

Article

Timing Predictability and Security in Safety-Critical Industrial Cyber-Physical Systems: A Position Paper [†]

Saad Mubeen ^{1,*},[‡][§] , Elena Lisova ^{1,‡}[§]  and Aneta Vulgarakis Feljan ^{2,§}

¹ School of Innovation, Design and Engineering, Mälardalen University, 72123 Västerås, Sweden; elena.lisova@mdh.se

² Ericsson Research, 16440 Kista, Sweden; aneta.vulgarakis@ericsson.com

* Correspondence: saad.mubeen@mdh.se

[†] This paper is an extended version of paper published in the 20th IEEE International Conference on Industrial Technology, ICIT 2019, Melbourne, Australia, 13–15 February 2019.

[‡] Current address: Högscoleplan 1, 72123 Västerås, Sweden.

[§] These authors contributed equally to this work.

Received: 30 March 2020; Accepted: 26 April 2020; Published: 30 April 2020



Abstract: Cyber Physical Systems (CPSs) are systems that are developed by seamlessly integrating computational algorithms and physical components, and they are a result of the technological advancement in the embedded systems and distributed systems domains, as well as the availability of sophisticated networking technology. Many industrial CPSs are subject to timing predictability, security and functional safety requirements, due to which the developers of these systems are required to verify these requirements during their development. This position paper starts by exploring the state of the art with respect to developing timing predictable and secure embedded systems. Thereafter, the paper extends the discussion to time-critical and secure CPSs and highlights the key issues that are faced when verifying the timing predictability requirements during the development of these systems. In this context, the paper takes the position to advocate paramount importance of security as a prerequisite for timing predictability, as well as both security and timing predictability as prerequisites for functional safety. Moreover, the paper identifies the gaps in the existing frameworks and techniques for the development of time- and safety-critical CPSs and describes our viewpoint on ensuring timing predictability and security in these systems. Finally, the paper emphasises the opportunities that artificial intelligence can provide in the development of these systems.

Keywords: timing predictability; cyber-physical systems; CPS; functional safety; security; embedded systems; CPS eco-system

1. Introduction

There exist several definitions of Cyber-physical systems (CPSs) in the literature. For example, according to the International Conference on CPSs (ICPPS) (<http://iccps.acm.org>), CPSs are defined as the “physical and engineered systems whose operations are monitored, coordinated, controlled, and integrated by computing and communication. In other words, CPSs are the systems with a coupling of the cyber aspects of computing and communications with the physical aspects of dynamics and engineering that must abide by the laws of physics”. According to Lee and Seshia [1,2], CPSs are described as the systems that emphasise the link between computation and physical processes, thereby linking time, space and energy. The physical processes are monitored and controlled by computers that are embedded within these systems (also called embedded computers), and vice versa the computations are affected by the physical processes. This paper focuses on Industrial CPSs (ICPSs) that are commonly found in the automation and automotive domains, among others. Note that we overload

the term CPS to refer to ICPSs. The radical transformation from an *embedded system* to a CPS comes from the emphasis on integration of physical processes and a more broad networking aspect. Similar to CPSs, there exist several definitions of embedded systems and there is no agreed-upon definition of these systems, as stated by Li and Yao [3]. According to Barr and Massa [4,5], an embedded system performs a dedicated functionality by means of computer hardware, software and perhaps additional mechanical components, sensors and other parts.

As a CPS can include one or more embedded systems, it is crucial to clearly define the boundary of embedded system (s) within the context of cyber-physical systems, which is often not clear. In this regard, an embedded system is defined as the system consisting of hardware, software and interfaces (ports) to receive/send sensor/actuator signals and network messages. The inputs (e.g., sensor signals) arriving at the input interface and the computed outputs (e.g., actuation signals) delivered to the output interface of the system are considered to be parts of the embedded system, as depicted in Figure 1. The sensors, actuators and physical processes that are sensed and controlled, respectively, are not considered to be parts of the embedded system. These entities, together with the embedded system and possibly communication with the rest of the eco-system constitute a CPS, as depicted in Figure 1. By an eco-system, we understand a set of communicating CPSs as well as their surrounding computation and communication infrastructure, i.e., realised via fog nodes, that can provide some control functionalities as well as a communication channel to Cloud. Thus, a CPS eco-system includes entities with which it interacts as well as the entities that are required by the communication infrastructure. To ease depicting of communication connections, we introduce a conceptual interface layer that captures all connections towards and from a CPS. Let us consider the example of the airbag system in a car. If the car crashes and its deceleration is fast enough, the crash sensors are triggered, which send crash signals to the computation unit that computes and produces the actuation signals to inflate the airbag. The sensor inputs, the computation unit (both hardware and software) and the actuation outputs are parts of the embedded system. The embedded system together with the crash sensors and the physical process (the environment) in which they are deployed, the airbag actuator and the airbag itself constitute the CPS. It should be noted that, if two or more embedded systems are connected through an on-board bus or a network, the system is still considered to be an embedded system (distributed embedded system to be precise), as shown by the on-board interconnection of several embedded systems in Figure 1.

The CPS in the above example is a time- and safety-critical system. The system is required to provide a logically correct response (or output) within a given amount of time that corresponds to the specified timing predictability requirement. Failing to meet a timing predictability requirement in a time-critical system can result in the system failure. Failing to meet a timing predictability requirement in a safety-critical system can have catastrophic consequences such as endangerment of human life or the environment. Assume that the car in which the CPS is deployed is an autonomous vehicle that cooperates with other vehicles and road-side units (RSUs) to perform a certain cooperative functionality, e.g., navigate collaboratively while avoiding accidents with other cars, pedestrians and stationary objects. The above-mentioned CPS in the car together with the corresponding CPSs inside the other vehicles or RSUs provide an example of a safety-critical CPS. Industrial environment of safety-critical CPSs often brings tighter timing requirements together with an assumed infrastructure that provides connectivity and can play a role of fog nodes. Generally speaking, a system is said to be predictable if its state/behaviour can be forecasted at any point in time, given a known execution environment or a set of assumptions. The prediction can be made either quantitatively or qualitatively, and at different stages during the system lifecycle. Predictability of a system is related to proving, demonstrating or verifying the fulfillment of the functional or extra-functional requirements that are specified on the system. The focus of this paper is on the system development stage or the so-called design time and on two extra-functional requirements that affect the predictability of a safety-critical CPS: timing and security. The term predictability has also been used lately in the artificial intelligence (AI) community, where a system is capable of predicting the future state changes and executing appropriate actions beforehand (e.g., for predictive maintenance). Triggered by this usage of the term,

we include a brief discussion on how AI can be used as a tool for addressing challenges in the light of system predictability.

