# Timing Model for Predictive Simulation of Safety-Critical Systems

Emilia Cioroaica[1][a], José Miguel Blanco[2][b] and Bruno Rossi[2][c]
[1]*Fraunhofer IESE, Kaiserslautern, Fraunhofer-Platz 1, Kaiserslautern, germany*
[2]*Masaryk University, Brno, Czech Republic*
*emilia.cioroaica@iese.fraunhofer.de, {brossi, jmblanco}@mail.muni.cz*

Keywords: Trust, Malicious Behavior, Virtual Evaluation, Runtime Prediction, Predictive Simulation, Automotive

Abstract: Emerging evidence shows that safety-critical systems are evolving towards operating in uncertain context while integrating intelligent software that evolves over time as well. Such behavior is considered to be unknown at every moment in time because when faced with a similar situation, these systems are expected to display an improved behavior based on artificial learning. Yet, a correct learning and knowledge-building process for the non-deterministic nature of an intelligent evolution is still not guaranteed and consequently safety of these systems cannot be assured. In this context, the approach of predictive simulation enables runtime predictive evaluation of a system behavior and provision of quantified evidence of trust that enables a system to react safety in case malicious deviations, in a timely manner.

For enabling the evaluation of timing behavior in a predictive simulation setting, in this paper we introduce a general timing model that enables the virtual execution of a system's timing behavior. The predictive evaluation of the timing behavior can be used to evaluate a system's synchronization capabilities and in case of delays, trigger a safe fail-over behavior. We iterate our concept over an use case from the automotive domain by considering two safety critical situations.

## 1 INTRODUCTION

With the introduction of advanced automated driving functions, software platforms of future automotive systems will become capable to support the execution of a multitude of applications. For enabling this transition, AUTOSAR (Autosar, 2021), through an active engagement in standardization activities for in-vehicle software concluded that architectures of vehicles need to become more flexible, highly available and capable to adapt over time to specific application requirements. As a result, an updated version of the AUTOSAR automotive platform has emerged, which is based on POSIX standard (Vector, 2022). This new version aims at supporting dynamic deployment of applications and connection of deeply embedded and non-AUTOSAR systems for preserving real-time capabilities.

But the trend of extending vehicles' capabilities will further on continue into connecting them to almost everything: smart homes, roadside infrastructure and other vehicles as well. Given the safety-critical nature of a vehicle, the emerging interconnection of systems needs to be regarded as safety-critical SoS (System of Systems) as well. Besides their safety-critical nature, emerging evidence show that these systems are engineered to sustain business growth through continuous runtime and design time co-engineering approaches. Initially delivered with a quality right above the minimum required threshold of acceptance, systems' upgrades are then delivered during the system operation time (techcrunch.com, 2021). This philosophy, that has recently transitioned from the domain of information systems into the safety-critical domain where systems operate in dynamic environments is raising considerable safety concerns (Stilgoe and Cummings, 2020).

In the course of system development, and safety-critical systems in particular, the emerging development paradigm integrates human-engineered activities, resulting in security considerations within a larger context of digital ecosystems (Bosch, 2015). The interplay of various actors (such as: users, organisations, developers part of an organisation) with a variety of goals ranging from purely cooperative and collaborative to competitive and even malicious in the realization of system's function raises crucial issues.

[a] https://orcid.org/0000-0003-2776-4521
[b] https://orcid.org/000-0001-9460-8540
[c] https://orcid.org/0000-0002-8659-1520

Malicious behaviour can be embedded within a software application deployed on the system in dynamic ecosystems where software applications, hardware resources, and platform components of cyber-physical systems are provided by various actors. For example, a systematic insertion of intended faults, such as logic bombs (Avižienis et al., 2004) can be strategically introduced in a system, remain dormant and get activated in a synchronized manner at the "right moment" to support a planned attack.

For enabling the ultimate safe operation of safety-critical systems open to accommodate or activate system functions at runtime, in the previous work we've introduced the concept of predictive simulation that targets the evaluation of the trustworthiness of software applications in dynamic environments by accounting of possible malicious behavior. In a predictive virtual environment, during runtime, Digital Twins (DTs) of systems and system components (including software components) are executed much faster than the wall clock in order to enable the gathering of the trusted behavior signatures of safe execution against which the execution of the software application in the real world execution is then checked for conformity (Cioroaica et al., 2019). Given the nature of fast behavioral execution, the result of the prediction can then be used for enabling system's safe reconfiguration through activation of an operational fail-over behavior that keeps the system in a safe state. In this paper we further specify the definition of a generic temporal model that can be used for the fast evaluation of a system timing behavior. When fed with real time data, this model becomes a specialized digital twin, that we referred to as Temporal Digital Twin (TDT).

