract Entity Patterns for Sensors and Actuators

Bijayita Thapa , Eduardo B. Fernandez * , Ionut Cardei , Maria Petrie

Posted Date: 20 March 2023

doi: 10.20944/preprints202303.0331.v1

Keywords: Cyber-physical system; Internet of things; Security pattern; Security solution frame

Article

Abstract Entity Patterns for Sensors and Actuators

Bijayita Thapa, Eduardo B. Fernandez*, Ionut Cardei, and Maria M. Petrie.

Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA; msyed2014@fau.edu

* Correspondence: fernande@fau.edu

Abstract: Sensors and actuators are fundamental units in Cyber-Physical and Internet of Things systems. Because they are included in a variety of systems, using many technologies, it is very useful to characterize their functions abstractly by describing them as Abstract Entity Patterns (AEPs), that are patterns that describe abstract conceptual entities. For concreteness, we study them here in the context of autonomous cars. An autonomous car is a complex system because, in addition to its own complex design, it interacts with other vehicles and with the surrounding infrastructure. To handle these functions, it must incorporate various technologies from different sources. An autonomous car is an example of a Cyber-Physical System, where some of its functions are performed by Internet of Things units. Sensors are extensively used in autonomous cars to measure physical quantities; actuators are commanded by controllers to perform appropriate physical actions. From AEPs we can derive concrete patterns, a structure combining related AEPs is an Entity Solution Frame (ESF). Both sensors and actuators are susceptible to malicious attacks due to the large attack surface of the system where they are used. Our work is intended to make autonomous cars more secure, which also increases their safety. Our final objective is to build a Security Solution Frame for sensors and actuators of autonomous cars that will facilitate their secure design. A Security Solution Frame is a solution structure that groups together and organizes related security patterns. This article is the first stage of a secure unit that can be used to design not only secure autonomous cars but also any system where sensors and actuators are used. This paper concentrates on AEPs and ESFs for sensors and actuators; that is, on the functional aspects of these devices.

Keywords: Cyber-physical system; Internet of things; Security pattern; Security solution frame

1. Introduction

Sensors and actuators are fundamental units in cyber-physical systems (CPS) and Internet of Things (IoT) systems. Because they can be included in a wide variety of systems, using many technologies, it is very useful to characterize their functions abstractly. CPS and IoT systems are usually complex and very heterogeneous, which makes their design very difficult; to handle this difficulty we use abstraction in the form of software patterns. A pattern is a solution to a recurrent problem in a given context, can be used as a guideline for design, and captures the experience and knowledge of designers to develop models that can be used for new designs [2]. In particular, we use a special type of patterns, *Abstract Entity Patterns* (AEPs), which are a type of analysis pattern that represent abstract conceptual entities that describe the fundamental attributes and behavior of an entity [53], in this case, conceptual models of sensors and actuators. AEPs are generalizations of Abstract Security Patterns (ASPs), a concept introduced in [8] and expanded in [12]. An abstract security pattern describes a conceptual security mechanism that includes functions able to stop or mitigate a threat or comply with a regulation or institutional policy; ASPs are used as paradigms to derive concrete patterns, which inherit their basic features. ASPs are secure conceptual entities, while AEPs are conceptual entities. A *Security Solution Frame* (SSF) [16], groups together related security patterns; it is built starting from ASPs and adding concrete security patterns suited to specific technologies. A Sensor AEP and an Actuator AEP describe the structure and operations that every sensor and actuator must have. A structure combining related AEPs (see Fig. 1 for two examples) is an *Entity Solution Frame* (ESF); that is, an SSF is a secure ESF.

For concreteness, we study sensors and actuators in the context of autonomous cars. An autonomous car is a complex system because, in addition to its own complex design, it interacts with other vehicles and with the surrounding infrastructure. To handle these functions, the car must incorporate various technologies from different sources. An autonomous car is an example of a cyber-physical system, which includes functions performed by Internet of Things devices. For example, IoT units allow an autonomous car to be connected wirelessly to roadside units, other vehicles, and fog and cloud systems; IoT units perform functions such as navigation and braking.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

Autonomous cars would be impossible without various types of sensors because they enable them to sense internal and external environments needed for safety and performance. Autonomous car Reference Architectures (RAs) are typically structured into hierarchical layers [1], and sensors and actuators are located in a low layer of the RA. The sensors used in autonomous cars monitor their operation, measure physical variables, make them capable of driving themselves, and enhance them to become fully autonomous. Some of the sensors, such as GPS receivers, radars, cameras, lidars, ultrasound, and laser scanners allow them to have functionalities such as lane-keeping, detect objects, automatic braking, automatic parking, etc. For example, if the car is drifting from its lane, lane-keep assist will automatically steer back the vehicle into its lane.

Security is a fundamental objective for autonomous cars and for most CPSs because their safe operation can be affected by security attacks. Because they usually perform basic functions in autonomous cars, the security of sensors and actuators is very important. Our objective for future work is to build a Security Solution Frame for sensors and actuators for autonomous cars that will facilitate their secure design; instead of securing units piecewise, designers can use a unit with several functions that has been proven to be secure. This SSF requires several security patterns and can be used later for building a security reference architecture for autonomous cars. To get to this SSF we need to build an ASF first and then add security defenses.

In this article, we present an AEP for sensors, from which we derive concrete patterns for radar and lidar sensors. Also, we build an AEP for actuators, from which we derive a pattern for a braking system. In future work, we will add security mechanisms to these patterns to derive security patterns for lidar and vision sensors, which will allow us to build an SSF for sensors. We have already published a Sensor ASP, from which we derived the Secure Radar Pattern [3].

Figure 1 presents a pattern diagram of the combination of two ESFs (a pattern diagram shows relationships between patterns [7]): the ESF based on the Sensor AEP shows Lidar, Radar, Vision, IMU (Inertial Measurement Unit), and GPS as derived patterns; the ESF based on the Actuator ESP shows Brake, Accelerator, and Steering as derived patterns. This model includes the typical components used in autonomous cars, specific types of cars may use more or fewer of these devices. IoT-Thing is an entity in an IoT system that controls actuators and activates the reception of data from sensors.

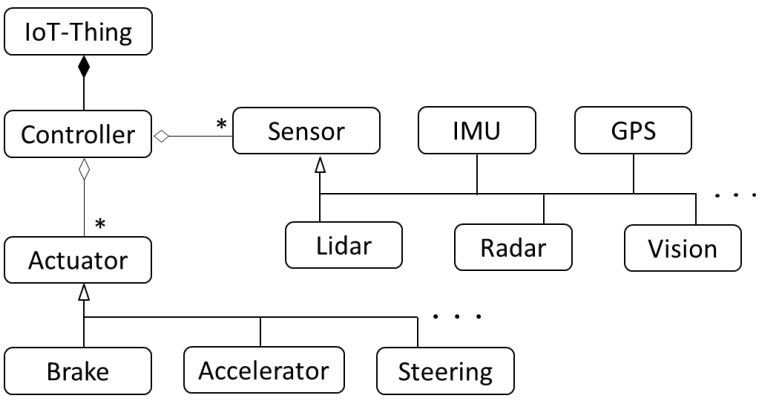

Figure 1. Pattern diagram of a combination of ESFs for sensors and actuators.

Actuators receive commands from the controller and take actions on subsystems, such as the brake, engine, steering, and transmission. Actuators play an important role in Adaptive Cruise Control (ACC) systems and other functions of various Advanced Driver Assistance Systems (ADAS); for example, the Automatic Emergency Braking (AEB), assists lane-keeping, and some other functions. If obstacles are detected using sensor data an Electronic Control Unit (ECU) avoids collisions by sending signals to actuators to reduce engine power, adjust steering, or apply brakes [4, 5]. Similar structures can be found in crane control systems, water purification systems, and many other CPSs.

Our contributions in this article include:

- AEPs for sensors and actuators. These patterns are paradigms for any concrete type of sensor or actuator, from which concrete patterns can be derived.
- Two concrete patterns derived from these AEPs: the Lidar Sensor and the Brake Actuator.
- The structure and behavior of the ESFs for sensors and actuators. These ESFs are the basis for SSFs, secure units that can be applied in the design of a variety of CPSs.

- A validation of the functional correctness of these AEPs and of their derived patterns.
- A roadmap to derive concrete security patterns for sensors and actuators and build the corresponding SSFs.

The rest of the paper is organized as: Section 2 includes background material to define some concepts necessary to understand this paper. In Section 3, we define Abstract Entity Patterns for sensors and actuators, while Section 4 shows concrete patterns for radar, lidar, and brake. In Section 5 we validate the new patterns and their corresponding ESFs. Section 6 discusses how to build the SSF for autonomous driving, while Section 7 discusses related work. Finally, Section 8 presents conclusions and future work.

2. Background

2.1. Cyber-Physical Systems (CPSs) and Internet of Things (IoT)

A Cyber-Physical System (CPS) is a system that integrates computational and networking systems with a physical system. Physical systems are connected through networking technologies, and they are integrated with physical units using sensing, actuation, control, and computation units. Some examples of CPSs are autonomous cars, robots, unmanned aerial vehicles (UAVs), manufacturing systems, industrial control systems, and many similar systems.