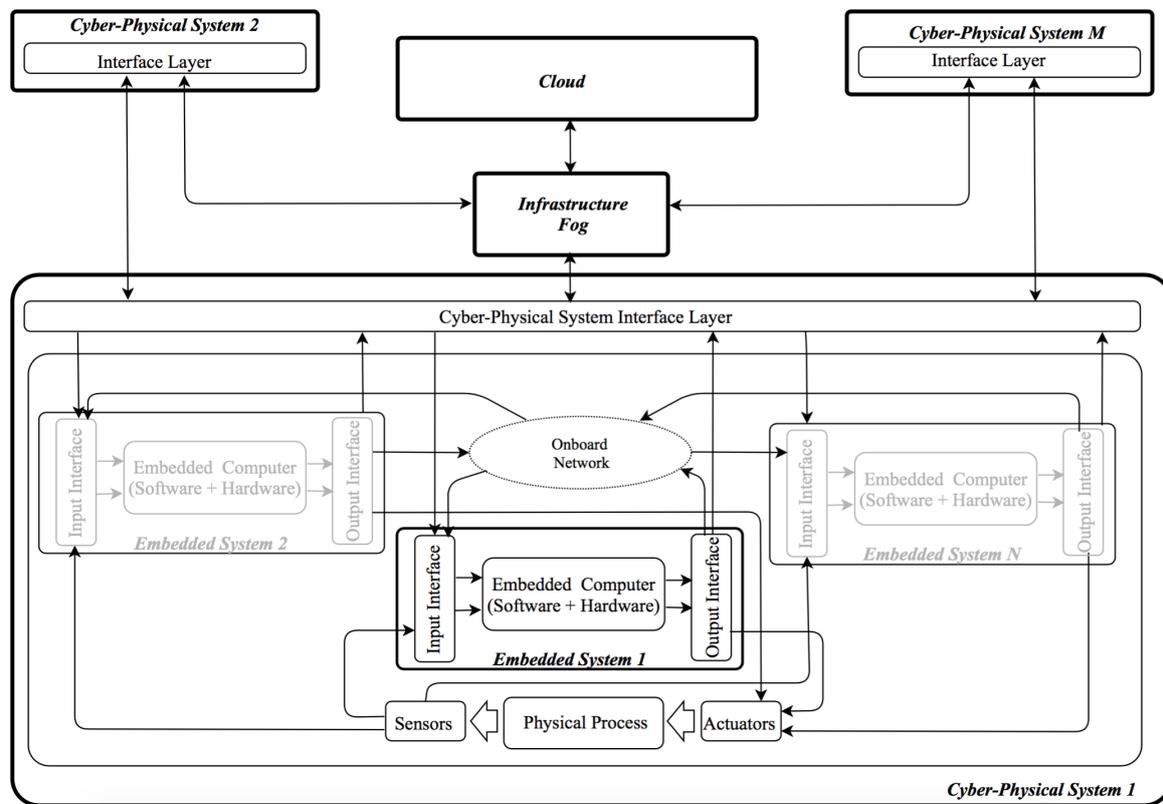


Figure 1. Defining the boundaries of embedded systems and CPSs in the CPS eco-system.

There is a plethora of existing research and initiatives that define and study timing predictability [6–10] and security [11–13] in the embedded systems community. Security in a broad sense can be defined as a system property that allows it “to perform its mission or critical functions despite risks posed by threat” [14], where a threat can be defined as “the potential source of an adverse event” [14]. A threat is realised by an attack that exploits a vulnerability, i.e., a flaw in the system, and targets one of the system assets. A concrete threat realisation is an attack. One of the main security objectives in embedded systems is to consider data integrity and authenticity, as it is crucial to have enough confidence that the data received from the sensors represents the physical process correctly. That is, the data are not modified by an adversary or injected by an adversary masking the real data. To ensure that the input data is correct and not modified maliciously, we can as well use prediction algorithms (extrapolation) and also security mechanisms for integrity and authentication checks. In this paper, we focus only on the two security objectives mentioned above as being most common for the time- and safety-critical CPSs; however, depending on the particular use-case, the relevant security objectives for industrial CPSs can include confidentiality, anonymity, availability, auditability, non-repudiability, third-party protection and conformance [15].

1.1. Paper Contributions

The main objective of this position paper is to conduct an investigation of the key issues involved in supporting and verifying timing predictability and security in safety-critical industrial CPSs during their development. In this regard, we pose three main Research Questions (RQs), as depicted in Figure 2 and presented at the end of this subsection. Before answering the research questions, we present a concrete structure of the CPS eco-system providing clear boundaries among various constituting components, as shown in Figure 1. To answer the posed research questions, we first explore the state

RQ (ii) How can the existing solutions be extended to support the development of time-critical, secure and safety-critical CPSs?

1.2. Paper Outline

The rest of the paper is organised as follows. Section 2 introduces an autonomous quarry as a running example for the paper, whereas Section 3 discusses timing predictability for embedded systems and CPSs. Next, Section 4 points out the bounding of timing predictability, security and functional safety. Section 5 outlines the authors position regarding the discussed challenges. Finally, Section 6 concludes the paper.

2. Running Example: Autonomous Quarry

We use a running example of an autonomous quarry [16] to illustrate the proposed ideas in this paper. The autonomous quarry shown in Figure 3 consists of several stages. Large stones are extracted from the extraction site by the excavators. The extracted material is then transported to the crushing site with the help of battery-powered autonomous haulers. At the crushing site, the large stones are crushed by a crusher machine. The crushed stones are then transported by the autonomous haulers that cooperate with each other for efficient transportation of the crushed material. It is critical for the production efficiency that the haulers arrive at the battery charging stations in time to avoid stopping unnecessarily in the middle of the quarry with drained batteries. To optimise the battery charge, the haulers should not approach the charging stations too early with still enough amount of remaining charge in their batteries.