In what follows, Section 2 provides an overview of the emerging trends in designing safety of vehicles within complex ecosystems. Section 3 introduces our approach in the general context of predictive simulation, Section 4 details our use case analysis based on an open data set that describe a platooning scenario and motivated by current trends. Section 5 presents our proof of concept emerged as iteration of the temporal logic model over the use case and Section 6 presents the conclusions, on-going and future work.

## 2 EMERGING TRENDS

### 2.1 Enabling Safety through Ecosystem Design

In the quest of enabling safety, transportation in the domains of avionics and railway, vehicles have been engineered without the capability to decide independently which path to take. In avionics, transportation works in coordination with an air traffic controller, in the railway domain, it works under the coordination of a railway control center. The central coordination ensures that no other vehicle is present on a specified path and therefore, collisions can easily be avoided. In the automotive domain on the other hand, a much higher number of vehicles are moving independently by accounting of the driver's autonomous decisions. And because the number of fatalities is caused by human flawed attention, achieving safety is envisioned through an increase of the level of automated driving functions. In SAE automation level 5, the driver will no longer be present (Levels, 2022).

With emerging availability of high-performance data transmission technologies such as 5G, the possibility of coordinating automated vehicles centrally is becoming an option for enabling an even higher level of safety (Schmeitz et al., 2019). For example, when automated vehicles are largely deployed for operating in specific ODDs (Operational Design Domain) like motorways where there is no road infrastructure that connects VRUs (Vehicle Road Units) for supporting transportation, an autonomous behavior that rely on a minimum of local sensors and autonomous decisions must be in place in order to enable safe driving synchronization with conventional vehicles and to avoid obstacles on the road. A central road supervisor can supervise a part of a motorway's traffic by coordinating driving decisions such as lane changes. This design enables early computation of decisions about correct lanes, followed by proactive setting of automatic speed limits. Consequently congestion can be avoided, the levels of safety increase and because of a smooth driving pollution can be reduced.

### 2.2 The danger of Unknowns

Ecosystem-enabled provision of safety-critical systems that integrate and/or activate software smart agent during runtime is endangered by possible malicious attacks. A major challenge in detecting malicious deviations arises from the non-deterministic nature of an intelligent software component. An ongoing adaptation of an intelligent behavior can either indicate a successful learning or a malicious deviation because at the operational level, a software component is allowed to provide different outputs for the same set of inputs. For intelligent behavior developed within digital ecosystems and deployed on safety-critical systems, it is very likely that sooner or later, intended faults will be injected into a system together with an update (Miller and Valasek, 2014).

While there is currently much focus on preventing the injection of malicious behavior within an ecosystem, less emphasis lies on detection and mitigation of its negative effects.

## 2.3 Runtime Safe Reaction to Security Intrusions

Assuring safety of these systems during their runtime operation is achieved through engineering of self-adaptive systems capable to respond to changes in their internal dynamics. Within these systems, a self-contained structure enables behavior reconfiguration during system operation (Krupitzer et al., 2015; Weyns et al., 2012; Srivastava and Mondal, 2015) by employing a range of dynamic risk management techniques that work with runtime evidence of trusted behavior (Khan et al., 2016; Leite et al., 2018). In this context, the execution of software smart agents that integrate intelligent behavior is characterized by uncertainty as a deviation from the norm is not always an indicative of bad behavior, it can be an intelligent adaptation as well (Bry and Roy, 2011). The distinction between the two is a crucial factor gaining increased attention within the safety-critical domain. In particular the danger of operating unknown behavior in uncertain context is gaining increasing attention (Hamon et al., 2020).

