(Do Not) Trust in Ecosystems

Emilia Cioroaica
Embedded Systems Quality Assurance
Fraunhofer IESE
Kaiserslautern, Germany
emilia.cioroaica@iese.fraunhofer.de

Thomas Kuhn
Embedded Systems
Fraunhofer IESE
Kaiserslautern, Germany
thomas.kuhn@iese.fraunhofer.de

Barbora Buhnova
Faculty of Informatics
Masaryk University
Brno, Czech Republic
buhnova@mail.muni.cz

***Abstract*—In the context of Smart Ecosystems, systems engage in dynamic cooperation with other systems to achieve their goals. Expedient operation is only possible when all systems cooperate as expected. This requires a level of trust between the components of the ecosystem. New systems that join the ecosystem therefore first need to build up a level of trust. Humans derive trust from behavioral reputation in key situations. In Smart Ecosystems (SES), the reputation of a system or system component can also be based on observation of its behavior.**

In this paper, we introduce a method and a test platform that support virtual evaluation of decisions at runtime, thereby supporting trust building within SES. The key idea behind the platform is that it employs and evaluates Digital Twins, which are executable models of system components, to learn about component behavior in observed situations. The trust in the Digital Twin then builds up over time based on the behavioral compliance of the real system component with its Digital Twin. In this paper, we use the context of automotive ecosystems and examine the concepts for building up reputation on control algorithms of smart agents dynamically downloaded at runtime to individual autonomous vehicles within the ecosystem.

***Index Terms*—Smart Ecosystems, Automotive, Virtual Evaluation, Building Trust, Malicious Behavior**

I. Introduction

Until recently, the software architectures of automotive platforms focused solely on the static deployment of control functions. Control functions were deployed and integrated by engineers of automotive Original Equipment Manufacturers (OEMs). During system operation, only minor changes were performed. Future automotive platforms based, e.g., on the new POSIX-based Adaptive AUTOSAR standard [1] will support dynamic deployment of automotive smart agents. This will be an enabler for smart ecosystems, which comprise a changing number of participants pursuing different goals and interacting with each other to achieve these goals in the best possible manner. A smart ecosystem could, for example, force vehicles entering a city to download smart agents that activate a speed limit in these vehicles that allow a maximum of 30 km/h.

The resulting dynamic architectures will yield new challenges for the software engineering of ecosystems. For example, the admission of smart agents into existing systems will become a relevant future topic. The admission of a smart agent into an ecosystem will enable this component to join an ecosystem, and for example permit a smart agent to supervise specific control functions of a vehicle, such as ensuring silent operation at night. Ecosystem admission should be based on functional correctness and on trust in the third-party component entering the ecosystem. In our work, we focus on the latter, as building trust in ecosystem components requires a new approach to testing, one that predicts the trustworthiness of a component to handle situations properly, instead of testing its correct implementation.

In this paper we propose a virtual Hardware-in-the-Loop (vHiL) testbed that supports rapid runtime evaluation of smart agents in smart (automotive) ecosystems. Based on this testbed, we introduce a novel strategy for building trust in ecosystems and ecosystem components by computing information about the reputation of smart agents. To this end, we employ the concept of Digital Twins (DT) of smart agents, which are used during operation of the smart agents to assess their runtime behavior. The evaluation results are then used to override detected malicious behavior before it takes effect, and to build up agent reputation over time. In this paper, we outline the whole approach and propose a solution for two specific challenges associated with it.

II. Emerging Trends

During the transition of individual embedded systems to ecosystem participants, formerly isolated systems are equipped with open interfaces that enable communication and collaboration. This opens up possibilities for achieving extended business goals and facilitating innovation, as third-party applications may connect to these systems. At the same time, however, it introduces new research challenges related to safety and security risks, especially in the context of safety-critical smart ecosystems. A safety-critical smart ecosystem, for example, needs to guarantee that the driving speeds in the vicinity of schools in cities are low.

A. Formation of Digital Ecosystems

Considerable work has been done in analyzing and describing Software Ecosystems (SECOs) formed around software products [2]. The recent classification from 2015 [3] emerged from a similar classification of Systems of Systems [4]. As presented by us in [5], there is a distinction between Smart Ecosystems (SES) and Software Ecosystems (SECOs). SES are formed around cyber-physical systems (CPS) and their software and hardware components can be provided by different actors, such as the government or an organization, but also by a software company or an individual developer.

The hardware-software interaction within an SES thus requires additional checks of trust.