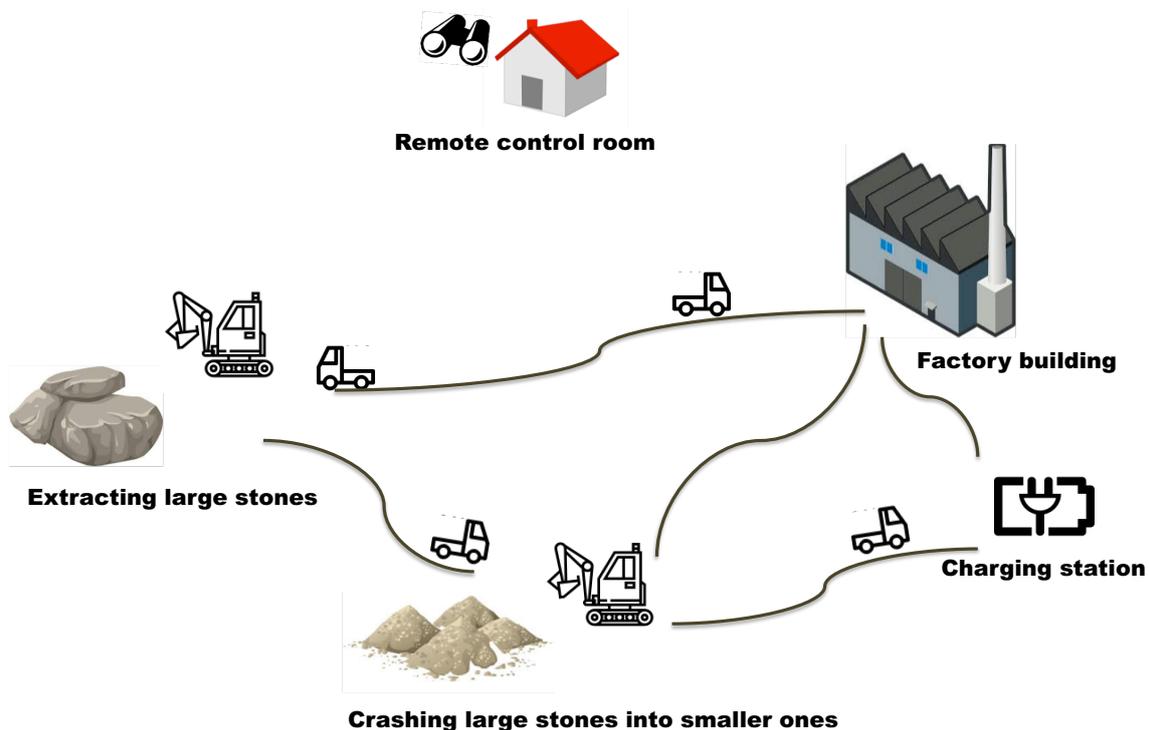


Figure 3. A running example of an autonomous quarry.

Another important aspect in this regard is the safe and efficient transportation. The haulers need to be equipped with an updated map of the quarry providing available routes and location of the static objects along the routes to prevent any possible accidents. For example, crashing into obstacles or falling into pits needs to be avoided. Each hauler needs to be aware of the location information of all the other haulers to avoid collisions with each other. The haulers are assumed to receive this

information (e.g., a map of dynamic objects) from a remote control centre, i.e., an infrastructure in this case. The remote control centre has a pre-established communication link with each hauler in the quarry. Furthermore, each hauler receives the control information, consisting of speed, direction and required actions from the remote control centre. In the case of an immediately detected hazard or a communication failure, the haulers are capable of overriding a command from the remote control centre. In this regard, the hauler can fully rely on the information received from the on-board sensors. Note that these systems operate in harsh environments, e.g., due to extreme amount of dust. In addition, these systems also share the environment with humans, thus safety is a crucial property to assure. In these systems, support for timing predictability is crucial in assuring safety. Interestingly, wireless communication channels and increased connectivity among the vehicles impose security threats that can affect predictability, thereby jeopardising the system's safety.

An analysis of such a quarry can provide an example of a possible hazard: the navigation and collision hazard. This hazard can occur due to the following reasons [17]: (i) a failure to timely detect an object; (ii) increased latency due to a computation load of a processor being used for object detection and localisation; (iii) incorrect localisation of a detected object; (iv) inability to stop the vehicle remotely or in an emergency state; (v) lack of access to situational awareness information; (vi) incorrect terrain data; (vii) lost or delayed command input; (viii) inaccurate positioning caused by a loss of the Global Navigation Satellite System (GNSS) correction; and (ix) incomplete or improper system updates and changes to the software.

3. Timing Predictability in CPSs

This section presents a comprehensive discussion on timing predictability in time-critical embedded systems and CPSs. Based on this discussion, the section advocates to consider timing predictability as a prerequisite for functional safety.

3.1. Predictability in Time-Critical Embedded Systems

This subsection discusses timing predictability in embedded systems with regards to on-board embedded systems in each hauler in the example provided in Section 2. In time-critical embedded systems, it is required that all actions by these systems are performed in a timely manner such that all specified timing requirements are met. Hence, at the design time, the developers of these systems need to verify that the systems are timing predictable. Timing predictability is a well-defined concept in the real-time systems domain [6–10]. For a given system model and a set of assumptions, the system is considered to be timing predictable if it is possible to show, prove or demonstrate that all specified timing requirements will be satisfied when the system is executed.

The timing requirements can be specified on various elements of the system model, e.g., on the individual tasks, set of tasks and task chains in a node, network messages and chains of tasks and messages in a distributed embedded system. Note that a node can be a single-core processor or a multi-core processor. Traditionally, the timing requirements referred to the deadlines corresponding to the response times of individual tasks, messages and task chains [18]. The response time of a task, message or a task chain is counted from the arrival of the input value at the input interface until the delivery of the corresponding computed output value. The response time of a task, message or a task chain shall not exceed the corresponding deadline. In the past few years, the research community extended the use of timing requirements beyond the traditional deadline requirement by considering several other timing requirements by means of timing constraints. The reaction and age constraints are two such constraints, among others. These requirements have been included in several domain-specific modelling languages [19], e.g., Timing Augmented Description Language (TADL2) [20], EAST-ADL [21] language, the Rubus Component Model (RCM) [22] and AMALTHEA [23]. These requirements have also been incorporated with the domain-specific standards such as the automotive standard AUTOSAR [24].

The predictability of a time-critical embedded system is supported by verifying its timing behaviour at the design time and providing a support for predictable execution environment at the runtime. The timing behaviour of the system can be verified by using the schedulability analyses, whereas the predictable runtime environment can be provided by means of a real-time operating system (RTOS). There is a wealth of schedulability analysis techniques developed by the research community [18,25,26]. Furthermore, there are many RTOSs that support predictable runtime environment for embedded systems, e.g., VxWorks, Rubus and FreeRTOS, to mention a few. In summary, the development of timing predictable embedded systems has already achieved significant level of maturity.

3.2. Predictability in Time-Critical CPSs

The time-critical embedded systems represent only one component of a time-critical CPS. Therefore, the span of timing predictability in the CPS should extend the boundaries of embedded systems to include the timing predictability impact of sensors and actuators that are deployed in the physical processes. The sensor values in the running example of the autonomous quarry can arrive from the other haulers, RSUs or the remote control centre. In this regard, we identify the following two key parameters [27].

1. *Ready Time of Inputs:* The ready time of inputs in the CPS is referred to as the interval of time between the instant when the value of a sensor that is deployed in the physical process changes and the instant when the changed value appears at the input interface of the embedded system. The ready time of all sensor inputs coming from the physical processes should be timing predictable.
2. *Ready Order of Inputs:* In the case of more than one sensor value arriving at the input interface of an embedded system, the computed output that controls the physical processes depends heavily on the arrival order of the inputs. The desired function of the CPS requires a specific arrival order of the inputs from the sensors that are deployed in the physical processes, which must be timing predictable. This is referred to as the order predictability.

It is interesting to note that these two parameters are not considered in the definition of timing predictability for embedded systems as they exist outside the interfaces of these systems. A CPS is considered to be timing predictable if it can be shown, proven or demonstrated that all the specified timing requirements are satisfied at the design time. These requirements include the timing requirements on the computations and communications as well as the timing requirements that constrain the ready time and ready order of the inputs. If a CPS offloads computations to the cloud, the definition of timing predictability should additionally consider the timing impact of the offloaded computations [28]. We argue that, even if a time-critical CPS is proven to be timing predictable at the design time, the predictability of the system can be jeopardised at runtime due to security threats to the time-critical data entering the system, e.g., from sensors, networks or other CPSs. Therefore, security of the data is integral to the timing predictability of the system. The security aspects are discussed in the next sections.