## 2.4 The need for Prediction

When system behavior is expected to change during runtime, evaluation of its trustworthiness needs to be performed during runtime as well. In our opinion, this can be achieved by employing runtime prediction mechanisms. Current practices for prediction in the industrial domain is focused on failure prediction for enabling predictive maintenance (Lei et al., 2018; Carvalho et al., 2019). We see a great potential in predictive methods integrated into runtime approaches that requires a transition from traditional predictive techniques that rely on expert engineering assessment and taking advantage of the fast computing resources. In this context, employing a traditional intelligent behavior (AI) for predicting another behavior would lack transparency making it difficult to argument from a safety standpoint. This happens because continuously learning techniques can provide no guarantee of trust, whereas runtime safety requires provision of evidence. In our opinion, assuring safety of critical systems necessitates runtime prediction mechanisms that can provide the required evidence in a deterministic manner by leveraging the traditional simulation mechanism in a runtime setting.

The predictive simulation method as introduced in (Cioroaica et al., 2019) and described within a complex auditing process in (Calabrò et al., 2022), accounts of the possibility of receiving a possibly untrusted software smart agent directly at runtime. Within a vehicle, such a software smart agent, can, for example, keep the maximum and minimum distance within platoons. Aiming at evolving the level of function automation within a vehicle, the software smart agent requires a runtime evaluation of its trusted behavior. Particularly challenging for the runtime virtual evaluation are two aspects: (a) the open nature of the platform capable to accommodate other interacting software agents as well, aspect which complicates the level of technical trust assessment of a software agent under evaluation and (b) the real time nature of the systems. A safety critical system needs to react in real time in order to avoid hazardous situations caused by a miss-behaving software smart agent and/or interconnected other software smart agents.

## 3 METHODOLOGY

In this section we describe the general methodology of predictive simulation that accounts for execution of different behavioral models, emphasising the timing aspects.

## 3.1 General Methodology

The approach that enables runtime detection of malicious deviations based on predictive simulation requires a design phase for engineering systems artefacts that support the later runtime prediction and conformity monitoring. In this phase, different models of the system behavior are created, including functional models that enable runtime evaluation of functional interaction, temporal models that enable timing predictions used in evaluating a software smart agent's synchronization capabilities and models that enable the runtime evaluation of the communication protocol. In the current paper we focus on the evaluation of timing aspects.

As depicted in Figure 1, for enabling the runtime prediction of timing behavior, in the pure predictive simulation phase, the temporal logic model is used to validate the accuracy of the digital twins that provide the timing abstractions. The development of the two artifacts: the temporal model and the software smart agent, can lead to a set of situations, namely:

1. No faults in either of the artifacts: Ideal situation
2. Same fault in both artefacts: Fair prediction

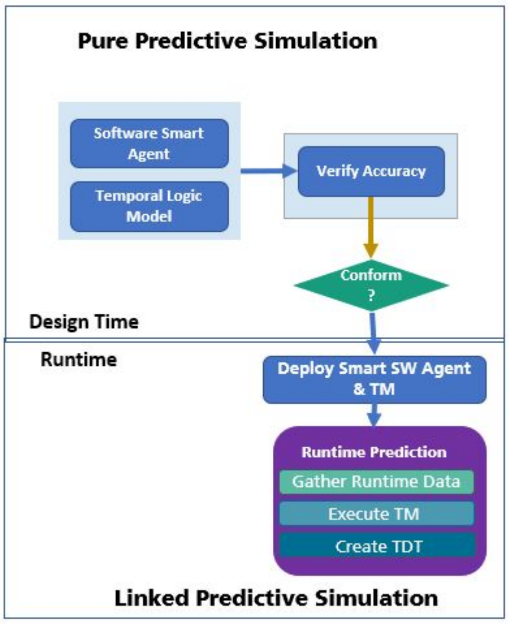

Figure 1: General Method

3. Different faults between artifacts: situation that leads to dishonest trust

For addressing these possible deviations, during pure predictive simulation, the behavior of a software smart agent, subject to trust evaluation and the corresponding temporal logic model are evaluated for consistency before being deployed.

Then, during runtime within within a simulation environment, before the execution of the software smart agent, the corresponding temporal model is fed with real time data and it is executed as a much faster speed. During this phase, the specialized Temporal Digital Twin (TDT) is executed in relation to other TDT of interacting components of teh software smart agent, that can be either other sofwtare component, hardware resources or system platforms.

## 3.2 Model for the timing behavior

In this subsection we introduce the general model for the timing behavior that enables the evaluation of synchronization's aspects of a software smart agent in accordance to the principles of predictive simulation stated above. The model we provide is specifically targeted towards capturing untrusted deviations from the minimum and maximum delay of execution. For the experienced reader, it is obvious that the model is based on a fragment of Linear Temporal Logic (LTL) (Pnueli, 1977), but with a refined validation clause regarding the temporal connective included similar to the one of (Blanco et al., 2021).

