Predictive Runtime Simulation for building Trust in Cooperative Autonomous Systems

Emilia Cioroaica, Daniel Schneider, Hanna AlZughbi, Jan Reich, Rasmus Adler, Tobias Braun
Embedded Systems Quality Assurance
Fraunhofer IESE
Kaiserslautern, Germany
{Emilia.Cioroaica, Daniel.Schneider, Hanna.AlZughbi, Jan.Reich, Rasmus.Adler, Tobias.Braun}@iese.fraunhofer.de

Abstract—**Future autonomous systems will also be cooperative systems. They will interact with each other, with traffic infrastructure, with cloud services and with other systems. In such an open ecosystem trust is of fundamental importance, because cooperation between systems is key for many innovation applications and services. Without an adequate notion of trust, as well as means to maintain and use it, the full potential of autonomous systems thus cannot be unlocked. In this paper, we discuss what constitutes trust in autonomous cooperative systems and sketch out a corresponding multifaceted notion of trust. We then go on to discuss a predictive runtime simulation approach as a building block for trust and elaborate on means to secure this approach.**

Index Terms—**Automotive, Trust, Trust at Runtime, Safety**

I. Introduction

The future of transportation systems will be characterized by even higher levels of automation up to the point of autonomy. Future vehicles will have comprehensive perception capabilities and intelligence to reason about their perception and to decide upon adequate driving maneuvers. Presently, there is a tendency to mistrust any information or service outside the own vehicle and thus either not use it at all or only in a very constrained way. From a safety perspective this seems reasonable, but it cannot be a satisfactory solution for the future. This is because limiting (a) the perception scope to the own set of sensors, and (b) the possibilities to influence the context to the own actuators means limiting the performance ceiling as well as achievable levels of safety.

Thus, solutions are required to tackle the problem of trustworthiness in an open ecosystem, where systems and services of different provenance come together to render higher level services and applications in cooperation - which would otherwise be impossible to realize. If such comprehensive and flexible cooperation is actually enabled, the potential will be huge. New kinds of services, applications and business cases will arise, and both performance and safety could be significantly improved. On a highway, for example, vehicles driving closely together could benefit from reduced air drag and fuel consumption. Correspondingly, cooperative platooning features are envisioned to exploit these benefits. Like that, many other beneficial collaboration scenarios are presently discussed and developed that all help to improve performance, quality and even safety.

In previous work, we introduced runtime approaches to make systems aware with respect to their safety and corresponding context inter-dependencies [1], [2]. We put emphasis on modularity and the capability to integrate in the field and, most importantly, to determine relevant safety properties and reason about the safety of a dynamically formed cooperation. This general pattern and characteristics of our approach would, in our opinion, also be adequate for the problem of trust assurance stated here. However, our research has been strongly focused on safety, which is ultimately only one facet of trust - although an important one. Therefore, in this paper, we set out to widen the scope and to explore additional facets of trust as well as corresponding building blocks for our envisioned runtime trust approach. In particular, we will elaborate on utilizing predictive runtime simulation for building trust and discuss as to how such an approach could be secured.

Accordingly, in Section II, we discuss how trust is generally constituted and go on in Section III to elaborate on corresponding building blocks of trust, and, in particular, on our envisioned predictive simulation approach. In Section IV we present related work and in Section V we discuss the conclusions and future work.

II. Notion and Facets of Trust

Merriam-Webster defines ***trust*** (amongst other definitions less relevant for our case) as (1) assured reliance on the character, ability, strength, or truth of someone or something, and (2) one in which confidence is placed. Translating this into the world of cooperating systems, we define trust (in cooperative systems) as: **Assured reliance on the (promised or rightfully expected) functional and non-functional properties of a service/function of a cooperating system.**

In relation to assurance, *confidence* plays a very important role and some means of corresponding specification (maybe quantification) and related reasoning capability would clearly be beneficial. In the field of dependability, Avizienis et al. [3] have defined trust as *"accepted dependance"* and have provided ways for describing dependability. First, dependability of a system is defined as its ability to avoid service failure that are more frequent and more severe than acceptable. Further, dependability is described via the attributes *availability, reliability, safety, integrity, confidentiality* and *maintainability*.