3.3. Timing Predictability as a Prerequisite for Functional Safety

Based on the discussion in the previous subsection, we make a strong case about timing predictability being a prerequisite for functional safety in time-critical CPSs. That is, if the CPS cannot be verified to be timing predictable at the design time, then it cannot be guaranteed to meet one or more of the timing requirements at run-time. Intuitively, the functional safety of the system cannot be guaranteed. For example, consider the navigation and collision hazard in the autonomous quarry discussed in Section 2. In this example, timely detection of objects is crucial in avoiding the hazard. If the timing predictability verification techniques and tools at the design time conclude that any of the timing requirements corresponding to the object detection functionality cannot be satisfied, the

system is deemed not timing predictable. This, in turn, implies that, if such a system is run, then the safety hazard can occur at runtime. Therefore, timing predictability should be considered as a vital prerequisite to functional safety of time-critical CPSs.

4. Security in Time-Critical CPSs

This section discusses the design-time security challenges in embedded systems and relates them to the security of time- and safety-critical CPSs. Further, it explores the feasibility of corresponding security solutions in embedded systems to the security challenges in the CPSs.

4.1. Security Challenges and Solutions in Embedded Systems and CPSs

The security challenges in embedded systems have been extensively addressed in the literature [11]. The main security challenges for embedded systems at the design time are the following.

1. *Resource Gap*: Many embedded systems struggle to fulfil the requirements on computation and energy consumption required for security solutions.
2. *Flexibility*: Given that security is dynamic in nature, it can be challenging for embedded systems (which are often static) to provide a platform that is flexible enough and able to support constant security updates.
3. *Tamper Resistance*: Embedded systems struggle to counteract attacks caused by malware, which are capable of executing downloaded applications.
4. *Security Assurance*: Assurance of security for embedded systems that tend to have increased complexity is a challenge.
5. *Cost*: Security solutions are usually costly, hence it is a challenge to find the right balance in regard to an acceptable security level and the system design investment given low-cost devices.

To address the first challenge of limited resources, many lightweight security solutions have been developed by the research community [29,30]. Overall, this challenge does not concern CPSs as they can have relatively more resources, e.g., the haulers in the running example may contain large batteries, powerful processors (e.g., multi-cores) and onboard networks supporting a high-bandwidth (e.g., CAN FD [31] or Ethernet). Hence, CPSs do not have strict resource limitations from a security perspective.

CPSs inherited the second challenge concerning development of flexible platforms supporting security updates, from embedded systems. Even though this challenge is already addressed in embedded systems [32], it is yet to be tackled in CPSs, which have a higher connectivity compared to embedded systems and hence possess higher risks and stricter security requirements. The hauler in the running example has more vulnerabilities that can be exploited by security attacks as compared to an ES (e.g., a brake-by-wire system), due to having more attack surfaces, e.g., the hauler receives time-sensitive information via wireless links from the other haulers and the control centre. Thus, an investigation is required to check feasibility of the solutions from the embedded systems domain for CPSs.

The challenge of tamper resistance is addressed for embedded systems via lightweight security solutions and overall by incorporating security considerations into the system design. For CPSs which have an increased connectivity and especially for safety-critical CPSs, the challenge requires a dedicated effort starting from the concept phase of the system design.

Security assurance complex system is challenging as the system security is not composable, i.e., a security level of a system composed from components cannot be argued only upon security levels of those components. Hence, given a secure ES as a part of the CPS, it is not straightforward to assure an overall system-level security. The assurance of embedded systems is addressed for real-time properties [33] and safety [34], e.g., the ISO 26262 [35] functional-safety standard for road vehicles provides guidelines for assuring that any unreasonable risks due to malfunctions of electrical and electronic systems are mitigated or prevented. However, there is still work to be done for solving the challenge

of security assurance for embedded systems. The notion of the security assurance case exists [36]; however, there are not yet that many works in the area. The main showstopper is the dynamic nature of security due to new threats and vulnerabilities constantly being discovered [37]. In the case of time- and safety-critical CPSs, this challenge gets even more complex as the system's decisions rely on time-sensitive information coming from other systems often via wireless communication links.

Given the cost of security solutions for embedded systems, there is a number of risk assessment techniques exploring a trade-off between the possible possessed risk and the solution to avoid it. Similar to the As Low As Reasonable Practicable (ALARP) [38] from the safety domain, an appropriate level of system security is defined when the resources required to be invested in breaching the security are compared to the value of the system assets. Many time-critical and cooperative CPSs are safety-critical, meaning that the expenses threshold for the design phase of the system is high due to the criticality level.

4.2. Security as a Prerequisite for Timing Predictability

Time predictability is based on assumptions regarding the considered system and surrounding it environment. One of the common assumptions in embedded systems is systems' input integrity, i.e., that the inputs are not forged, deleted, injected or modified by an adversary.

However, the possibility of an adversary to physically access the system is not captured by the classical assumptions of time predictability in ESs. Thus, time predictability has an implicit dependency on the input data security. Given the shift from embedded systems to CPSs, a concern regarding the assumption covering security rises even higher, as the system becomes more open and complex. For CPSs in industrial context, increased connectivity of these systems makes securing the data integrity more challenging. It also brings new attack surfaces and possible vulnerabilities unless security is not addressed properly at the design time. For instance, once the time predictability of the autonomous haulers in the considered example is analysed, it can be guaranteed under a set of assumptions, which includes the assumption regarding the sensor data integrity being intact. An autonomous hauler receives command information regarding its movement from the control centre. The assumption is also that the hauler has local intelligence that allows it to sustain a temporary loss or disruption of control information, e.g., caused by a failure in the communication channel [39]. To make its own decisions about its current actions, a hauler requires an updated map of the quarry and the approximated location of the other vehicles. If command information, e.g., the speed, acceleration and direction, can be forged by an adversary, the time predictability of the system verified at the design time will not hold any more. One more possible scenario can be created by a failure occurring in the communication channel due to communication quality degradation or jamming, the time predictability of the system can be jeopardised, e.g., by forged sensor information or map. Hence, to support time predictability in time- and safety-critical industrial CPSs, the systems' security must also be supported.

4.3. Security as a Prerequisite for Functional Safety

Safety and security are interconnected and can influence each other; there is a recognised need to consider them jointly [40]. A connected safety-critical CPS, cannot be argued as being acceptably safe unless security impact on safety is considered [41]. Security breaches may lead to safety hazards and in this way change the probability of their occurrence, which leads to a necessity to re-assess risks. The term security informed safety can be applied in the case when safety is the overall goal of an effort and security is considered as a supportive property.

Safety as well as security is not composable, thus an impact of CPS security on CPS safety has to be considered on the system level. There are many research works focusing on safety and security co-analysis [42] and its different phases such as requirements engineering or Hazard Analysis and Risk Assessment (HARA) [43] and Threat Assessment and Remediation Analysis (TARA) [44]. However, one of the main remaining gaps in the direction of safety and security co-analysis, is run-time support.