LTL models traditionally provide means for formally checking events' occurrence over time captured in traditional connectives of *until* and *since*, our models differ from general LTL models in the sense that the temporal connective is limited in its future scope and the validity of statements is to be contained in said scope. Besides this restriction for the temporal connective, our model provides customized predictive-simulation restrictions that check for future events as well. Overall, the model we are providing is highly expressive for timing considerations of behavior evaluated within the predictive simulation paradigm.

With all of the above we begin by defining simple and complex statements. For any simple statements $p$, $q$,..., any complex statements $A$, $B$,..., the unary connectives $\neg$ (Negation), $\Diamond$ (In the future), and the binary connectives $\wedge$ (Conjunction), $\vee$ (Disjunction), $\rightarrow$ (Entailment), the following recursive forming rules apply:

- (a) For any simple statement $p$, $p$ is a well-formed statement. Furthermore, if $A = p$, then $A$ is well-formed statement.
- (b) If $A$ is a well-formed statement and $*$ is a unary connective, then $*A$ is a well-formed statement.
- (c) If $A$ and $B$ are well-formed statements and $*$ a binary connective, then $A * B$ is a well-formed statement.
- (d) There are no more well-formed statements than those defined by the clauses (a), (b) and (c).

By simple and complex statements we are referring to any kind of data generated by events. Let it be noted that, while the we are defining a nice array of connectives, we have excluded any quantifiers connectives (e. g., $\forall x$, for all $x$) and focus on the propositional fragment rather than the first or higher order ones. This helps to keep the forthcoming model to a minimum, therefore making its implementation easy as only simple operations would be required. It also allows to reduce the computational complexity and make its implementation in resource-constrained devices much easier.

Now we introduce the model. A model $M$ is the structure $M = \langle K, T, \models \rangle$, where $K$ is the set of *vehicles* $a$, $b$, $c$,...; i. e., $K = \{a, b, c, ..\}$; each element of $K$, each *vehicle*, is a set in itself that includes a minimum and maximum time delay, $m$ and $h$ respectively, among other optional characteristics $o_1$, $o_2$, $o_3$,...; i. e., $a = \{m, h, o_1, o_2, o_3, ...\}$. $T$ is a set of temporal points $t_1, t_2, t_3, ...$; i. e., $T = \{t_1, t_2, t_3, ...\}$. Finally, $\models$ is a relation from $K$ to the set of statements such that the following clauses apply:

- (1) $a \models A \wedge B$ if and only if (iff) $a \models A$ and $a \models B$

- (2) $a \models A \vee B$ iff $a \models A$ or $a \models B$
- (3) $a \models \neg A$ iff $a \not\models A$
- (4) $a \models A \rightarrow B$ iff $a \models \neg A$ or $a \models B$
- (5) $a, t \models \Diamond A$ iff $h = t + d_1$, $m = t + d_2$ & $\exists s$, $s \in T$, with $t < s$, $m < s < h$, and $a, s \models A$, and $\forall u$, $u \in T$, if $t < u < s$, then $a, u \models A$

Since the model is based on a fragment of LTL, most of the results of LTL, such as soundness and completeness (Burgess, 1984; Xu, 1988), decidability (Bozzelli et al., 2006), or satisfiability complexity (Bauland et al., 2007) can be extended to the model by the means of a simple corollary.

## 4 USE CASE

In this section we describe the iteration of our model over an use case form the automotive domain. More concretely, with the definitions of minimum and maximum delays for future deviations, we have iterated our temporal model over an open data set. As it will be described next, our model captures the delay concerns for safety critical situations such as gap closing and gap opening.

To this end, we start with the short description of the use case and our evaluation of possible malicious failures in Subsection 4.1, continue with a background of safety-related aspects that drive the evolution of the use case in Section 5.

### 4.1 Motivation

Multiple solutions have been provided for enabling highly and fully automated vehicles. Emerging practical solutions have been developed within the EU AUTOPILOT project (Autopilot, 2022), where, with the support of IoT (Internet Of Things) infrastructure, vehicles can automate their driving. Aiming at benefiting from development of IoT (Internet of Things) that boosts the connection between various objects over any type of service of network, solutions for automated driving have emerged in the past years.