The IoT defines systems in which things (objects) are interconnected through the internet and can interact and collaborate to reach common goals. In IoT systems, a *thing* is a physical or virtual object with a unique identifier. IoT systems may include actuators, sensors, controllers, and smart devices. IoT applications are used not only in autonomous cars but also in many other smart systems such as buildings, homes, health care, defense, transportation, education, and agriculture. Often, CPSs and IoT systems are combined as the case of autonomous cars (CPSs), which use combinations of specific types of sensors and actuators (IoT systems).

2.2. Autonomous Cars

Autonomous cars are also known as driverless cars or self-driving cars; they are capable of sensing their surroundings and driving themselves. To drive by itself (autonomous driving), an autonomous car relies on advanced sensors, actuators, artificial intelligence, and other components and technologies. Radar, lidar, vision, and ultrasonic sensors are used; each of them has its special advantages. Radar sensors detect the position of nearby objects by using radio waves. Lidar sensors measure distances, identify lane markings, detect road edges, detect obstacles, etc., using laser light. Camera sensors detect traffic lights, road signs, pedestrians, other vehicles, and other objects. Similarly, ultrasonic sensors are useful for detecting curbs and other objects while parking. All the data collected by these sensors are sent to a controller which plans a path and then sends instructions to the actuators to control acceleration, braking, and steering [6].

2.3. Patterns

A pattern is a solution to a recurrent problem in a given context [2,7]. A pattern embodies the knowledge and experience of designers, can be reused in new applications, is useful for communication between designers, and is also useful to evaluate and re-engineer existing systems [8]. Both patterns and collections of related patterns provide systems and software engineers a new design paradigm or architectural style and help them avoid pitfalls in their system designs [9]. A pattern is described with a template composed of a set of structured sections that include Intent (problem solved by the pattern), Context (environment where the pattern is applicable), Problem (general problem solved by the pattern and the forces that constrain the solution), Solution (the solution in words and using UML static and dynamic diagrams), Implementation (hints for using the pattern solution), Known Uses (three or more uses of the solution in practice), Consequences (advantages and liabilities in using the pattern) based on the forces of the problem, Related Patterns (other patterns that complement the solution or provide alternative solutions). Patterns can be analysis patterns, architectural patterns, design patterns, privacy patterns, security patterns, and other types. A pattern is a type that is instantiated (maybe several times) in the model of a system design to perform specific functions; e.g., to control access in several points in a system. Pattern diagrams indicate how patterns relate to each other, showing the contribution a pattern brings to another [10].

An Abstract Entity Pattern (AEP) is a type of analysis pattern that describes the core structures and functions of physical or conceptual entities [53]. An Abstract Security Pattern (ASP) is a type of security pattern that describes a conceptual security mechanism that realizes one or more security policies to handle (stop or mitigate) a threat or comply with a security-related regulation or institutional policy without implementation aspects [11,12]. Starting from ASPs,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

security patterns can be defined at all the layers of a system architecture to match any type of context [13]. Some of the ASPs correspond to basic security mechanisms; for example, access control, and authentication, whereas some of them specify more detailed aspects, such as Role-Based Access Control (RBAC). Others define secure versions of abstract patterns, e.g., secure sensor. Abstract patterns are used as abstract prototypes for derived concrete patterns, which makes it easy to see what security constraints must be applied in a specific context. For example, from a Sensor ASP, we can derive a secure radar sensor pattern. Because domain models and Security Reference Architectures are abstract architectures, we can use AEPs and ASPs in their construction.

A Reference Architecture (RA) is a generic software architecture that describes the functionality of a system at a high level of abstraction without containing implementation details, and it is a useful tool to understand and build a complex system by describing its main components and their relationships [14,15]. A Security Reference Architecture (SRA) describes a conceptual model of security and provides a way to specify security requirements for a wide range of concrete architectures [15]. An RA can be built as a combination of several patterns and can be based on one or more domains. An RA should define the fundamental concepts of a system, expressed as AEPs, and the interactions among these units [8]. We can add security patterns to an RA to neutralize identified threats, and thus obtain an SRA.

Patterns and RAs are validated through examples and analysis of their completeness and security or other attributes; it does not make sense to validate experimentally an instance of a pattern because evaluating its properties only verifies them for that instance, not for all the instances of the pattern or RA. The use of patterns for system design is grounded on the principles of Design Science [54], and several pattern conferences each year introduce new patterns as well as applications of this design style.

2.4. Security Solution Frames (SSFs)

An SSF is a structure that groups and organizes related security patterns, and it can be built starting from ASPs as roots [16]. For example, a Secure Channel SSF collects patterns that are used for building secure channel structures using Symmetric Cryptography, Asymmetric Cryptography, and Digital Signatures [11]. SSFs can facilitate the work of designers by collecting several related patterns together to realize security requirements and guiding designs from an abstract conceptual level to a concrete implementation-oriented level. SSFs define vertical and horizontal pattern structures, in which different levels of abstraction define a vertical structuring whereas different concerns define a horizontal association [11]. ASPs can be related to SSFs using pattern diagrams as shown in Figure 2; ASPs define the roots of these hierarchies, where each lower level is a pattern specialized for some specific context [11]. In other words, the forces of the ASP can be used in more specific forms in a concrete pattern, which adds new forces. For example, an SSF for autonomous driving or similar system includes a family of sensor and actuator patterns defined by the ASPs for secure sensors and actuators. The combination of ESFs and ASPs is a powerful tool for designers because they define conceptually complete as well as secure units that can be used in a variety of systems.

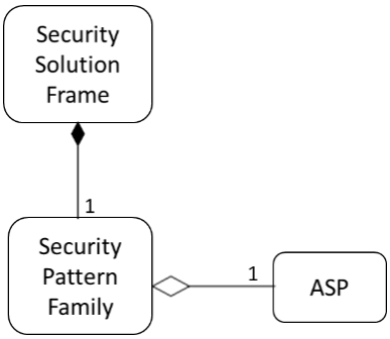

Figure 2. An SSF [11].

3. Abstract Entity Patterns for Sensors and Actuators

We show here AEPs for sensors and actuators. As the pattern for sensors has been published, we only present a summary here.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

3.1. Abstract Entity Sensor [3]

3.1.1. Intent

A sensor measures physical values from its environment and converts them into data that can be interpreted by humans or machines. This pattern describes the architecture of sensors at an abstract level.

3.1.2. Context

We consider an environment where we use cyber-physical and IoT systems; for example, autonomous cars. In these systems, we need to measure and collect physical quantities to be used by a control system or a logging system.

3.1.3. Problem

In IoT and CPSs there is a need to measure a variety of physical quantities, e.g., temperature, distance, pressure, etc. These are analogous quantities and must be converted to digital data and sent to other places or stored. What kind of devices can perform these functions?

The possible solution is constrained by the following forces:

- The measuring devices must collect physical values from their environment.
- The collected values should be able to be converted to digital data.
- The data collected must be sent accurately to a designated destination or stored.
- The device must have a sufficient amount of resources, such as computational capacity, power supply (battery life), etc.
- There is a need to collect data from different types and numbers of devices. Complex systems need a variety of measurements.
- Devices used for data collection should not be very expensive, which would increase the cost of the systems that use them.

3.1.4. Solution

A sensor is a device that can collect values from its surrounding environment and convert them into data. The data collected by the sensor is then used for various purposes in different systems.

Structure

In the basic sensor, we only need the sensor itself, a power supply, an analog-to-digital converter (ADC), and interfaces to the rest of the system. Figure 3 presents a class diagram of the sensor ACP. The Sensor (sensing unit) collects a signal representing some physical measure and sends it to an ADC, from where it can be captured through an Interface that communicates with other system components. This interface can also be used to receive commands to activate/deactivate the sensing unit.

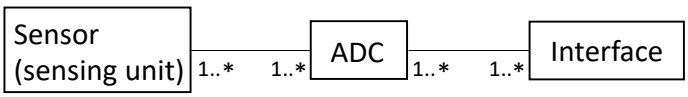

Figure 3. Class diagram of the Sensor ACP.

3.1.5. Known Uses

Since this is an abstract pattern, it has no Known Uses. However, its derived patterns have many uses.

3.1.6. Consequences

This pattern has the following advantages:

- Sensors can collect physical values from their environment.
- Sensors have their own ADCs to convert information into digital data.
- Sensors have interfaces, and data collected by sensors can be sent accurately to a designated destination.
- Sensors have power supplies and may use them only when required. Sensor technologies have increased due to the inexpensive availability of computational resources.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

- Various systems/applications use different types and numbers of sensors and collect data; however, sensor fusion technology can combine sensory data from disparate sources.

This pattern has the following disadvantages:

- Elaborated sensors are more expensive; their use must be justified by need.
- Since sensors may collect any activities around them, their use can raise concerns over the privacy of individuals. This point depends on the type of sensor.