Looking at the example of an autonomous quarry the navigation and collision hazard can be triggered by security causes such as Denial of Service or forgery attacks [17], i.e., by an attacker making a communication media unavailable or modifying control messages. Incorporating such security causes into safety analyses implies providing safety arguments that cover security considerations and performing a joint safety-security risk assessment. In an ideal case, safety and security process are joint to fully capture all interconnections and make work on the dependencies more efficient [45].

5. Position and Opportunities for Further Research

5.1. Timing Predictability of Safety-Critical CPSs

To support the development of time-critical CPSs, the research questions posed in Section 1 are refined as follows.

1. Are the existing models, languages and frameworks for the development of the CPSs expressive enough to specify the timing requirements not only on the computation and communication times but also on the ready times and ready order of the inputs that are acquired from the physical processes?
2. Are there any existing methods and techniques that can formally verify the specified timing requirements at the design time to support pre-runtime timing predictability verification of the CPSs?

If the answer to Research Question 2 is “yes”, can the identified techniques efficiently support and be integrated to the existing safety analyses?

3. Are there any existing techniques to support timing predictable run-time environment for the CPSs that can provide bounded delays with regards to the computation, communication, ready times and ready order of the sensor inputs?

To answer the first research question, we explore the existing development models, languages and frameworks for CPSs. The existing works in the computation and communication parts support the specification of timing requirements, which are sufficient to support the corresponding part of time-critical CPSs. For example, the timing model in the AUTOSAR standard (used in the automotive domain) provides 21 different timing constraints that can be specified on the system. Similarly, the EAST-ADL software architecture description language and several other component models including RCM support the specification of the timing requirements [22,46]. Another example can be seen in the avionics domain, where several existing frameworks support the specification of timing requirements [47,48].

However, we identify that the existing techniques, methods and frameworks do not support the specification of timing requirements corresponding to the input ready times and the input ready order. There are a few recent works that discuss the input ready times and the input ready order in the context of CPSs [27]. However, the formal semantics of these terms and their corresponding timing requirements are still missing from the state of the art. This, in turn, hampers the specification of holistic or end-to-end timing requirements in time-critical CPSs, i.e., the timing requirements that constrain the delivery time of the output of an actuator corresponding to the time when a new input is generated from a sensor that is deployed in the physical process.

The timing model of the AUTOSAR standard includes several timing constraints, e.g., the *Order Constraint*. This constraint is also supported by several other modelling languages such as EAST-ADL, TADL2 and RCM. We believe this constraint can be extended to support the input ready order timing requirement in the CPSs. Basically, the Order Constraint constrains an order among the occurrences of events [22,24]. If this constraint is adapted to constrain the arrival order of the sensor values from the physical process, then it can be applied to constrain the input ready order in the CPSs. On the other hand, the semantics of the input ready times and the corresponding requirement need to be properly

defined and included in the existing techniques and frameworks that are used for the development of the CPSs.

The second research question is answered by exploring the existing schedulability techniques [18,25,49–52]. The support to verify timing predictability at the design time in the computation and communication parts are quite mature. If the Order Constraint is extended to support the specification of timing requirement on the inputs arrival order in CPSs, the corresponding timing analysis can be used to verify this requirement at the design time [22]. To the best of our knowledge, the timing analysis to verify the timing requirements on the input ready times is still missing from the state of the art.

Regarding the sub-question of the second research question, several existing timing predictability verification techniques for the computation and communication parts have already been integrated to the existing safety processes. For example, the classical safety analysis Failure Mode and Effects Analysis (FMEA) [53] includes such failures modes as late or early relating to the timing of inputs.

However, to the best of our knowledge, the integration of timing predictability of inputs arrival order and inputs ready times to the existing safety analyses is yet to be done.

The answer to the third research question is similar to the answer of the second research question. The existing execution frameworks and tools support predictable runtime environments in the computation and communication parts, as discussed in Section 3.1. However, these frameworks and underlying techniques need to be extended to provide upper bounds on the input ready times and to enforce the inputs ready order.

5.2. Security of Safety-Critical CPSs

To address the security aspects of safety-critical CPS, the research questions discussed in Section 1 are refined as follows.

1. Are the existing models, languages and frameworks for the development of CPSs expressive enough to specify security requirements in CPSs?
2. Are there any existing methods and techniques that can formally verify the specified security requirements in CPSs?

If the answer to Research Question 2 is “yes”, can the identified techniques efficiently support and be integrated to the existing safety analyses?

3. Are there any existing techniques to support secure run-time environment for CPSs?

It can be observed that there exist several approaches, analyses and even frameworks addressing security in time- and safety-critical industrial CPSs. Although, the systematic incorporation of security into the system development during the design time and its run-time assessment at the operational phase, which is of special importance for safety-critical CPSs, is not yet mature. The solutions considered in this paper that support time time predictability hold only if the data integrity, authentication and authorisation and other relevant security objectives are supported. Given the strong dependency between timing predictability, safety and security, this paper states security as a prerequisite for timing predictability in time- and safety-critical CPSs.

In the development secure CPSs, one of the main challenges is coming from the fact that a particular solution (e.g., a hash-function insuring messages integrity) cannot provide any guarantees just by itself, as its implementation and security policies specifying the solution usage play a crucial role in verification of whether it actually covers the required security objective. Hence, we advocate addressing security via the top-down approach at the system level and building a security assurance case in order to systematise the way security is provided and supported. An assurance case can be defined as “an enabling mechanism to show that the system meets its prioritized requirements” [54]. The system trustworthiness and reasoning upon it is the core of an assurance case. It can be built for a system property such as safety [55], security [36] or ethics [56]. We envision the security assurance case

as a way to collect and structure arguments over the system being acceptably secure. Being dynamic by its nature, security requires run-time updates and refinements, e.g., patches. However, it is not feasible in terms of time, money and efficiency to develop a security case from the scratch each time there is an update. The challenge of handling the updates in an efficient way within a security case is a gap in the current research that needs to be addressed. There are also works on joint safety and security assurance cases [42] that allow addressing security-informed safety.

Now, we answer the research questions (posed in Section 1) given the state of the art in safety-critical CPSs. First, there are several existing methods for requirements elicitation in the security domain [57,58] as well as for joint consideration of safety and security [59,60]. Second, there are many existing techniques and tools for the formal verification of these requirements [61,62]. Moreover, many works are aiming on alignment of safety and security process in their different phases including requirements engineering one [42]. However, there is a gap in supporting run-time frameworks that can provide a secure run-time environment for the systems, which is crucial given how often updates and patches can be required.

5.3. Artificial Intelligence for Safety-Critical ICPS

CPS are challenging to get right, therefore formal verification provides rigorous ways of establishing safety of controllers with respect to a physical model of the system under control. However, formal verification can guarantee safe system operation, but only if there are no discrepancies between the real implementation and the verified model. Such discrepancies between reality and model are imminent in physical systems operating in open environments. Moreover, to overcome complexity challenges and to make it possible to be analysed, often the verifiable models of a system use simplifying abstractions.