Security is defined as the combination of *availability, integrity and confidentiality*.

From an engineering perspective, when deriving and implementing dependability requirements, the notion of dependability via service failures is more practical than the consideration of dependability attributes. The more severe the consequences of a service failure are, the lower is the acceptable likelihood of occurrence. The severity of consequences can be described via *risks*. If we consider risk as a potential path from the failure to a loss event (accident in the automotive domain) and measure it by combining the likelihood of the path happening with the severity of the loss, all risks of a service failure derive an acceptable failure frequency. Such a risk-based notion of dependability is also applicable in situations when the loss refers to something else than personal damage or loss of live, as we will detail in subsection III-A.

We thus consider trust as the justified and commonly accepted argumentation that all service failures are sufficiently low with respect to the overall risk of their occurrence. The argumentation for trust then needs to be based on evidences and rationales in order to be as objective as possible and not based on beliefs. Obviously, cultural differences may lead to different borders of risk acceptance and thus also to different feelings of trust. Nevertheless, trust should be defined for larger group of people so that these people commonly accept the dependence on the product and its services. In this paper, we focus on the threats of trust and the means to achieve trust as both significantly changes when we transition from traditional systems to autonomously acting systems that collaborate at runtime to render a higher-level functionality (application service).

III. Building of Trust

As trust depends on the functional and non-functional properties of a system, the engineering methods and techniques applied during development time clearly have a significant impact. Ensuring properties of trust affects any typical development phase, from requirements elicitation to final validation testing. Since *confidence*, *justified belief* and *assurances* are key notions in relation to trust in cooperative autonomous systems, we suggest to employ methods and techniques from the field of safety engineering - because in safety engineering a sound argumentation (bolstered by sufficient evidence) as to why the safety requirements will be met is front and center of the engineering artifacts. A suitable and widely accepted concept for building such an argumentation is the assurance case [4], which might be specified using GSN (Goal Structure Notation) [5] or SACM (Structured Assurance Case Metamodel) [6].

Thus, we envision a trust assurance case as the backbone artifact for building trust at development time. However, we argue that for open cooperative systems development time assurances are often not sufficient (due to context uncertainties and unknowns) or not acceptable from a performance point of view (due to worst case assumptions and consequent detrimental effects on key system properties such as availability). For safety, we introduced a corresponding runtime approach in ConSerts [1]. Moreover, for safety and some additional properties of dependability, we recently introduced the augmented concept of digital dependability identities [2].

Apart from checking assumptions made at development time, it would also be possible at runtime to monitor the performance of the cooperating systems against the promises they made. Ultimately, a reputation metric could be introduced as one additional indicator of the level of trust in a system. This idea was already brought up in one of the early landmark papers on autonomic computing [7]. Here the authors propose storing of information about a system's reputation in order to address the need of computing trustworthiness in potential collaborators. We plan to augment our approach accordingly and propose storing of reputation on a public ledger, as shown in Fig. 1.

In this Section, we sketch out means for building trust at development time and at runtime. Thereby, we put special emphasis on aspects which, up to now, have been insufficiently reflected in our runtime approaches.

More precisely, argumentation for trust needs extend at runtime with provision of mechanisms that assure trust in situations that are encountered for the first time. In regard to this, in subsection III-B we present mechanisms used for argumenting trust at runtime using predictive simulation methods, evidence of evaluated system properties and data collection from the field.

A. *Formation of vehicle platoon*

When they are part of an automotive smart ecosystem [8], autonomous vehicles can dynamically and securely deploy software applications (also called smart agents) that can, for example, enable formation of platoons by enabling secure exchange of information regarding context and root planning.

In a platoon, multiple vehicles are traveling in the same direction closely following each other. By joining a platoon, a vehicle can optimize its fuel consumption and reduce CO2 emissions by closely driving to the vehicle in front. Moreover, platoons are expected to increase the safety in highways by advanced vehicle-to-vehicle communication that allow complex maneuvers.