3.2. Abstract Entity Actuator

3.2.1. Intent

An actuator converts a command from a controller into an action in the physical environment. This pattern describes the architecture of actuators at an abstract level.

3.2.2. Context

We consider environments, such as vehicles, manufacturing systems, and similar, where cyber-physical and IoT systems perform control functions. In these systems, we need to apply force for activities such as braking, moving a piece, or changing the position of a device.

3.2.3. Problem

Sensors used in IoTs and CPSs collect data and send it to a controller, which generates commands based on the data sent to it. The controller sends these commands to a system to perform actions like braking, steering, or moving objects. What kind of systems can perform these actions?

A possible solution is constrained by the following forces:

- *Command faithfulness*: Commands sent by controllers must be faithfully executed.
- *Resource availability*: There must be enough resources, e.g., batteries, to perform the actions.
- *Functional availability*: Systems must be available to act when needed.
- *Action variety*: There must be ways to perform different varieties of actions according to the needs of applications.
- *Power heterogeneity*: Different types of actions (electric, hydraulic, pneumatic) require different sources of power, such as electric or hydraulic power.

3.2.4. Solution

Physical actions can be performed by actuators by taking commands generated by a controller and performing the physical operations indicated by those commands. The digital commands of the controller must be converted into analog commands appropriate to the type of actuator.

Structure

In the basic actuator system, we only need the actuator, a digital-to-analog converter (DAC), and a controller. The main function of the actuator is to execute the actions (instructions) sent by the Controller for operating on a physical medium. Figure 4 presents the class diagram of the Actuator AEP. The controller receives information from IoT Things or other subsystems and generates instructions (in the form of digital signals). The resulting signals are sent to a DAC. The DAC converts a digital signal into an analog signal, which is sent to the Actuator, which performs the required command.

Figure 4. Class diagram of the Actuator ACP.

Dynamics

Figure 5 shows the sequence diagram of the use case “Execute a Command”. The scenario is:

- The controller receives data with instructions for physical actions.
- The controller generates digital commands.
- The controller sends digital commands to the DAC.
- The DAC converts these digital commands into analog commands

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

- The DAC sends the analog commands to the actuator.
- The actuator executes the requested physical action.

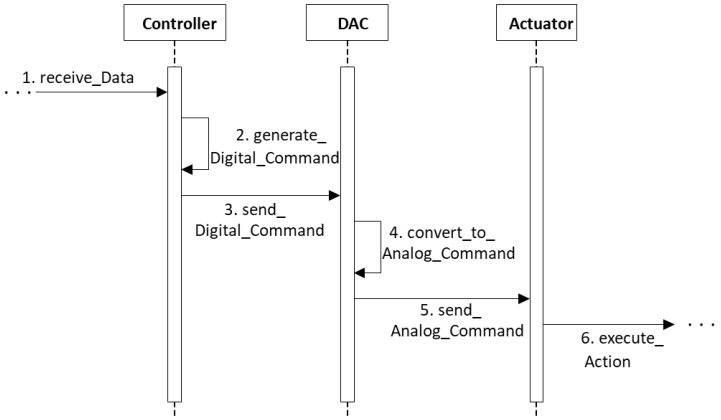

Figure 5. Sequence diagram of Use Case "Execute a Command".

3.2.5. Known Uses

Since this is an abstract pattern, it has no Known Uses. However, its derived patterns have many uses.

3.2.6. Consequences

This pattern has the following advantages:

- Actuators include mechanisms to perform physical actions following commands.
- Actuators can be provided with enough resources to perform their work, such as electrical power, fuel, or hydraulic/pneumatic energy.
- Different sources of power can be used in the actuators to perform different types of action. For example, electrically driven actuators may use power from the electricity grid [17]
- It is possible to build actuators appropriate for different types of applications.
- This pattern has the following disadvantages:
- Inexpensive concrete actuators may have limited resources and cannot perform elaborate commands, they may not meet design constraints on mass and volume. Advanced actuators may be costly.

4. Concrete Patterns for Sensors and Actuators

We show now concrete patterns derived from the AEPs of Section 3. We have already published a pattern derived from the Sensor AEP, a Radar pattern [3]. Here we add patterns for Lidar and Brake. As indicated earlier, they must respect the attributes and behavior of the corresponding AEPs.

4.1. Lidar Sensor

AKA: laser altimetry, laser radar, light detection and ranging.

4.1.1. Intent

This pattern describes the architecture of Light Detection and Ranging (lidar) sensors, which sense and measure the distance of an object with respect to a specific point.

4.1.2. Example

Consider an autonomous car, which drives itself from its start point to its destination; self-driving requires the perception and localization of its physical environment, planning, and motion execution and control. This car obtains information from its surrounding environment using various types of sensors; these sensors must calculate the distance to objects correctly, even in unfavorable weather conditions; otherwise, the car may collide with an object or a pedestrian.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

4.1.3. Context

Large numbers of heterogeneous sensors are used in systems like IoT and CPSs to perform various functions; autonomous cars are examples of this context. For security and safety reasons, autonomous cars must be able to sense their surroundings regardless of weather conditions and communicate with other vehicles and infrastructure. These cars need to navigate streets with pedestrians, double (two-way) traffic, large and small vehicles, stop signs, and traffic lights.

4.1.4. Problem

Various types of sensors play an important role in measuring the distance to objects (other cars, pedestrians, road signs, bicyclists, trees, etc.) present in the physical environment of autonomous cars. But not all sensors can detect objects accurately in unfavorable weather conditions, when objects are dark in color, during fog, or when the distance between the car and the object is large. For example, radar sensors have low resolution and processing speeds to produce an output and are not able to scan the physical environment at 360 degrees. Therefore, the use of only radar sensors is not good enough for autonomous cars. Wrong readings can lead to serious consequences. How can we get accurate measurements and detect objects correctly?

In addition to the forces of the corresponding AEP, the possible solution is constrained by the following forces:

- *Accuracy:* The measurement must be accurate enough and produced in real-time.
- *Cost:* The solution must have a reasonable cost.
- *Performance*: The performance of the sensor should not be limited by poor weather conditions, such as heavy rain, cloud, snow, fog, atmospheric haze, and strong and dazzling light, which can attenuate the signal and negatively affect the detection of objects.
- *Complementarity:* Different types of sensors should complement the measures of other sensors; otherwise, they would be a waste of money and functions.
- *Reliability:* Data measured and collected by sensors must be reliable.

4.1.5. Solution

Lidar sensors can sense and measure precisely the distance of objects with respect to some reference point. A lidar system uses pulsed lasers to accurately measure a 3-dimensional view of objects. The measurement is done by modulating the intensity, phase, and (or) the frequency of the waveform of the transmitted light and measuring the time required for that modulated waveform to come back to the receiver [18]. Lidar sensors use laser light waves that are a form of electromagnetic (EM) waves in the optical spectra and have shorter wavelengths compared to radio waves or microwaves; this shorter wavelength provides a better resolution.

Structure

Figure 6 shows the class diagram for a typical lidar system, which is approximately similar to a radar sensor system, the main difference being that lidar sensors use laser light waves while radar sensors use radio waves. The main components of a lidar sensor system include an Optical System (Optics), a Transmitter, a Receiver, and a Control and Processing Unit (CPU). The Optical System includes Transmitter and Receiver Optics to help focus the laser light by both reducing the divergence and unwanted noise in the return wave. Some lidar sensors have separate Transmitter Optics and Receiver Optics, but some lidar sensors have both of them combined into one unit, known as Transceiver Optics. The transmitter includes an Impulse generator to generate a laser beam of various wavelengths ranging from the infrared through the visible. This laser beam is used by lidar to make remote measurements of distance to Targets (Objects) and to generate monochrome images or 3D models.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

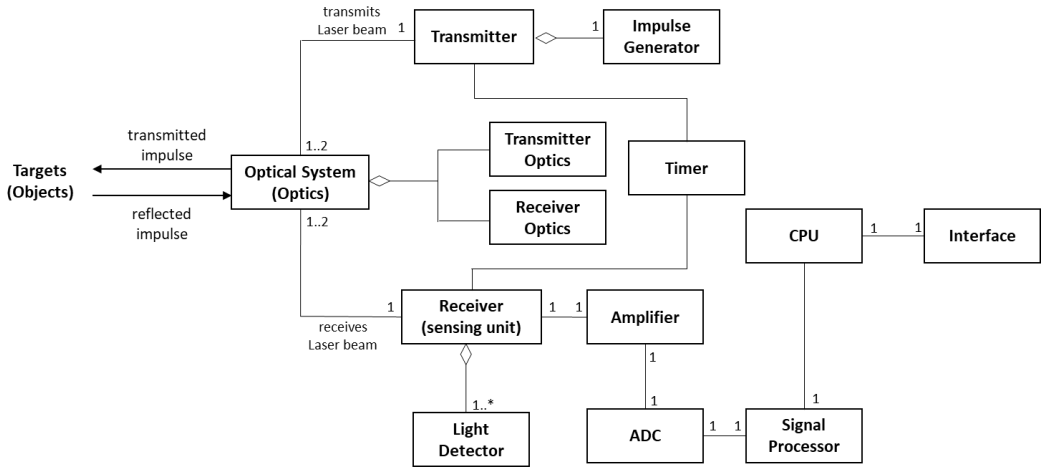

Figure 6. Class diagram of a Lidar Sensor.