CPSs can leverage artificial intelligence and machine learning (ML) algorithms to act well in open environments, and therefore, in recent years, there has been a big interest in applying ML components into safety-critical CPS. In difference to formal methods, ML does not need a knowledge of the behaviour of the whole system and, instead, uses learning algorithms that generalise responses from static data (e.g., a set of labelled images to classify) or from dynamic experience (e.g., responses to trial and error). However, it is challenging to achieve high-levels of safety assurance due to the complexity and unpredictability of ML techniques. Moreover, ML algorithms also raise additional safety issues since: (1) most expressive and powerful ML models are not transparent and behave as a black box; and (2) the ML training data are usually incomplete. In reinforcement learning (RL) [63], for example, a controller or an agent perceives the state of the environment, and it acts in order to maximise the long-term return that is based on a real valued reward signal, without a need for a perfect model of the environment. The RL agent needs to balance exploitation with exploration, where exploitation is the strategy to select best action based on previously learned policy (i.e., behaviour), while exploration is a strategy to search for better policies using actions outside the current, learned, policy. Exploration creates opportunities, but also induces risk that actions selected during this phase will not generate increased reward. Therefore, most approaches towards reinforcement learning provide no guarantee about the safety of the learned controller or about the safety of actions taken during learning, which is against the best practices demands of safety critical CPS, such as planes [64] or vehicles [65]. Safe reinforcement learning [66] is an emerging field in artificial intelligence that addresses control problems in which it is important to respect safety constraints.

Another aspect related to artificial intelligence and CPS is the introduction of cloud platforms. While the fog and cloud bring immense computational power to CPS, which can be further used for artificial intelligence training and inference, they also introduce system level resilience and communication latency assurance challenges which can be fundamentally important to resolve in the CPSs that have real-time and safety-criticality requirements. In practise, the more safety critical a system is, the closer to the system the inference and decisions should be made. Finally, in relation to the research questions posed in Section 1, artificial intelligence and formal verification combined

can be of great help in the development of time-critical and secure CPSs. Nevertheless, there is a need for future work to resolve the above mentioned challenges in utilising the combination of artificial intelligence and formal verification techniques in time- and safety-critical CPSs.

5.4. Limitations

There are several limitations of the work presented in this paper. The first limitation is concerned with the definition of CPSs and the CPS eco-system presented in Figure 1. As discussed in Section 1, there exist several definitions of CPSs. Our definition conforms to the well-known definitions of CPSs by Lee and Seshia [1,2] and by the International Conference on CPSs (ICPPS). The second limitation is concerned with the magnitude of the gaps identified in the state of the art. We have not performed a systematic literature review [67,68]; instead, we have investigated the design-time support for specific properties of time-critical CPSs, including timing predictability, functional safety and security. Another limitation is the consideration of timing predictability verification only at the design-time and not at the run-time. Finally, we have answered the posed research questions after investigating the state-of-the-art solutions. However, we have not thoroughly investigated the commercial/proprietary solutions and tools, especially those that do not expose the underlying models, techniques and methods.

6. Conclusions and Future Work

Cyber-physical systems require a tight combination of and coordination between computational and physical processes. These systems resulted in the recent years from the confluence of technologies in embedded systems, distributed systems, dependable systems and often real-time systems with advances in networking, microcontrollers, sensors, actuators and even artificial intelligence. Time- and safety-critical CPSs must operate safely, securely, efficiently and in real-time, and therefore predictability with regards to timing and security requirements is critical for their development. To identify the key issues involved in the development of these systems, in this position paper, we first draw a parallel between embedded systems and CPSs. We then explore the research and look into a number of existing frameworks and techniques devoted to developing timing predictable and secure safety-critical embedded systems. We conclude that the state of the art in the embedded systems domain is inadequate for the more complex and open industrial CPSs, as the boundaries of the CPS extend beyond the system's network interfaces. Furthermore, the existing techniques and frameworks for the verification of timing predictability in embedded systems are based on the assumptions about the system and its environment. Shifting from embedded systems to CPSs, these assumptions do not hold anymore, as even if a time-critical CPS is proven to be timing predictable at the design time, the predictability of the system can be jeopardized when the system is executed at the run-time due to security threats on the data entering the system via its sensors, networks, other CPSs or even the cloud. We make a strong case to advocate that timing predictability should be considered a prerequisite for functional safety, while security should be considered a prerequisite for both timing predictability and functional safety. We also conclude that the existing techniques and frameworks for building timing predictable CPSs hold only if the data integrity, authentication and authorization are supported. Therefore, we argue that there is a gap and future work should be devoted to developing frameworks that render security as a prerequisite for time- and safety-critical CPSs. In addition to all the above, we highlight that CPSs can leverage formal verification to guarantee safe system operation at design time, as well as use artificial intelligence to act well in open environments. We illustrate our ideas and position on one such system, namely the autonomous quarry.

Author Contributions: Conceptualization, S.M., E.L. and A.V.F.; methodology, S.M.; investigation, S.M., E.L. and A.V.F.; writing—original draft preparation, S.M., E.L. and A.V.F.; writing—review and editing, S.M. and E.L. All authors have read and agreed to the published version of the manuscript.

Funding: The work in this paper was supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) through the project DESTINE, the Swedish Foundation for Strategic Research (SSF) through the project Serendipity, and the Swedish Knowledge Foundation (KKS) through the projects HERO and DPAC.

Acknowledgments: The authors would like to thank the industrial partners Volvo CE, Ericsson, and Arcticus Systems, among others. Finally, thanks to the anonymous reviewers for their valuable input.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
CPS	Cyber-physical System
ES	Embedded System
ICPS	Industrial Cyber-physical System
ML	Machine Learning
RL	Reinforcement Learning
RSU	Road-side Unit
RTOS	Real-time Operating System
AUTOSAR	AUTomotive Open System ARchitecture
RCM	Rubus Component Model
TADL	Timing Augmented Description Language
ADL	Architecture Description Language
ISO	International Organization for Standardization