Towards achievement of the strategic goal of forming platoons, software components that enable collaborations between vehicles are deployed on these vehicles as black boxes and in this particular case they implement the operational goal of sending and receiving information about speed, distance and state of a vehicle. However, such a software component may not operate according to its specifications. One extreme cause of specification violation is the malicious faults contained within software components. According to Lapries taxonomy [3], a malicious fault is a fault introduced on purpose by a developer. When a vehicle provides incorrect service because of malicious behavior of a software component, the whole platoon will be affected. For example, if one vehicle receives the information that the vehicle in front is seven meters away and starts accelerating and then suddenly brakes because, through radar it has identified that the minimum distance is violated, then the vehicles with internal combustion engine will not save but will increase their fuel consumption. This happens because of string instability as described in [9], [10]. In the worst case, vehicles in a platoon can even crash into each other.

Traditionally, if a smart agent passes all verification test cases, then it is deployed on a vehicle. However, neither can malicious behavior of software components be detected through systematic testing, nor can all situations encountered at runtime be forseen at design time. Therefore, parts of the evaluation needs to be shifted at runtime as wewill extend in the next subsection.

B. Building trust at runtime

Building of trust requires additional aspects to be considered, such as: evaluation of control function' trustworthiness and discovering of any malicious behavior, if possible in a simulated environment. Early detection of malicious behavior enables activation of fail-over behavior that can bring an autonomous system in a safe state.

When the actions of one vehicle are the result of the control decision of the software application running on the vehicle, as presented in subsection III-A, then these control decisions need to be predicted. A novel method of building trust in a software application by predicting its behavior at runtime in a simulated environment is the one we introduced in [8]. Evaluation in a simulated environment needs to be performed faster than the wall clock. We call this evaluation *predictive simulation* that requires abstractions of the software behavior received as executable specifications. Running the behavior of a software component in a predictive simulation gives time to the system to react to situations that have been discovered to be dangerous. Using the outputs of system evaluation with a predictive simulation, safety at runtime can be achieved with the approach we introduced in [11].

Abstract specifications are created towards the scope of evaluation. If, for example, scheduling behavior is to be analyzed, then the abstraction will contain the timing behavior of the software component. If the function interaction between different applications needs to be evaluated, then the abstraction will contain the functional behavior of a software component. If the protocol of communication needs to be evaluated, then the abstractions will contain inputs and outputs of the sofwtare component. Predictive evaluation at runtime is then performed through a secure execution of the abstract specifications in a simulated environment. The smart agent itself is executed within a Trusted Execution Environment (TEE), of the vehicle's Embedded Control Unit (ECU). Building of trust in the overall evaluation requires comparison of the two executions and creation of a *conformity reputation* that provides conformity evidence between the smart agent and its specifications.

Automatic deployment of smart agents without warning messages for the driver can be enabled. Given the safety critical nature of an autonomous vehicle, this mechanism requires considerations of security aspects. For instance, the smart agent needs to be signed before deployment (arrow "2" in Fig. 1). While securing automation, code signing would also enable the vehicles to verify the authenticity of the software application, that is, verify its source and check if it has not been altered after being signed (during transit).

Additionally, the integration between systems is done through smart the contracts that are implemented on a blockchain. Smart contracts will include some incentive within, that adds up to the reputation of the provider, in case the contract gets mutually signed. Record of reputations of all providers is stored on the blockchain for future reference. System providers with high reputation are, therefore, more likely to be trusted in the future.