In lidar sensors, the receiver could have a single or multiple Light Detectors. A Light Detector, also known as a Light Sensor, is a circuit or device that records the intensity of light and converts light energy into electric energy. An amplifier is a circuit used to increase the amplitude of signals from the receiver and send them to the ADC for digitization. The Analog-to-Digital Converter (ADC) is a device that transforms an analog signal into a digital signal. The Signal Processor cleans up signals for better resolution and separates targets from clutter. The CPU interprets digital signals (data) and creates output for the interface. The Interface is used to communicate with other system components.

Dynamics

Figure 7 presents a sequence diagram of the use case "Measure Distance to Targets (Objects)" for Time of Flight (ToF) scanning lidar. The ToF lidar measures the roundtrip travel time of photons from laser to targets (objects) and back, then converts it to distance; by using the distances measured, the lidar constructs the 3D point clouds [1]of the objects [19].

[1] The point clouds are datasets that represent objects or space; they are mostly generated using 3D laser scanners and lidar technology and techniques [51]. They are created at sub-pixel accuracy, at very dense intervals, and in real-time [52].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

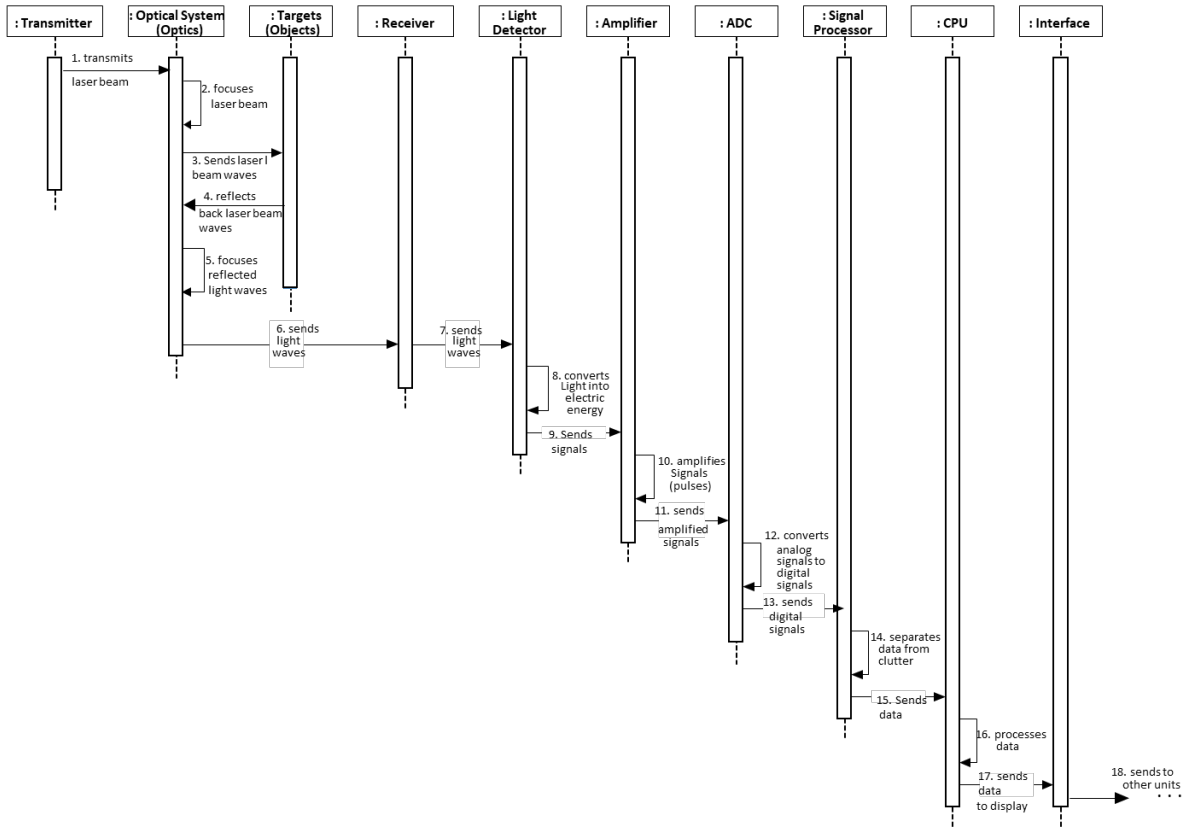

Figure 7. Sequence diagram of the Use Case "Measure Distance to Targets (Objects)".

The Transmitter sends the laser beam to the Transceiver Optics, which focuses the laser beam and transmits it to the targeted objects (such as pedestrians, cars, and bicycles). The reflected laser light from the object is focused by the Receiver Optics and sent to Light Detector. The Timer measures the time taken for roundtrip travel time to object and back. The Light Detector records the intensity of this light. The Amplifier amplifies the pulses and transmits them to the ADC where the pulses are converted into digital form. The Signal Processor separates data from noise. The CPU processes the data, and sends it to the Interface, from where it is sent to other components in the system.

4.1.6. Implementation

Users of lidar sensors can buy these sensors from lidar manufacturing companies, such as Luminar Technologies, Velodyne Lidar Inc., Teledyne Optech, Waymo, and many other vendors, based on what type of lidar sensors are required for their systems. There are several varieties of lidar sensors, which include mechanical scanning lidar, spinning lidar, flash lidar, optical phased array lidar, etc. Lidars can produce precise 3D images of all the objects in a street such as pedestrians, traffic lights, vehicles, bicycles, etc. Based on the number of optical apertures used, lidar sensors could be monostatic or bistatic. A monostatic lidar uses the same optical aperture as transmitter optics and receiver optics, whereas a bistatic lidar uses different apertures for transmitter optics and receiver optics. Also, lidar sensors can be classified as Time of Flight (ToF), Frequency-Modulated Continuous-Wave (FMCW), and Amplitude-Modulated Continuous Wave (AMCW). The ToF lidar sensors are the most common form of lidar sensor on vehicles [20]. The ToF lidar sensors spin around a 360-degree range and map their physical environment by measuring waves of laser light that are sent to objects and reflected on them, whereas in the FMCW lidar sensors, a continuous stream of light is sent instead of pulses of light. AMCW lidar sensors modulate the intensity of the light pulses keeping the frequency constant and they require high-speed radio-frequency electronics for light intensity modulation [21].

Based on their functionality, Lidars can also be divided into Airborne and Terrestrial lidars. Airborne lidars are used on helicopters or in drones, whereas Terrestrial lidars are used on vehicles or mobile devices. Airborne lidars are further divided into Bathymetric lidar and Topographic lidar, and Terrestrial lidars are divided into Mobile and Static lidars. In autonomous cars, usually, a mechanical spinning lidar is implemented on top of its roof. This lidar sends out

 doi:10.20944/preprints202303.0331.v1

pulses of light and measures the time taken for them to bounce back, then creates a map of 3D points to produce a snapshot of the objects around the car's surroundings.

4.1.7. Example Resolved

Lidar sensors are installed in autonomous cars by many car manufacturers. The use of lidar sensors is one of the best choices for mobile autonomous systems because they fill gaps of other sensors. They perform better for detecting the exact position, shape, and size of objects compared to other sensors. Because of the high resolution of lidar sensors, they can map a static environment as well as detect and identify moving objects, such as vehicles and pedestrians [22]. Lidar sensors play an important role by sensing and measuring the accurate distance of an object with respect to a specific point. The information collected by lidar sensors along with other sensors is used in autonomous cars for their perception and localization, planning, and motion execution and control.

4.1.8. Known Uses

Lidar sensors are used extensively in many applications across various industries and fields. Some of them are included here:

- Autonomous systems: Honda Motor's new self-driving Legend luxury sedan uses lidar sensors. Also, Waymo and many other autonomous car manufacturers are using lidar. Some car manufacturers, such as Germany's Daimler, Sweden's Volvo, the Israel-based Mobileye, and Toyota Motors have adopted lidar sensors (produced by Luminar Technologies) for their self-driving prototypes [23].
- Lidar sensors are used for Adaptive Cruise Control (ACC), Autonomous Emergency Brake (AEB), Anti-lock Braking System (ABS), etc. In addition, five lidar units produced by the German company Sick AG were used for short-range detection on Stanley, the autonomous car that won the 2005 DARPA Grand Challenge [24].
- General Motors uses lidar maps with its hands-free driving technology for its Cadillac's Super Cruise [20]. Also, Volkswagen is using Argo AI's lidar technology and planning to test self-driving vans, and Volvo has intentions to use lidar from the supplier Luminar [20].
- Forestry: lidar sensors are used to measure vegetation height, density, and other characteristics across large areas, such as forests, by capturing information about the landscape. Lidar directly measures vertical forest structure; this direct measurement of canopy heights, sub-canopy topography, and the vertical distribution of intercepted surfaces provides high-resolution maps and much data for forest characterization and management [25]. Aerial lidar was used to map the bush fires in Australia in early 2020 [24].
- Apple products: lidar sensors are used on iPhone 12 Pro, Pro Max, and iPad Pro, to improve portrait mode photos and to improve background shots in night mode.
- A Doppler lidar system was used in the 2008 Summer Olympics to measure wind fields during the yacht competition [24]
- A robotic Boeing AH-6 (light attack/reconnaissance helicopter) performed a fully autonomous flight in June 2010 using lidar to avoid obstacles [24].