Benefiting from an IoT infrastructure, connected vehicles become moving "things" within complex ecosystems. In this way, the vision of automated driving is taking advantage of the IoT potential. Based on our evaluation of the use case, within such a complex digital ecosystem that aggregates a multitude of interconnected devices and services, various failing points caused by transmission of false information can be imagined.

Typically, a platoon consists of a lead vehicle that transmits the maneuvering commands and one or more following vehicles enabled with automatic steering. The safe automated following maneuvers depend on V2V communication. The lead vehicle sends acceleration messages to the following vehicle. Concretely, it is using ITS-G5 and ultra-wide band (UWB) for exchanging time sensitive information(Schmeitz et al., 2019). Because the Platoon service keeps providing speed and lane advice to the platoon members, if V2V communication is delayed for too long, the platoon service can default into providing individual speed messages. Assuming measurement errors from interconnected devices, the following vehicle which is in the automated driving mode can accelerate for keeping with the advice speed received from the platoon service. This would lead to a crash.

In the example where the platoon service takes the information of multiple components in the IoT for generating speed and lane advice, false information about the timing in traffic lights can, for example, lead to speed increase targeted towards avoiding waiting time at a red light followed by sudden stops when the light turns red earlier than expected. Other failures can be cause by wrong information provided by the TM (Traffic Manager) of the TLC (Traffic Light Controller) that cause platoons to drive on wrong lanes. However, a simplified evaluation of the functional behavior, even though it can discover the intention of transmitting wrong values, it does not completely evidentiate the intention of malicious behavior. A correct value sent with a certain delay can endanger the safety of systems. Therefore, for capturing hidden malicious behavior, the timing aspects is very important. and we therefore focus our proof on concept on safeguarding a platoon in case of timing delays.

### 4.2 Technical Landscape

After a first analysis of the data sets available at and the use case description from (Schmeitz et al., 2019) we have divided the first version of the system architecture that evidentiate possible targets of security attacks. According to Figure 2 a *Vehicle* is composed of multiple subsystems including *Sensor*s, *Communication Media*, *Receivers* and *Actuators*. The communication media considered in this use case is of two types: *CAN Bus* that transports information between all ECUs of the vehicles and a *DSRC Channel* (Dedicated Short Range Communication Channel) representing the wireless communication channel used for sending messages between vehicles. The *Sensors* considered are 2 types of *GPS* that provides position, velocity, and timing information and *Radar*. external DSRC module (DSRC-VU) connected by a CAN bus using a communication protocol based on SAE J1939 (SAE, 2022). The Middleware which is the *IoT Platform* is a *Communication Unit* which is a type of *Station*. Other stations from where information is being logged represents the *HW Unit* and the *Communication Stack*. In this context, a *Software Application* that communicates with another software application uses the *Station* for outputting *Communication Messages* that are collected by a *Logging Component*.

Any of this components can constitute a possible target of attack that manifests both at the component and at the system level. Our first iteration of the concept accounts for behavioral deviations that are visible at the system level in two safety-critical situations.

In this regard, if a delay is perceived during the gap closing, then another vehicle, which is not part of the platoon get positioned between the leader and the following vehicle leading to a crash between the new vehicle and the platoon follower. If , on the other hand, the messages describing the gap opening procedure cause by a vehicle cut are delayed, then, the follower vehicle will continue the usual maneuvers for closing the gap, situation which will lead to another crash. A vehicle outside the platoon can cut in the platoon in situations when, for example, it needs to retract from overtaking the platoon because of other vehicles approaching from the opposite side on the left lane.

## 4.3 Analysis of the Data Set

In this subsection we present the reasoning of the data set that has guided the initial iteration of the temporal model over the platooning use case. The data set is available at (Zenodo, 2022) and represents data about vehicle platoon formation with live traffic data.

We evaluated on vehicle based on the Positioning System Component from 7:20:00 AM to 11:33:20 PM. At 11:33:37 PM we have an event `ID1` for `PLATOONING` which in our case is an event type sent by the `Log_application_Id` number 12 as platoon log action triggered by the Vehicle. Then the Log application ID 5 is reporting the position of the steering wheel `"472,690180982929"` in the *DrivervehicleInteraction* table (Zenodo, 2022). After this, more information is sensed within the environment and the coordinates of the vehicle are logged.