References

- Lee, E.A.; Seshia, S.A. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*; MIT Press: Cambridge, MA, USA, 2016.
- Lee, E.A. Cyber Physical Systems: Design Challenges. In Proceedings of the 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), Orlando, FL, USA, 5–7 May 2008; pp. 363–369.
- Li, Q.; Yao, C. *Real-Time Concepts for Embedded Systems*, 1st ed.; CRC Press, Inc.: Boca Raton, FL, USA, 2003.
- Barr, M. Embedded Systems Glossary. Available online: <http://www.netrino.com/Embedded-Systems/Glossary> (accessed on 20 February 2020).
- Barr, M.; Massa, A. *Programming Embedded Systems*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2006.
- Mubeen, S.; Lisova, E.; Feljan, A.V. A Perspective on Ensuring Predictability in Time-critical and Secure Cooperative Cyber Physical Systems. In Proceedings of the IEEE International Conference on Industrial Technology (ICIT), Melbourne, Australia, 13–15 February 2019; pp. 1379–1384.
- Stankovic, J.A.; Ramamritham, K. What is predictability for real-time systems? *Real-Time Syst.* **1990**, *2*, 247–254. [[CrossRef](#)]
- Thiele, L.; Wilhelm, R. Design for Timing Predictability. *Real-Time Syst.* **2004**, *28*, 157–177. [[CrossRef](#)]
- Kirner, R.; Puschner, P. Time-Predictable Computing. In *Software Technologies for Embedded and Ubiquitous Systems*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 23–34.
- Grund, D.; Reineke, J.; Wilhelm, R. A Template for Predictability Definitions with Supporting Evidence. In Proceedings of the Bringing Theory to Practice: Predictability and Performance in Embedded Systems, Grenoble, France, 18 March 2011; Open Access Series in Informatics; Volume 18, pp. 22–31.
- Ravi, S.; Raghunathan, A.; Kocher, P.; Hattangady, S. Security in Embedded Systems: Design Challenges. *ACM Trans. Embed. Comput. Syst.* **2004**, *3*, 461–491. [[CrossRef](#)]
- Serpanos, D.N.; Voyiatzis, A.G. Security Challenges in Embedded Systems. *ACM Trans. Embed. Comput. Syst.* **2013**, *12*, 66:1–66:10. [[CrossRef](#)]
- Jurjens, J. Developing Secure Embedded Systems: Pitfalls and How to Avoid Them. In Proceedings of the Companion to the Proceedings of the 29th International Conference on Software Engineering (ICSE COMPANION'07), Minneapolis, MN, USA, 20–26 May 2007; pp. 182–183.
- Kissel, R. *Glossary of Key Information Security Terms*; U.S. Department of Commerce, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2006.

15. Lisova, E.; Uhlemann, E.; Åkerberg, J.; Björkman, M. Towards secure wireless TTEthernet for industrial process automation applications. In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), Barcelona, Spain, 16–19 September 2014; pp. 1–4.
16. Doyle, M.G. How Volvo CE Is Engineering a Quarry Run by Electric Loaders and Haulers for Big Cuts to Costs and Emissions. 2016. Available online: <https://www.equipmentworld.com> (accessed on 15 September 2018).
17. Šurković, A.; Hanić, D.; Lisova, E.; Čaušević, A.; Lundqvist, K.; Wenslandt, D.; Falk, C. Incorporating Attacks Modeling into Safety Process. In *Computer Safety, Reliability, and Security*; Gallina, B., Skavhaug, A., Schoitsch, E., Bitsch, F., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 31–41.
18. Sha, L.; Abdelzaher, T.; Rzén, K.E.A.; Cervin, A.; Baker, T.P.; Burns, A.; Buttazzo, G.; Caccamo, M.; Lehoczky, J.P.; Mok, A.K. Real Time Scheduling Theory: A Historical Perspective. *Real-Time Syst.* **2004**, *28*, 101–155. [[CrossRef](#)]
19. Lo Bello, L.; Mariani, R.; Mubeen, S.; Saponara, S. Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems. *IEEE Trans. Ind. Inf.* **2019**, *15*, 1038–1051. [[CrossRef](#)]
20. Timing Augmented Description Language (TADL2) Syntax, Semantics, Metamodel Ver. 2, Deliverable 11 August 2012. Available online: <https://itea3.org/project/timmo-2-use.html> (accessed on 20 February 2020).
21. EAST-ADL Domain Model Specification, V2.1.12. Available online: http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf (accessed on 20 February 2020).
22. Mubeen, S.; Nolte, T.; Sjödin, M.; Lundbäck, J.; Lundbäck, K.L. Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints. *Softw. Syst. Model.* **2017**, *18*, 39–69. [[CrossRef](#)]
23. AMALTHEA4public Project. 2020. Available online: <http://www.amalthea-project.org> (accessed on 20 February 2020).
24. AUTOSAR Technical Overview, Release 4.1, Rev. 2, Ver. 1.1.0. The AUTOSAR Consortium, October 2013. Available online: <http://autosar.org> (accessed on 20 February 2020).
25. Feiertag, N.; Richter, K.; Nordlander, J.; Jonsson, J. *A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics*; CRTS Workshop; IEEE Communications Society: Piscataway, NJ, USA, 2008.
26. Mubeen, S.; Mäki-Turja, J.; Sjödin, M. Support for End-to-End Response-Time and Delay Analysis in the Industrial Tool Suite: Issues, Experiences and a Case Study. *Comput. Sci. Inf. Syst.* **2013**, *10*, 453–482. [[CrossRef](#)]
27. Sun, B.; Li, X.; Wan, B.; Wang, C.; Zhou, X.; Chen, X. Definitions of predictability for Cyber Physical Systems. *J. Syst. Archit.* **2016**, *63*, 48–60. doi:10.1016/j.sysarc.2016.01.007. [[CrossRef](#)]
28. Mubeen, S.; Nikolaidis, P.; Didic, A.; Pei-Breivold, H.; Sandström, K.; Behnam, M. Delay Mitigation in Offloaded Cloud Controllers in Industrial IoT. *IEEE Access* **2017**, *5*, 4418–4430. [[CrossRef](#)]
29. Rohmad, M.S.; Hashim, H.; Saparon, A. Lightweight cryptography on programmable system on chip: Standalone software implementation. In Proceedings of the 2015 IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE), Langkawi, Malaysia, 12–14 April 2015; pp. 151–154.
30. Bansod, G.; Raval, N.; Pisharoty, N. Implementation of a New Lightweight Encryption Design for Embedded Security. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 142–151. [[CrossRef](#)]
31. Robert Bosch GmbH, CAN with Flexible Data-Rate (CAN FD), White Paper, Ver. 1.1. 2012. Available online: https://www.can-cia.org/fileadmin/resources/documents/proceedings/2012_hartwich.pdf (accessed on 20 February 2020).
32. Hiroaki, I.; Edahiro, M.; Sakai, J. Towards scalable and secure execution platform for embedded systems. In Proceedings of the 2007 Asia and South Pacific Design Automation Conference, Yokohama, Japan, 23–26 January 2007; pp. 350–354.
33. Konrad, S.; Cheng, B.H.C. Real-time Specification Patterns. In Proceedings of the 27th International Conference on Software Engineering, ICSE '05, St. Louis, MO, USA, 15–21 May 2005; ACM: New York, NY, USA, 2005; pp. 372–381.
34. Kornecki, A.; Zalewski, J. Safety assurance for safety-critical embedded systems: Qualification of tools for complex electronic hardware. In Proceedings of the 1st International Conference on Information Technology, Gdansk, Poland, 18–21 May 2008.