4.1.9. Consequences

This pattern has the following advantages:

- *Accuracy*: Accuracy is high in lidar sensors because of their high spatial resolution, produced by the small focus diameter of their beam of light and shorter wavelength.
- *Cost*: The production of lidar sensors in high volume can bring their cost down. Luminar Technologies has developed low-priced lidar sensors priced at $500 to $1000 [23]. Some companies are trying to reduce the cost of lidar even more. Also, lidar manufacturing companies are using 905-nm diode lasers (inexpensive and can be used with silicon detectors) to bring lidar costs down [26].
- *Performance*: High-performance lidar sensors produced by Luminar Technologies can detect dark objects, such as debris or a person wearing black clothes [23]. Also, the performance of lidar sensors can be improved by combining them with other types of sensors like vision-based cameras, radar, etc.
- *Complementarity:* All the data collected by lidar, and other sensors are sent to Sensor Fusion Processor to combine them and build a more accurate model.

 doi:10.20944/preprints202303.0331.v1

- *Reliability:* According to [27], lidar-based methods for object detection are very robust and consistent. Also, solid-state lidars provide immunity to shock and vibration with greater than 100K hours of mean time between failures (MTBF) making them reliable to use in vehicles [28].

This pattern has the following disadvantages:

- Heterogeneity in lidar sensors makes complex their integration with other systems. Different types of lidar sensors are produced by different lidar manufacturing companies and there are no strict international protocols that guide the collection and analysis of the data using lidar [29].
- Lidar technology is not accepted by all auto manufacturing companies. For example, Tesla uses camera-based technology, instead of lidar sensors.
- Computational requirements for real-time use are high [24].

4.1.10. See Also (Related Patterns)

- A pattern for Sensor Network Architectures describes sensor network configurations [30].
- Sensor Node Design pattern [31], is a design pattern intended to model the architecture of a wireless sensor node with real-time constraints; it is designed and annotated using the UML/MARTE standard.
- Sensor Node pattern [32] describes the architecture and dynamics of a Sensor Node that includes a processor, memory, and a radio frequency transceiver.
- This pattern is derived from the Sensor AEP.

4.2. Autonomous Emergency Braking System (AEB)

AKA: Forward Automatic Emergency Braking, Autonomous Braking System, Automatic pre-collision Braking, Intelligent Emergency Braking

4.2.1. Intent

This pattern describes the braking system of autonomous cars, which alerts the passengers, reduces the speed of a vehicle, stops when it is necessary, and maintains the vehicle's direction (in the case of autonomous cars).

4.2.2. Context

We consider an environment for autonomous cars, where large numbers of heterogeneous sensors are used to collect data from their physical environment and perform various functions. As mentioned in section 4.1.3, autonomous cars must be able to sense their surroundings and communicate with other vehicles and infrastructure for their safety and security. These cars need to navigate streets with pedestrians and other vehicles and generate commands to execute actions such as speed up, slow down, or stop them if needed.

4.2.3. Problem

Large numbers of cars run on the road and accidents can occur because of many reasons. Also, there could be uncertain risky situations that autonomous cars may encounter while driving on the road or while parking. For example, while driving on the road, another vehicle or pedestrian may come on the way, or maybe some objects might have fallen on the middle of the road. Autonomous cars must be able to take decisions and react very fast in situations that may otherwise lead to a collision.

The possible solution to this problem is constrained by the following forces:

- *Accuracy*: Systems in autonomous cars must be able to execute commands from controllers accurately.
- *Reliability*: Autonomous cars must have reliable systems that perform correctly when needed.
- *Resources*: There must be enough resources, e.g., energy and computing power, to perform the actions.
- *Safety*: Systems must be able to stop cars in time with minimal delay.
- *Availability*: Systems must be available to act all the time in emergencies.
- *Autonomy*: Systems must be able to operate autonomously based on needs.
- *Performance*: There must be ways to perform different varieties of actions (slow down, speed up, or stop the systems if necessary) according to the needs of the systems.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

4.2.4. Solution

Safety is a priority for a passenger car. To avoid accidents or any risky situations for the safety of drivers, passengers, and pedestrians, cars must be able not just to detect obstacles in their physical environments that can potentially cause accidents. They must also be able to execute braking actions by themselves to slow down or stop based on need. Since an Autonomous Emergency Braking (AEB) System works autonomously, it can react faster than a human being and apply the brake quickly to slow down or stop the car in an emergency situation.

Structure

Figure 8 presents the main components of the Autonomous Emergency Braking (AEB) system. We have already described the sensor, controller, and actuator earlier in this paper. Whenever an obstacle is detected in the physical environment of the autonomous cars, sensors send data for detected objects to the controller. The controller sends instructions to the actuator to execute an action. The Brake Actuator applies an action to the brake to slow down or stop the car. Based on the speed and movement of objects in the surroundings, Brake Action is associated with Brake Actuator to execute the action, either to slow down the car or to completely stop.

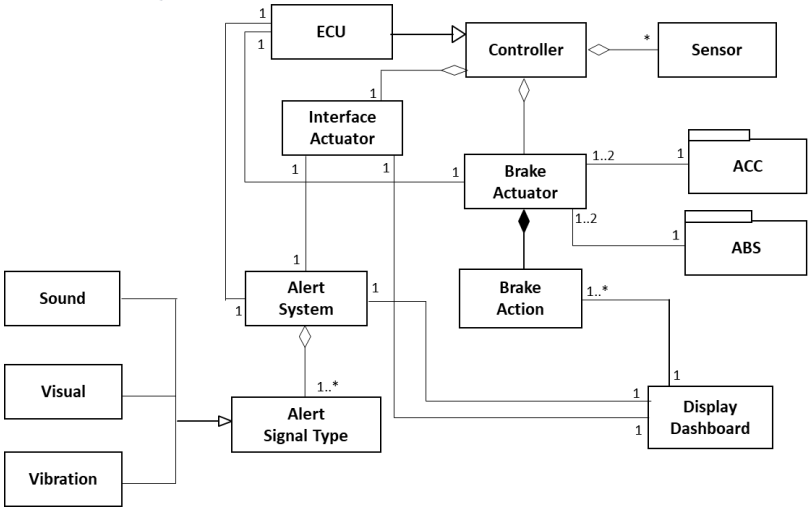

Figure 8. Class diagram of an Autonomous braking system.

The Interface Actuator activates the Display Dashboard and displays an alert signal if the brake is activated. An Alert System, also known as a forward-collision warning system, alerts the person in the car whenever an obstacle is detected, and the brake actuator acts to apply the brake. The Alert System has Alert Signal Types representing types of alerts, such as Sound, Visual, and Vibration. When sensors detect the car getting close to an obstacle, the system flashes a light, buzzes an alarm, or even shakes the steering wheel to alert the person in the car [33]. AEB is an advance over an Alert system because instead of providing just a warning to the person in the car, it engages the braking system without any human input [34].

The Electronic Control Unit (ECU) is an important part of the car, which controls all the electronic features in the car. Autonomous cars have more than a hundred ECUs for different functionalities. An ECU in an AEB system receives inputs from sensors when the car is approaching an obstacle and communicates them to the braking actuator. The ECU then sends signals to the brake to perform an action automatically based on the inputs to avoid or prevent a collision. The ECU can be then seen as a subclass of the Controller class.

Braking systems in cars produce friction of the brake pads against the surface of the wheels to slow down the car's speed. Anti-lock Braking System (ABS) is used in the AEB system to decrease the braking distance and retain steerability by preventing the wheels from locking up when a full brake is suddenly applied. In autonomous cars, ABS is applied automatically to reduce the braking forces in case of locked wheels. To make decisions and take the actions when to use ABS, the Controller in braking systems uses ABS sensors, Gyroscopes, etc., on the brake pads and wheels, and then the Brake Actuator executes the actions.

Another system that plays an important role with AEB is the Adaptive Cruise Control (ACC) system. The ACC system is used to reduce the risk of collisions. The ACC system allows cars to slow down and speed up autonomously based on the flow of traffic. For example, if it is detected that another vehicle is moving slowly in front of the car, the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

ACC system will slow down the car by using the autonomous braking system, and if the forward vehicle is no longer on its path or moving faster, the ACC system of the car will accelerate it back to its set cruise control speed [35].