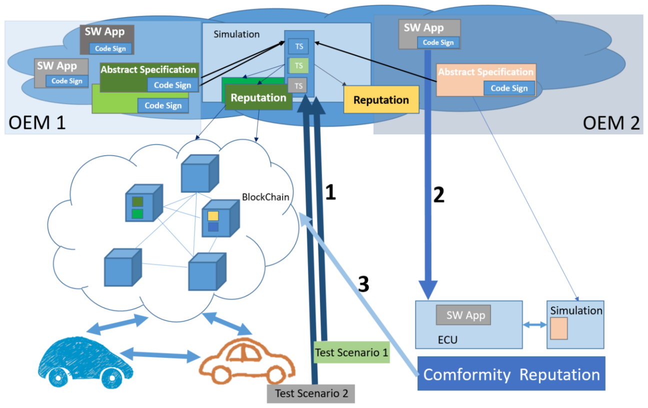

Fig. 1. Infrastructure that enables building trust in automonous collaborative vehicles

The vehicle that wants to join the platoon, would also needs to check if the platoon is controlled by a trusted entity. To achieve this, we propose the use of digital certificates, which are being issued by the car manufacturer (or a consortium of manufacturers) acting as a Certificate Authority (CA). Upon joining the platoon, the vehicle would first check against the digital certificate of the leading vehicle and accordingly decides if it trusts the CA that issued that certificate.

Trust also needs to be build based on evidence about relevance of both verification scenarios and verified system properties. Computation-based trust is formed around reputation of systems behavior. Information about reputation can come as a recommendation from a different party, and then the source and the target community for reputation need to be compatible [12]. One way of achieving compatibility between communities is by creating dedicated environments where test cases can be collected, shared and reputation of system computed and analyzed based on same criteria. This enables creation of a common framework for building arguments towards trust at runtime.

At the system level, a particular effort in this regard is the concept of *Digital Dependability Identities (DDI)* [2]. A DDI is a container capable of containing, analyzing, manipulating and reasoning about models of many relevant dependability aspects such as hazard, safety analysis and security threat models at once. This is possible due to the formal integration of these aspects within the DDI. Due to the DDI's extensability, adding validation scenario models to the DDI allows a more comprehensive reasoning about trust assurance. Collection of scenarios where software functions are verified at design time, derived from situations encountered at runtime creates evidence about the relevance of test situations and builds trust in relevance of test cases. In Fig. 1 the collections of test scenarios from the field is depicted with arrow "1"

IV. Related Work

Different understandings of trust and reputation have been applied in research in the past years. These understandings lead to a various number of definitions. Most of them are centered around terms like "firm belief", "decision making", "scoring", "ranking", "behavior information" and "feedback". Together with the definitions, there are multiple methods used to compute trust and reputation values. Some methods calculate trust value from reputation parameters, the others perform reputation calculation by aggregating trust parameters. Mainly scholars differentiate between belief-based trust [13], [14], [15] and computation-based trust [16], [17] and the need for argumenting trust has been presneted in [18].

Trustworthiness of a trustee was defined as a voluntary behavior of not taking advantage of a trustor's vulnerable position when making self-serving decisions that conflict with the trustor's objectives [19]. In context of safety critical situations in the automotive domain, trustworthiness of a collaborator cannot be granted by default, it needs to be computed. In order to support computation of trust our hierarchical classification of goals evaluation (presented in subsection III-A) is based on the hierarchical nature of decision making of Strategical decisions, tactical decisions and operational decisions described in [20], [21].

V. Conclusions and Future Work

Trust assurance will be fundamental to the success of cooperative autonomous systems. Due to unknowns and uncertainties at design time, trust assurance requires a shift of assurance measures into runtime. In this paper, we first sketched out a definition of trust inspired by the risk-based notion of dependability and centered it around engineering activities. At development time, the engineering activities are centered around the argumentation of relevant trust properties within a trust assurance case. At runtime, they comprise dynamic checks and reasoning (e.g. as supported by DDI) and, in addition, might comprise means such as predictive simulation, which we discussed in a bit more detail. The prediction is centered towards discovery of malicious behavior of smart agents and enables autonomous systems to start a fail over behavior in case of identified risks. Future work will include implementation of methods for building trust at runtime.

Acknowledgment

This work has been funded by the German Ministry of Education and Research (BMBF) through the research project CrESt (Collaborative Embedded Systems).

References

[1] D. Schneider and M. Trapp, "Conditional safety certificates in open systems," in *Proceedings of the 1st workshop on critical automotive applications: robustness & safety*. ACM, 2010, pp. 57–60.