35. International Organization for Standardization (ISO). *ISO 26262: Road Vehicles—Functional Safety*; ISO: Geneva, Switzerland, 2011.
36. Weinstock, C.B.; Lipson, H.F.; Goodenough, J. Arguing Security—Creating Security Assurance Cases. Software Engineering Institute, Carnegie Mellon University, USA, 2014. Available online: https://resources.sei.cmu.edu/asset_files/WhitePaper/2013_019_001_293637.pdf (accessed on 20 February 2020).
37. Johnson, P.; Gorton, D.; Lagerström, R.; Ekstedt, M. Time between vulnerability disclosures: A measure of software product vulnerability. *Comput. Secur.* **2016**, *62*, 278–295. [[CrossRef](#)]
38. Melchers, R. On the ALARP Approach to Risk Management. *Reliab. Eng. Syst. Saf.* **2001**, *71*, 201–208. [[CrossRef](#)]
39. Girs, S.; Šljivo, I.; Jaradat, O. Contract-Based Assurance for Wireless Cooperative Functions of Vehicular Systems. In Proceedings of the 43rd Annual Conference of the IEEE Industrial Electronics Society (IECON), Beijing, China, 29 October–1 November 2017.
40. Čaušević, A. A Risk and Threat Assessment Approaches Overview in Autonomous Systems of Systems. In Proceedings of the the 26th IEEE International Conference on Information, Communication and Automation Technologies, Sarajevo, Bosnia-Herzegovina, 26–28 October 2017.
41. Hänninen, K.; Hansson, H.; Thane, H.; Saadatmand, M. Inadequate Risk Analysis Might Jeopardize the Functional Safety of Modern Systems. *CoRR* **2018**, abs/1808.10308. Available online: <http://xxx.lanl.gov/abs/1808.10308> (accessed on 20 February 2020).
42. Lisova, E.; Šljivo, I.; Čaušević, A. Safety and Security Co-Analyses: A Systematic Literature Review. *IEEE Syst. J.* **2019**, *13*, 2189–2200. [[CrossRef](#)]
43. Leveson, N.G. *Safeware: System Safety and Computers*; ACM: New York, NY, USA, 1995.
44. SAE J3061. *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*; SAE International: Warrendale, PA, USA, 2016.
45. Johnson, N.; Kelly, T. An Assurance Framework for Independent Co-assurance of Safety and Security. *CoRR* **2019**, abs/1903.01220. Available online: <https://arxiv.org/pdf/1903.01220.pdf> (accessed on 20 February 2020).
46. Mubeen, S.; Lawson, H.; Lundbäck, J.; Gålnander, M.; Lundbäck, K. Provisioning of Predictable Embedded Software in the Vehicle Industry: The Rubus Approach. In Proceedings of the 4th IEEE/ACM International Workshop on Software Engineering Research and Industrial Practice, Buenos Aires, Argentina, 21 May 2017.
47. Paulitsch, M.; Ruess, H.; Sorea, M. Non-functional Avionics Requirements. In *Leveraging Applications of Formal Methods, Verification and Validation*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 369–384.
48. Löfwenmark, A. Timing Predictability in Future Multi-Core Avionics Systems. Licentiate’s Thesis, Department of Computer Science and Information Systems, Linköping University, Linköping, Sweden, 2017.
49. Feld, T.; Biondi, A.; Davis, R.I.; Buttazzo, G.; Slomka, F. A survey of schedulability analysis techniques for rate-dependent tasks. *J. Syst. Softw.* **2018**, *138*, 100–107. [[CrossRef](#)]
50. Mubeen, S.; Mäki-Turja, J.; Sjödin, M. MPS-CAN Analyzer: Integrated Implementation of Response-Time Analyses for Controller Area Network. *J. Syst. Archit.* **2014**, *60*, 828–841. [[CrossRef](#)]
51. Becker, M.; Mubeen, S.; Dasari, D.; Behnam, M.; Nolte, T. A generic framework facilitating early analysis of data propagation delays in multi-rate systems (Invited paper). In Proceedings of the 2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hsinchu, Taiwan, 16–18 August 2017.
52. Becker, M.; Dasari, D.; Mubeen, S.; Behnam, M.; Nolte, T. End-to-end timing analysis of cause-effect chains in automotive embedded systems. *J. Syst. Archit.* **2017**, *80*, 104–113. [[CrossRef](#)]
53. International Electrotechnical Commission. *IEC 60812: Analysis Techniques for System Reliability-Procedure for Failure Mode and Effects Analysis (FMEA)*; International Electrotechnical Commission: Geneva, Switzerland, 2006.
54. North Atlantic Treaty Organization. *Engineering for System Assurance NATO in Programmes*; NATO: Washington, DC, USA, 2010. Available online: <https://standards.globalspec.com/std/1236626/nato-aep-67> (accessed on 20 February 2020).
55. Weaver, R.; Fenn, J.; Kelly, T. A Pragmatic Approach to Reasoning about the Assurance of Safety Arguments. In Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software, Canberra, Australia, 9–10 October 2003.

56. Šljivo, I.; Lisova, E.; Afshar, S. Agent-centred Approach for Assuring Ethics in Dependable Service Systems. In Proceedings of the 13th IEEE World Congress on Services, Honolulu, HI, USA, 25–30 June 2017.
57. Haley, C.; Laney, R.; Moffett, J.; Nuseibeh, B. Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Trans. Softw. Eng.* **2008**, *34*, 133–153. [[CrossRef](#)]
58. McDermott, J.; Fox, C. Using abuse case models for security requirements analysis. In Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99), Phoenix, AZ, USA, 6–10 December 1999; pp. 55–64.
59. Raspotnig, C.; Karpati, P.; Katta, V. A Combined Process for Elicitation and Analysis of Safety and Security Requirements. In *Enterprise, Business-Process and Information Systems Modeling*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 347–361.
60. Gu, T.; Lu, M.; Li, L. Extracting interdependent requirements and resolving conflicted requirements of safety and security for industrial control systems. In Proceedings of the 1st International Conference on Reliability Systems Engineering (ICRSE), Beijing, China, 21–23 October 2015; pp. 1–8.
61. Howard, G.; Butler, M.; Colley, J.; Sassone, V. Formal Analysis of Safety and Security Requirements of Critical Systems Supported by an Extended STPA Methodology. In Proceedings of the IEEE European Symposium on Security and Privacy Workshops (EuroSPW), Paris, France, 26–28 April 2017; pp. 174–180. doi:10.1109/EuroSPW.2017.68. [[CrossRef](#)]
62. Iliasov, A.; Romanovsky, A.; Laibinis, L.; Troubitsyna, E.; Latvala, T. Augmenting Event-B modelling with real-time verification. In Proceedings of the 1st International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA), Zurich, Switzerland, 2 June 2012; pp. 51–57.
63. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
64. Johnson, L.A. DO-178B, Software Considerations in Airborne Systems and Equipment Certification. *Crosstalk October 1998*, 199. Available online: <http://www.dcs.gla.ac.uk/johnson/teaching/safety/reports/schad.html> (accessed on 20 February 2020).
65. *ISO 26262: Road Vehicles-Functional Safety, International Standard ISO/FDIS*; ISO: Geneva, Switzerland, 2011.
66. Garcia, J.; Fernández, F. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **2015**, *16*, 1437–1480.
67. Kitchenham, B.; Brereton, P. A systematic review of systematic review process research in software engineering. *Inf. Softw. Technol.* **2013**, *55*, 2049–2075. [[CrossRef](#)]
68. Zhang, H.; Babar, M.A. Systematic reviews in software engineering: An empirical investigation. *Inf. Softw. Technol.* **2013**, *55*, 1341–1354. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).