Dynamics

Autonomous braking systems have several possible use cases, such as applying brakes, adjusting brakes…; Figure 9 presents a sequence diagram of the use case "Applying brakes in the presence of objects". Sensors in autonomous cars detect targets (objects) in their physical environment and send data to the IoT-Thing. These data are processed and sent to the Controller to make decisions on whether to slow down, swerve, or stop the car based on the distance to the object, road characteristics, and the speed of the car. If an object is moving slowly, then slow down the car, or else, swerve the car. Before executing an action, an Alert is displayed. The Interface Actuator executes an action in the Alert System, and an alert is displayed in the Display Dashboard in the form of sound, visual, or vibration. Based on the decision made, the Brake Actuator gets commands from the Controller, and it executes an action accordingly to apply the brake.

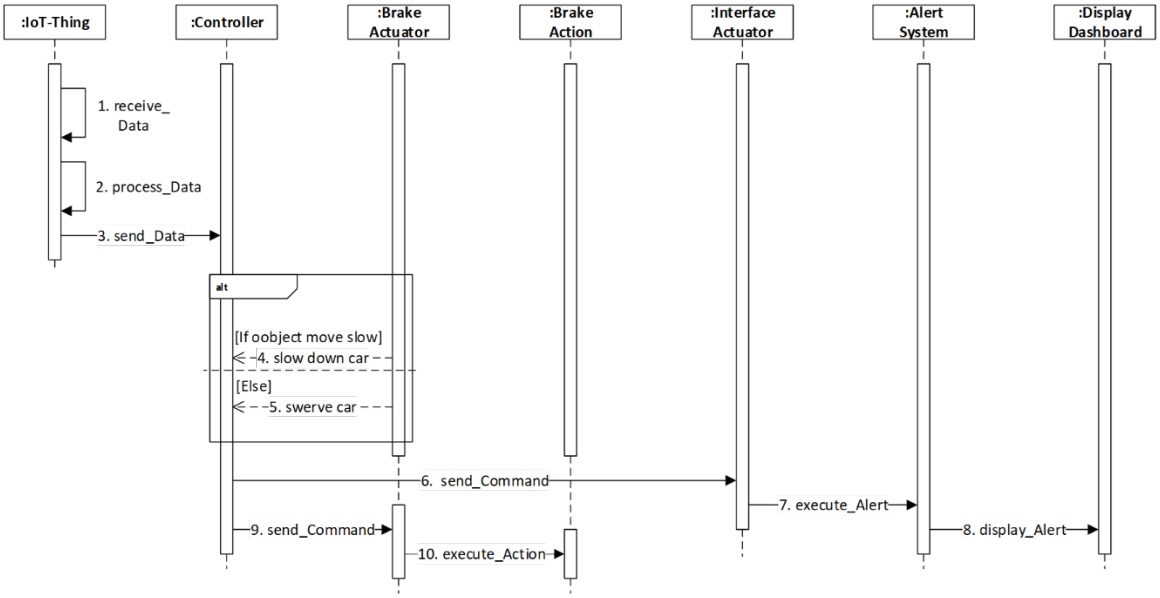

Figure 9. Sequence diagram of use case "Applying brakes in the presence of objects".

4.2.5. Implementation

The braking systems are already programmed and implemented in cars by manufacturing companies during their manufacturing phase and tested in the testing phase. Most of the recent cars have AEB systems implemented in them. Europe regulations require AEB to protect against forward collisions and collisions with big vehicles [33]. Also, AEB will soon be mandatory on every car and truck sold in the U.S. [36]. As per the agreement between automakers and the National Highway Traffic Safety Administration (NHTSA), by September 1, 2022, nearly all new vehicles sold in the United States will have AEB technology as standard equipment [5].

The AEB systems are designed differently based on the traffic environment (low speed: designed for city traffic speed, high speed: designed for highway or inter-urban speed, and pedestrian detection). Kia's AEB systems use a radar sensor installed at the front of the vehicle and sense objects on the road ahead all the time; if it detects any potential obstacle, the systems instruct the brakes to act to avoid the collision [37]. The types and number of sensors used in an AEB vary from one car manufacturing company to another. Most recent vehicles use radar and vision sensors for emergency braking, for example – when an ego vehicle[2] detects that other vehicles ahead of it have suddenly applied brakes [38]. The different types of data sets collected by various sensors are combined using sensor fusion technology.

4.2.6. Known Uses

- Tesla Model 3 is designed to determine the distance from a detected object traveling in front of the car and apply the brakes automatically to reduce the speed of the car when there is a possible collision. When the AEB applies

[2] An ego vehicle is a vehicle that contains sensors to sense the physical environment around it.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

brakes, the dashboard displays a visual warning and sounds a chime [39]. The system is also active when the autopilot feature is disengaged.

- Subaru models have an AEB system called EyeSight. This system is comprised of two cameras mounted above the windscreen to monitor the path ahead and it brings the car to a complete stop if it detects any issues [4]. Subaru's AEB system is integrated with ACC and lane departure warning.
- Mercedes-Benz introduced an early version of AEB in their S-Class 2005 model, which was one of the first cars with AEB. The recent S-class model of Mercedes-Benz has short-range and long-range radar, optical cameras, and ultrasonic detectors to detect the closest obstacles [40].
- The AEB system used in Volvo XC40 applies brakes automatically when objects like cyclists, pedestrians, city traffic, and faster-moving vehicles are detected [41].
- Other examples are Ford F-150, Honda CR-V, Toyota Camry, and many other car brands/models [42].

4.2.7. Consequences

This pattern has the following advantages:

- *Accuracy*: The units in the architecture described earlier are able to brake automatically in the presence of objects.
- *Reliability*: Autonomous braking systems can be built to be reliable by using high-quality materials and some redundancy.
- *Resources*: It is possible to provide enough resources, e.g., power, to perform the actions because cars generate energy that can be reused.
- *Safety*: Autonomous braking systems stop cars at a precise time (when the collision is expected) to avoid accidents. They are effective in reducing collisions.
- *Availability*: AEB systems give alerts and are available to act in emergencies all the time by the appropriate implementation.
- *Autonomy*: AEB systems operate autonomously and unattended.
- *Performance*: AEB systems can handle different situations, such as sensor errors, road conditions, obstacles, speed, velocity, position, direction, timing, etc., and slow down or stop cars completely if necessary.

This pattern has the following disadvantages:

- AEB systems are less effective in the dark, in the glare from sunrise and sunset, and in bad weather because sensors may not be able to detect objects efficiently. They also may not be very effective at very high speeds.
- Each car manufacturer has its specific approach to braking system designs and names them differently, therefore no two braking systems work in the same manner, except for fundamental characteristics. This may make maintenance complex.

4.2.8. See Also (Related Patterns)

- A Pattern for a Secure Actuator Node [43], describes how an actuator node performs securely on commands sent by a controller and communicates the effect of these actions to other nodes or controllers.
- A Real-Time Design Pattern for Actuators in Advanced Driver Assistance Systems [44] defines a design pattern for an action subsystem of an advanced driver assistance system to model the structural and behavioral aspects of the subsystem.
- Design Patterns for Advanced Driver Assistance Systems in [45] describes three patterns, namely i) Sensing, ii) Data Processing, and iii) Action-Taking, to cover design problems of sensing, processing, and control of sensor data, and taking actions for warning and actuation.
- This pattern is derived from the Actuator AEP.

5. Validation of AEPs and their derived patterns

As indicated in Section 2, software patterns are prototypes that can be instantiated in many possible ways; therefore, validating instances (specific implementations) by measuring their performance, security, or other quality factor, does not imply that all instances have similar attributes, and thus they do not prove anything about the pattern. For a designer, a pattern is a suggestion, not a plug-in unit. Patterns are abstractions of good practices, so we cannot validate them formally either; however, we can represent them in precise ways by using UML, a semiformal representation [55],

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2023 doi:10.20944/preprints202303.0331.v1

as we do here. Design artifacts in general, can be evaluated by applying Design Science methods [54], which suggest using qualitative measures to evaluate their clarity, completeness, and usability. Pattern language conferences analyze their qualitative properties and publish the patterns to be further analyzed by others. Their final validation comes from practitioners using them to build new systems or analyze existing systems. For those practitioners who understand their use they become a powerful tool to improve the speed and quality of their designs because the knowledge and experience of many other practitioners has been incorporated in their descriptions. An AEP is a canonical representation of the basic functions of a conceptual unit that can be used to derive correct concrete patterns that include all their fundamental properties [8]. As discussed in Section 6, AEPs and their derived patterns can be used as a basis for building secure systems. We can then perform this type of validation in all the patterns used in an ESF, which then validates the ESF itself; in particular, our AEPs for sensors and actuators, as well as its derived patterns, have been validated in this way, and we are confident that they can be used in the design of ESFs for cars and other systems that use these mechanisms; they can also be used as bases for the corresponding SSFs as we discuss in the next section.