Further on, we have observed that the data for the *Target* is the same as the *EnvironmentSensorRelative*. This can be explained by the fact that the target sent by the cloud is the same as the data that the follower vehicle is receiving through the in-systems sensors. By advancing in time, while the *Target* Information keeps the same, the *EnvironmentSensorRelative* data is slightly changing. This is due to the motion on the roads: sometimes the vehicles gets closer or distanced from the target point of meeting. During all this time, the log `application ID 1` is logging the speed.

From the whole data set, we have selected the data that is relevant to the scenario described in (Zenodo, 2022), namely the data between the time stamps: `1538576583120` (Wednesday, 3 October 2018 14:23:03.120) and `1538576583600` (3 October 2018 14:23:03.600 ).

## 5 PROOF OF CONCEPT

The iteration of the temporal logic model introduced in Subsection 3.2 has been guided by the analysis of the platoon behavior and deepened through a detailed analysis of the data set presented in 4.3

In a regular platoon formation there might be unexpected requests for gap-opening. This request would take the shape of data statements $A$, $B$, $C$... and would be associated to a vehicle $a$, $b$, $c$... that needs to validate (and perform) said statement. This kind of actions have a time signature linked to them $t$, $r$, $s$.... Whenever a gap-opening is requested the validity of the situation would depend on the time that the vehicle takes to perform it. There is a maximum time limit $h$ for the gap-opening to happen that is defined as the original time signature plus the maximum delay. Given this, the data statement for the gap-opening would be validated if the data statement holds on any time signature that is set before the maximum limit established.

With all of the above, given a data statement for a gap-opening $A$ and the completion of the gap opening data statement $B$ for the vehicle $a$ at a time signature $t$ will be valid, $a,t \models \Diamond A \vee B$, iff for each time signature $u$, such that $t \leq u \leq h$ then $a,u \models \Diamond A$ and independently of the value of $B$. Similarly, it will not be valid if there is any time signature $v$ such that $a,v \models \Diamond A$, meaning that the gap-opening failed at the time signature $v$ and also $B$ would not be valid. On the other hand, it would be valid if the *Gap-opening_request* is done, and therefore not valid with respect to the future, but the vehicle has validated $B$, the fact that the gap-opening has been completed.

Given the following substitutions:

- $A$ = Gap-opening_request
- $B$ = Gap-opening_completed
- $a$ = Follower_1

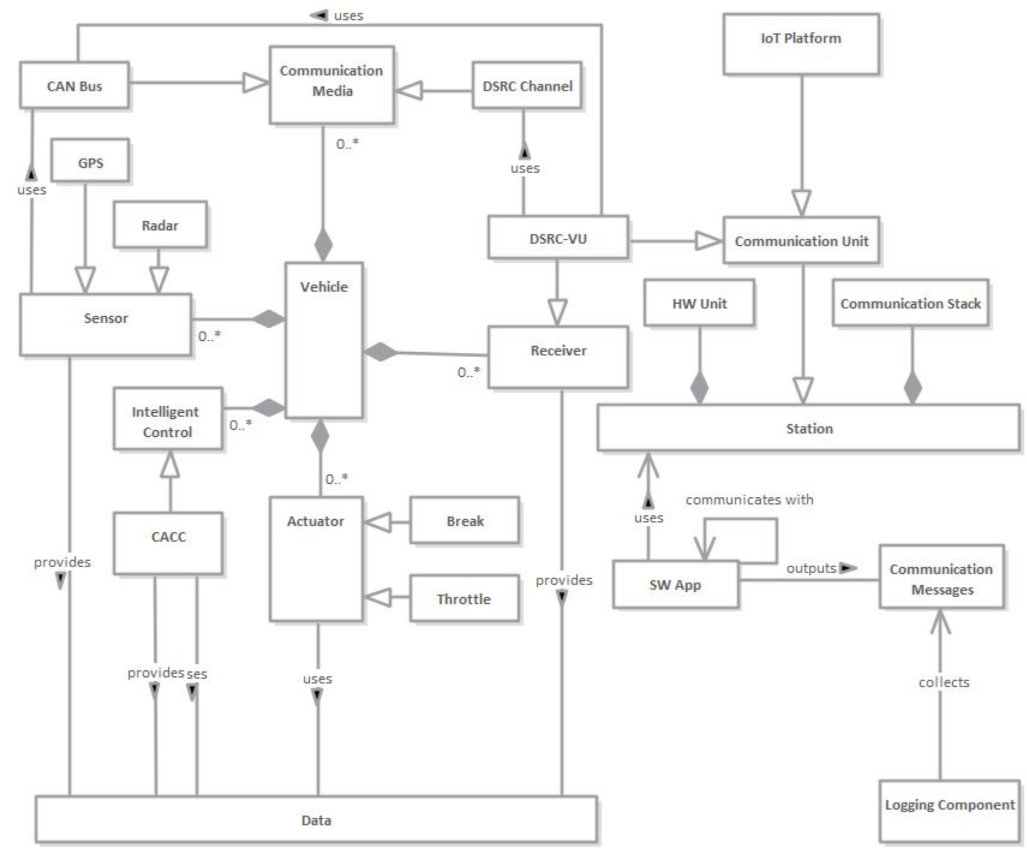

Figure 2: System Architecture

- $d$ = 00:00:15
- $d'$ = 00:00:02
- $t$ = 01:25:54
- $h$ = 01:26:11
- $m$ = 01:25:56
- $u$ = 01:26:03
- $v$ = 01:25:58

We can present the example of a gap opening as the validation of:

- $Follower\_1, 01:25:54 \models \Diamond(Gap-opening\_request) \vee Gap-opening\_completed$

This means that there would be no malicious behaviour as long as $Gap-opening\_request$ is validated towards the maximum time delay, or the gap opening has been completed, thanks to $Gap-opening\_completed$ being part of a Disjunction. In this case, for a supposed time signature $u$, the valid one, we would know that

- $Follower\_1, 01:26:03 \models Gap-opening\_request$

And either if

- $Follower\_1, 01:26:03 \models Gap-opening\_completed$

or

- $Follower\_1, 01:26:03 \not\models Gap-opening\_completed$

It would be valid given that for any time lesser than the time signature 01:26:11, the gap-opening request is valid, independently of the validity of the completion of the gap opening because, let us remember, a disjunction would be valid as long as one of its terms is. On the other hand, the non-valid request at time signature $v$ would look like this:
$Follower\_1, 01:25:58 \not\models Gap-opening\_request$
$Follower\_1, 01:25:58 \not\models Gap-opening\_completed$

It would not be valid because there's, at least, one time signature below the maximum, in this case 01:25:58, that does not validate the gap-opening request and it has not finished yet.

## 6 SUMMARY AND FUTURE WORK

In this paper we have presented the concept of predictive simulation used for enabling both: the runtime assessment of a system or system component trustworthiness and the needed self-reconfiguration in case of malicious/ untrusted deviations. For enabling the prediction of timing behavior that enables evaluation of synchronization capabilities, we have introduced a generic temporal model for the timing behavior that can be used for evaluating timing deviation. To this end, we have proposed a set of restrictive rules on expected behavior by analyzing a set of open data from an automotive use case. Our initial proof of concept has been performed by iterating the model over the behavior of the platoon in two safety-critical situation.

Ongoing work is directed towards reverse engineering the behavior of single systems that can be subject of the predictive simulation evaluation with respect to timing considerations. Future work will go into creation of models for enabling predictive evaluation of the function interaction between system components.

## ACKNOWLEDGEMENTS

This work has been partially funded by European Funds for Regional Development (EFRE) in context of "Investment in Growth and Employment" (IWB) P1-SZ2-3 F&E: Technologieorientierte Kompetenzenfelder -MWVLW "Neue Erprobungskonzepte fuer sichere Software in hochautomatisierten Nutzfahrzeugen" , by the European Union's Horizon 2020 research and innovation programme under grant agreement No 952702 (BIECO) and by ERDF/ESF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

## REFERENCES

Autopilot (2022). Autopilot EU Project. https://autopilot-project.eu/. [Online; accessed 03-April-2022].

Autosar (2021). AUTOSAR. https://www.autosar.org/. [Online; accessed 11-August-2021].

Avižienis, A., Laprie, J.-C., and Randell, B. (2004). Dependability and its threats: a taxonomy. In *Building the Information Society*, pages 91–120. Springer.

Bauland, M., Schneider, T., Schnoor, H., Schnoor, I., and Vollmer, H. (2007). The Complexity of Generalized Satisfiability for Linear Temporal Logic. In Seidl, H., editor, *Foundations of Software Science and Computational Structures*, Lecture Notes in Computer Science, pages 48–62, Berlin, Heidelberg. Springer.