[2] D. Schneider, M. Trapp, Y. Papadopoulos, E. Armengaud, M. Zeller, and K. Höfig, "Wap: digital dependability identities," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2015, pp. 324–329.

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[4] R. Hawkins, I. Habli, T. Kelly, and J. McDermid, "Assurance cases and prescriptive software safety certification: A comparative study," *Safety science*, vol. 59, pp. 55–71, 2013.

[5] T. Kelly and R. Weaver, "The goal structuring notation–a safety argument notation," in *Proceedings of the dependable systems and networks 2004 workshop on assurance cases*. Citeseer, 2004, p. 6.

[6] Object Management Group, "Structured Assurance Case Metamodel (SACM) 2.0," https://www.omg.org/spec/SACM/, 2018, [Online; accessed 17-March-2019].

[7] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, no. 1, pp. 41–50, 2003.

[8] E. Cioroaica, T. Kuhn, and B. Buhnova, "(Do not) trust in ecosystems," in *Proceedings of the 41th International Conference on Software Engineering (ICSE-NIER 2019)*. ACM, 05 2019, preprint available at https://www.researchgate.net/publication/331498836_Do_Not_Trust_in_Ecosystems.

[9] L. Cui, J. Hu, B. B. Park, and P. Bujanovic, "Development of a simulation platform for safety impact analysis considering vehicle dynamics, sensor errors, and communication latencies: Assessing cooperative adaptive cruise control under cyber attack," *Transportation Research Part C: Emerging Technologies*, vol. 97, pp. 1–22, 2018.

[10] J.-N. Meier, A. Kailas, R. Adla, G. Bitar, E. Moradi-Pari, O. Abuchaar, M. Ali, M. Abubakr, R. Deering, U. Ibrahim *et al.*, "Implementation and evaluation of cooperative adaptive cruise control functionalities," *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 1110–1115, 2018.

[11] R. Adler, P. Feth, and D. Schneider, "Safety engineering for autonomous vehicles," in *Dependable Systems and Networks Workshop, 2016 46th Annual IEEE/IFIP International Conference on*. IEEE, 2016, pp. 200–205.

[12] S. Ruohomaa and L. Kutvonen, "Trust management survey," in *International Conference on Trust Management*. Springer, 2005, pp. 77–92.

[13] I. Serov and M. Leitner, "An experimental approach to reputation in e-participation," in *2016 International Conference on Software Security and Assurance (ICSSA)*. IEEE, 2016, pp. 37–42.

[14] N. Dessi, B. Pes, and M. G. Fugini, "A distributed trust and reputation framework for scientific grids," in *2009 Third International Conference on Research Challenges in Information Science*. IEEE, 2009, pp. 265–274.

[15] R. de Oliveira Albuquerque, F. F. Cohen, J. L. T. Mota, and R. T. de Sousa Júnior, "Analysis of a trust and reputation model applied to a computational grid using software agents," in *2008 International Conference on Convergence and Hybrid Information Technology*. IEEE, 2008, pp. 196–203.

[16] B. Zong, F. Xu, J. Jiao, and J. Lv, "A broker-assisting trust and reputation system based on artificial neural network," in *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2009, pp. 4710–4715.

[17] H. Rahimi and H. El Bakkali, "Towards a new design for trust reputation system," in *2012 International Conference on Multimedia Computing and Systems*. IEEE, 2012, pp. 943–948.

[18] M. Gharib and P. Giorgini, "Analyzing trust requirements in socio-technical systems: a belief-based approach," in *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer, 2015, pp. 254–270.

[19] Ö. Özer and Y. Zheng, "Trust and trustworthiness," 2017.

[20] E. Hollnagel, A. Nåbo, and I. V. Lau, "A systemic model for driver-in-control," 2003.

[21] H. H. Van der Molen and A. M. Bötticher, "A hierarchical risk model for traffic participants," *Ergonomics*, vol. 31, no. 4, pp. 537–555, 1988.