6. An SSF for Autonomous Driving

We have shown that we can define Abstract Entity patterns as paradigms from which we can derive concrete patterns. From them we can build Entity Solution Frames (ESFs) that are packages of related patterns that can be used to design complex architectures. However, when considering safety-critical systems such as autonomous cars we also need to consider security; that is, we must build secure versions of these patterns, which leads to ASPs and their derived patterns; that is, to Security Solution Frames (SSFs). The counterpart of Fig.1, which combines two ESFs, is a pair of SSFs as shown in Fig. 10. This pair could be used by car engineers to design secure units of the car architecture. In general, which specific patterns are included in an SSF and which SSFs need to be combined depends on the application [10]. In an industrial setting these SSFs would be complemented with similar packages to describe other parts of the architecture; each pattern in an SSF indicates possible implementations (in one of its sections).

Patterns highlighted in blue in Figure 10 have been introduced in this paper. Our next task is to build the missing white patterns; we wrote a pattern for a Secure IoT Thing and its controller [11]; a pattern for a sensor node derived from the sensor AEP can be found in [43]; an ASP for the sensor and a security pattern for radar are in [3]. This means that we need to build security patterns for Lidar sensors and Autonomous Emergency Braking Systems, which will allow us to complete an SSF for sensors and actuators. For that purpose, we will enumerate the use cases of each device and find their threats following the method in [2], where each use case is decomposed into its activities, and we can analyze the possible threats of each activity. Starting from AEPs and their derived patterns provides a systematic way to enumerate threats as opposed to approaches where threats are postulated in ad hoc ways. We first find the threats in an AEP and then in each derived pattern we add the new threats due to their contexts. This method shows the value of having correct functional models to build secure models from them and further justifies our effort in building ESPs.

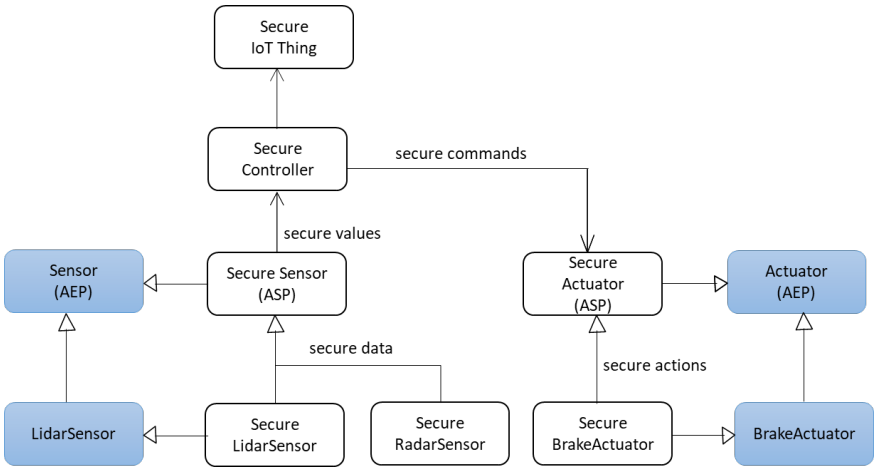

Figure 10. Pattern diagram of the SSFs for Secure Autonomous Driving.

7. Related Work

We are not aware of any work discussing AEPs for sensors and actuators; however, some work discusses AEP-like artifacts for other entities. In [46,47] present a way for applying conceptual patterns in User Interfaces (UI) and show some examples of abstract UI patterns. They explore the impact of these patterns in the software lifecycle and discuss design approaches. However, their patterns are not described using UML class and sequence diagrams or some type of template, as we do.

According to the Oxford Living Dictionary, "an ontology is a set of concepts or categories in a subject area or domain that shows their properties and the relations between them." A related definition is given in [48] that defines an ontology as a specification of conceptualization. The SOSA (Sensor, Observation, Sample, and Actuator) is a light-weight ontology that provides a general-purpose specification for modeling the interaction between the entities which are used for observation, sampling, and actuation [49]. Ref. [49] provides an overview of the main classes and properties of SOSA and briefly discusses their integration with their Semantic Sensor Network (SSN) ontology [50]. They motivate some of the core design decisions (for SOSA ontology design patterns) and focus on an example-driven description of these classes. The SOSA ontology design pattern (which is also known as core structure) provides a base for three modeling perspectives (observing, sampling, and actuating). They also discuss selected modeling problems, how to approach them, and give examples for the usage of SOSA classes and relationships in different application areas. In our work, we focus on describing abstract conceptual entities using patterns, which are described using templates with predefined sections, and from which we derive concrete patterns. Their use of patterns is closer to the normal work of designers and makes our approach more practical.

Ref. [44] defines a specific real-time design pattern for an action subsystem of an Advanced Driver Assistance Systems (ADAS), which models structural, behavioral, and real-time aspects of the automatic actuators and the Human Machine Interface (HMI) elements. [45] presents three patterns for sensing, data processing, and action-taking. Each pattern includes the sections: name, context, problem, forces, solution, consequences, and related patterns; the Sensing patterns specify the various kinds of sensors. The Data Processing pattern defines how to manage data considering their validity duration. The Action-Taking pattern defines how warnings and actuations can be provided to avoid critical situations. Those patterns complement our patterns.

8. Conclusions

Patterns describing conceptual models of systems (Abstract Entity Patterns (AEPs)) are very useful to derive correct concrete patterns for a variety of system types. These patterns significantly simplify the design of complex systems. They are also valuable to build secure versions of fundamental units in critical systems. AEPs can be used to develop concrete patterns for various types of entities. We can then build Entity Solution Frames (ESFs), that are packages of related patterns that can be used to design complex architectures; these ESFs would be complemented with similar packages to describe other parts of the architecture. We have not found a similar design approach in the literature. For concreteness and as proof of concept, we built some AEPs and ESFs for sensors and actuators.

In our future work, we will derive security patterns for lidar and vision camera sensors and for actuators, which will be used to build SSFs for autonomous cars. We will also evaluate the security of the resultant SSF, which can be derived conveniently in system desifgns based on patterns [56].

References

1. Thapa B, Fernandez EB (2020) A Survey of Reference Architectures for Autonomous Cars. Proceedings of the 27th Conference on Pattern Languages of Programs (PLoP '20). The Hillside Group, USA
2. Fernandez EB (2013) Security Patterns in Practice: Designing Secure Architectures Using Software Patterns. John Wiley & Sons.
3. Thapa B, Fernandez EB (2021) Secure Abstract and Radar Sensor Patterns. Proceedings of the 28th Conference on Pattern Languages of Programs (PLoP'21).
4. Thomes S (2021) How Autonomous Emergency Braking (AEB) is redefining safety. https://www.einfochips.com/blog/how-autonomous-emergency-braking-aeb-is-redefining-safety/ (accessed Mar. 18, 2022).
5. Wardlaw C (2021) What is Automatic Emergency Braking? https://www.jdpower.com/cars/shopping-guides/what-is-automatic-emergency-braking (accessed Mar. 18, 2022).
6. Salfer-Hobbs M, Jensen M (2020) Acceleration, Braking, and Steering Controller for a Polaris Gem e2 Vehicle. Intermountain Engineering, Technology and Computing (IETC). pp. 1–6. doi: 10.1109/IETC47856.2020.9249175.

7. Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M (1996) Pattern-Oriented Software Architecture. Vol 1. Wiley Publishing
8. Fernandez EB, Yoshioka N, Washizaki H, Yoder J (2022) Abstract security patterns and the design of secure systems. Cybersecurity. https://doi.org/10.1186/s42400022-00109-w
9. Schmidt DC, Fayad M, Johnson RE (1996) Software patterns. Commun ACM, vol. 39, no. 10, pp. 37–39. doi: 10.1145/236156.236164.
10. Fernandez EB, Washizaki H, Yoshioka N, Okubo T (2021) The design of secure IoT applications using patterns: State of the art and directions for research. Internet of Things, vol 15.
doi: 10.1016/j.iot.2021.100408
11. Fernandez, EB, Astudillo, H, Orellana, C. A pattern for a Secure IoT Thing. 26th European Conference on Pattern Languages of Programs (EuroPLoP'21), July 07–11, 2021, Graz, Austria. ACM, New York, NY, USA. https://doi.org/10.1145/3489449.3489988
12. Fernandez EB, Washizaki H, Yoshioka N (2008) Abstract security patterns. Proceedings of the 15th Conference on Pattern Languages of Programs - PLoP '08. doi: 10.1145/1753196.1753201
13. Gamma E, Helm R, Johnson R, Vlissides J (1994) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley
14. Avgeriou P (2003) Describing, Instantiating and Evaluating a Reference Architecture: A Case Study. Enterprise Architecture Journal, 342:1-24
15. Fernandez EB, Monge R, Hashizume K (2016) Building a security reference architecture for cloud systems. Requirements Engineering, vol. 21, no. 2, pp. 225–249. doi: 10.1007/s00766-014-0218-7
16. Uzunov AV, Fernandez EB, Falkner K (2015) Security solution frames and security patterns for authorization in distributed, collaborative systems. Computers & Security, vol. 55, pp. 193–234. doi: 10.1016/j.cose.2015.08.003
17. Zupan M, Ashby MF, Fleck NA (2002) Actuator Classification and Selection—The Development of a Database. Advanced Engineering Materials, vol. 4, no. 12, pp. 933–940. doi: 10.1002/adem.200290009
18. Behroozpour B, Pandborn PAM, Wu MC, Boser BE (2017) Lidar System Architectures and Circuits. IEEE Communications Magazine, vol. 55, no. 10, pp. 135–142. doi: 10.1109/MCOM.2017.1700030
19. Rablau C, (2019) Lidar: a new self-driving vehicle for introducing optics to broader engineering and non-engineering audiences. Fifteenth Conference on Education and Training in Optics and Photonics: ETOP 2019, vol.11143, Quebec, Canada. doi: 10.1117/12.2523863
20. Haj-Assaad S, (2021) What Is LiDAR and how is it used in cars? https://driving.ca/car-culture/auto-tech/what-is-lidar-and-how-is-it-used-in-cars (accessed Sep. 16, 2021)
21. Torun R, Bayer MM, Zaman IU, Velazco JE, Boyraz O, (2019) Realization of Multitone Continuous Wave Lidar. IEEE Photonics Journal, vol. 11, no. 4, pp. 1–10. doi: 10.1109/JPHOT.2019.2922690
22. Kocic J, Jovicic N, Drndarevic V (2018) Sensors and Sensor Fusion in Autonomous Vehicles. 2018 26th Telecommunications Forum (TELFOR), pp. 420–425. doi: 10.1109/TELFOR.2018.8612054
23. Watanabe N, Ryugen H (2021) Cheaper lidar sensors brighten the future of autonomous cars. Nikkei Asia. https://asia.nikkei.com/Business/Automobiles/Cheaper-lidar-sensors-brighten-the-future-of-autonomous-cars (accessed Sep. 09, 2021)
24. Lidar - Wikipedia. https://en.wikipedia.org/wiki/Lidar (accessed Mar. 18, 2022).
25. Dubayah R, Drake J (2000), Lidar Remote Sensing for Forestry. Journal of Forestry, vol. 98, no. 6, pp. 44–46
26. Hecht J (2018), Lidar for Self-Driving Cars. Optics and Photonics News, vol. 29, no. 1, p. 26. doi: 10.1364/OPN.29.1.000026
27. Kumar GA, Lee JH, Hwang J, Park J, Youn SH, Kwon S, (2020) LiDAR and Camera Fusion Approach for Object Distance Estimation in Self-Driving Vehicles. Symmetry (Basel), vol. 12, no. 2, p. 324. doi: 10.3390/sym12020324
28. Why Optical Phased Array is the Future of Lidar for Autonomous Vehicles - LIDAR Magazine. https://lidarmag.com/2021/08/18/why-optical-phased-array-is-the-future-of-lidar-for-autonomous-vehicles/ (accessed May 29, 2022)
29. Advantages and Disadvantages of LiDAR – LiDAR and RADAR Information. https://lidarradar.com/info/advantages-and-disadvantages-of-lidar (accessed Apr. 02, 2022)
30. Cardei M, Fernandez EB, Sahu A, Cardei I (2011) A pattern for sensor network architectures. Proceedings of the 2nd Asian Conference on Pattern Languages of Programs (AsianPLoP '11), pp. 1–8. doi: 10.1145/2524629.2524641
31. Saida, R, Kacem, YH, BenSaleh, MS, Abid, M (2020) A UML/MARTE Based Design Pattern for a Wireless Sensor Node. In: Abraham, A., Cherukuri, A., Melin, P., Gandhi, N. (eds) Intelligent Systems Design and Applications. ISDA 2018. Advances in Intelligent Systems and Computing, vol 940. Springer, Cham. https://doi.org/10.1007/978-3-030-16657-1_55
32. Sahu A, Fernandez EB, Cardei M, Vanhilst M (2010) A pattern for a sensor node. Proceedings of the 17th Conference on Pattern Languages of Programs (PLOP '10), pp. 1–7. doi: 10.1145/2493288.2493295
33. Ross P (2021) Europe Mandates Automatic Emergency Braking. IEEE Spectrum. https://spectrum.ieee.org/europe-mandates-automatic-emergency-braking (accessed Mar. 18, 2022)
34. A quick guide to ADAS | Delphi Auto Parts. https://www.delphiautoparts.com/usa/en-US/resource-center/quick-guide-adas (accessed Mar. 18, 2022).
35. Szuszman P (2005) Adaptive cruise control system overview. 5th Meeting of the U.S. Software System Safety Working Group, Anaheim, California, USA

36. Kurczewski N (2021) Best Cars with Automatic Emergency Braking. U.S. News. https://cars.usnews.com/cars-trucks/best-cars-with-automatic-emergency-braking-systems (accessed Mar. 18, 2022).
37. What is autonomous emergency braking?. Kia British Dominica. https://www.kia.com/dm/discover-kia/ask/what-is-autonomous-emergency-braking.html (accessed Apr. 02, 2022).
38. Kapse R, Adarsh S (2019) Implementing an Autonomous Emergency Braking with Simulink using two Radar Sensors. Cornell University. https://arxiv.org/abs/1902.11210. doi: https://doi.org/10.48550/arXiv.1902.11210
39. Tesla (2022) Collision Avoidance Assist. Model 3 Owner's Manual. https://www.tesla.com/ownersmanual/model3/en_us/GUID-8EA7EF10-7D27-42AC-A31A-96BCE5BC0A85.html#CONCEPT_E2T_MSQ_34 (accessed Mar. 18, 2022).
40. Laursen L (2014) Autonomous Emergency Braking. IEEE Spectrum. https://spectrum.ieee.org/autonomous-emergency-braking (accessed Mar. 18, 2022).
41. Healy J (2019) 10 best cars with automatic emergency braking. Car Keys. https://www.carkeys.co.uk/guides/10-best-cars-with-automatic-emergency-braking (accessed Mar. 18, 2022).
42. Doyle L (2020) Can brake assist be turned off?. ici2016.org. https://ici2016.org/can-brake-assist-be-turned-off/ (accessed Apr. 02, 2022).
43. Orellana C, Astudillo H, Fernandez EB (2021) A Pattern for a Secure Actuator Node. 26th European Conference on Pattern Languages of Programs (EuroPLoP), pp. 1–6. doi: 10.1145/3489449.3490007.
44. Marouane H, Makni A, Duvallet C, Sadeg B, Bouaziz R (2013) A Real-Time Design Pattern for Actuators in Advanced Driver Assistance Systems. 8th International Conference on Software Engineering Advances (ICSEA), pp. 152–161.
45. Marouane H, Makni A, Bouaziz R, Duvallet C, Sadeg B (2016) Definition of Design Patterns for Advanced Driver Assistance Systems. Proceedings of the 10th Travelling Conference on Pattern Languages of Programs - VikingPLoP '16, pp. 1–10. doi: 10.1145/3022636.3022639.
46. Molina PJ, Meliá S, Pastor O (2002) User Interface Conceptual Patterns. Interactive Systems: Design, Specification, and Verification, 9th International Workshop, DSV-IS 2002, Rostock Germany, pp. 159–172. doi: 10.1007/3-540-36235-5_12.
47. Grone B (2006) Conceptual patterns. 13th Annual IEEE International Symposium and Workshop on Engineering of Computer-Based Systems (ECBS'06), Potsdam, Germany, pp. 241– 246. doi: 10.1109/ECBS.2006.31.
48. Gruber T (1995) Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, vol. 43, no. 4–5, pp. 907–928
49. Janowicz K, Haller A, Cox S, Phuoc DL, Lefrançois M (2019) SOSA: A lightweight ontology for sensors, observations, samples, and actuators. Journal of Web Semantics, vol. 56, pp. 1–10. doi: 10.1016/j.websem.2018.06.003.
50. Haller A, Janowicz K, Cox S, Phuoc DL, Taylor K, Lefrançoi M (2021) Semantic sensor network ontology. https://www.w3.org/TR/vocab-ssn/ (accessed Nov. 21, 2021)
51. Thomson C (2019) What are point clouds? 5 easy facts that explain point clouds. Vercator. https://info.vercator.com/blog/what-are-point-clouds-5-easy-facts-that-explain-point-clouds (accessed Jun. 10, 2022)
52. Leberl F, Irschara A, Pock T, Meixner P, Gruber M, Scholz S, Wiechert A (1010) Point Clouds. Photogrammetric Engineering & Remote Sensing, vol. 76, no. 10, pp. 1123–1134. doi: 10.14358/PERS.76.10.1123
53. Fernandez EB, Washizaki H (2022) "Abstract Entity Patterns (AEPs)", submitted for publication.
54. Wieringa R (2009). "Design Science as nested problem solving". 4th International Conference on Design Science Research in Information Systems and Technology.
55. Rumbaugh J, Jacobson I, Booch G, *The Unified Modeling Language Reference Manual* (2nd Edition), Addison-Wesley, 2005.
56. Villagran-Velasco Olga, Fernandez E.B., Ortega-Arjona J., Refining the evaluation of the degree of security of a system built using security patterns, Procs. *15th Int. Conference on Availability, Reliability and Security* (ARES 2020), Dublin, Ireland, Sept. 2020.