Blanco, J. M., Rossi, B., and Pitner, T. (2021). A Time-Sensitive Model for Data Tampering Detection for the Advanced Metering Infrastructure. In *Annals of Computer Science and Information Systems*, volume 25, pages 511–519. ISSN: 2300-5963.

Bosch, J. (2015). Speed, data, and ecosystems: the future of software engineering. *IEEE Software*, 33(1):82–88.

Bozzelli, L., Křetínský, M., Řehák, V., and Strejček, J. (2006). On Decidability of LTL Model Checking for Process Rewrite Systems. In Arun-Kumar, S. and Garg, N., editors, *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science, pages 248–259, Berlin, Heidelberg. Springer.

Bry, A. and Roy, N. (2011). Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE international conference on robotics and automation*, pages 723–730. IEEE.

Burgess, J. P. (1984). Basic Tense Logic. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, Synthese Library, pages 89–133. Springer Netherlands, Dordrecht.

Calabrò, A., Cioroaica, E., Daoudagh, S., and Marchetti, E. (2022). Bieco runtime auditing framework. In Gude Prego, J. J., de la Puerta, J. G., García Bringas, P., Quintián, H., and Corchado, E., editors, *14th International Conference on Computational Intelligence in Security for Information Systems and 12th International Conference on European Transnational Educational (CISIS 2021 and ICEUTE 2021)*, pages 181–191, Cham. Springer International Publishing.

Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. d. P., Basto, J. P., and Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024.

Cioroaica, E., Kuhn, T., and Buhnova, B. (2019). (do not) trust in ecosystems. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pages 9–12. IEEE.

Hamon, R., Junklewitz, H., and Sanchez, I. (2020). Robustness and explainability of artificial intelligence. *Publications Office of the European Union*.

Khan, F., Hashemi, S. J., Paltrinieri, N., Amyotte, P., Cozzani, V., and Reniers, G. (2016). Dynamic risk management: a contemporary approach to process safety management. *Current opinion in chemical engineering*, 14:9–17.

Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., and Becker, C. (2015). A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17:184–206.

Lei, Y., Li, N., Guo, L., Li, N., Yan, T., and Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to rul prediction. *Mechanical systems and signal processing*, 104:799–834.

Leite, F. L., Schneider, D., and Adler, R. (2018). Dynamic risk management for cooperative autonomous medical cyber-physical systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 126–138. Springer.

Levels, S. (2022). SAE Levels of Driving Automation . https://www.sae.org/blog/sae-j3016-update. [Online; accessed 12-April-2022].

Miller, C. and Valasek, C. (2014). A survey of remote automotive attack surfaces. *black hat USA*, 2014:94.

Pnueli, A. (1977). The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. ISSN: 0272-5428.

SAE (2022). SAE J1939. https://www.sae.org/standardsdev/groundvehicle/j1939a.htm. [Online; accessed 14-April-2022].

Schmeitz, A., Schwartz, R., Ravesteijn, D., Verhaeg, G., Altgassen, D., and Wedemeijer, H. (2019). Eu autopilot project: Platooning use case in brainport.

Srivastava, N. K. and Mondal, S. (2015). Predictive maintenance using modified fmeca method. *International journal of productivity and quality management*, 16(3):267–280.

Stilgoe, J. and Cummings, M. (2020). Can driverless vehicles prove themselves safe? *Issues in Science and Technology*, 37(1):12–14.

techcrunch.com (2021). Tesla has activated its camera. https://techcrunch.com/2021/05/27/tesla-has-activated-its-in-car-camera-to-monitor-drivers-using-autopilot/.

Vector (2022). Posex-based AUTOSAR. https://cdn.vector.com/cms/content/know-how/_technical-articles/AUTOSAR/AUTOSAR_POSIX_Hanser_201902_PressArticle_EN.pdf. [Online; accessed 03-March-2022].

Weyns, D., Iftikhar, M. U., Malek, S., and Andersson, J. (2012). Claims and supporting evidence for self-adaptive systems: A literature study. In *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 89–98. IEEE.

Xu, M. (1988). On Some U, S-Tense Logics. *Journal of Philosophical Logic*, 17(2):181–202. Publisher: Springer.

Zenodo (2022). Autopilot Open Data Set. https://zenodo.org/record/3606616#.YXldMJ4zb-g. [Online; accessed 14-April-2022].