The key difference between digital ecosystems and systems of systems is that digital ecosystems involve actors with goals, which significantly influences the dynamics within an ecosystem [6]. In cooperation, the actors might have not only collaborative goals, but also competitive goals, which may influence the health of the ecosystem [7], [8]. In smart ecosystems, where hardware and software components of cyber-physical systems are provided by different actors, malicious behavior can be introduced along with software components by actors who join a smart ecosystem based on declared collaborative goals, but who are actually acting in competition.

Until now, admission to a digital ecosystem has been based on the actors' commitment to published roadmaps organized and provided by an ecosystem orchestrator for the long term [9]. Safety-critical ecosystems, however, are particularly faced with the challenge of intended malicious behavior which may be hidden in the smart agents. As a consequence, besides being functionally correct, a safety-critical ecosystem also needs to assess the participants' trustworthiness before granting them admission. Assessing the trustworthiness of ecosystem participants requires new platforms that enable behavior evaluation at runtime.

B. Formation of Trust in Ecosystems

While the risks associated with maliciously behaving actors in smart ecosystems have been recognized there is still no satisfactory solution for the issue yet. We argue that one of the most promising directions in this context can be driven by the formation of trust in smart agents.

Looking at the beginnings of development of autonomic computing systems, reputation can be a good indicator of the level of trust in a system. The authors of [10] propose the possibility to store information about a system's reputation in order to address the need for trustworthiness in potential partners. However, this approach implies the execution of a system's behavior in the real world and hence entails the risks associated with executing malicious behavior that can be introduced along with smart agents. A solution for building reputation in a system without executing its behavior in the real world has become possible by recent advancements in the field of automation. The notion of a Digital Twin (DT) has been introduced by NASA [11] as a realistic digital representation of a flying object used in lab-testing activities. Since then, the notion of DT has also been adopted in the emerging Industry 4.0 [12] for representing the status of production devices and to enable forecast of change impacts.

While the concept of DT is so far only being employed for simulation and testing detached from object operation, we see great potential in the incorporation of its usage into an object's operation time (the smart agent in our case), which we are proposing in this paper. In this way, we can create safe conditions for building reputation via multiple observations of a component's behavior in specific situations. In the smart ecosystem domain, the reputation of a smart agent can be built based on collected evidence about the decisions that the smart agent makes in these specific situations.

III. Method for Building Trust

In this section, we introduce our approach to building trust in smart agents, which is based on the concept of Digital Twins and their safe real-time evaluation during the runtime of smart agents. Although the approach is general, we narrow it down to the context of automotive smart ecosystems, which allows us to be more specific in the explanation.

In automotive smart ecosystems, a vehicle receives a new smart agent which interacts with the automotive control software as a black box. The vehicle executes this agent on one of its Electronic Control Units (ECU). Building trust in this black box requires reputation from a trusted source, in our case the ecosystem of the city where the car is driving. In order to build up this trust, our platform (introduced in Section IV) evaluates a DT of the smart agent in a virtual environment in three steps:

1) The smart agent is downloaded to the vehicle together with its corresponding DT. The DT is an executable description of the algorithm that can be controlled in a simulated environment. Complementary to the algorithm, the DT defines an acceptable behavior range for the combination of input and output values and the internal state of the algorithm.
2) Phase 1 of our approach validates both the correctness and the trustworthiness of the smart agent by evaluating its DT behavior in the context of a simulation. The DT shows a projection of the behavior of the smart agent's control algorithm in all situations. This projection yields an abstracted behavior that reflects the control algorithm's behavior with bounded accuracy. In this way, the process of building trust in the smart agent does not require software execution on a real ECU, but merely evaluation of the behavior of the DT in a secured virtual environment (cf. Fig. 1. phase 1).
3) Phase 2 builds trust in the conformity of the smart agent with its DT (cf. Fig. 1. phase 2). This phase requires execution of the smart agent on the ECU. Conformity is checked by validating the behavior of the DT against the behavior of the real smart agent. This also requires a trustworthy monitoring platform.

In this paper, we focus on the concept of Phase 1. It can be realized with two possible simulation strategies: (1) *Pure predictive simulation* is based on a set of well-defined situations that evaluate DT behavior in a virtual environment, while (2) *linked predictive simulation* virtualizes the vehicle's current situation and predicts sensor data to reflect a forecast situation in the near future. Linked predictive simulation evaluates the DT in situations that are not covered by pure predictive simulation. For example, when approaching a crossing at an intersection, our platform monitors the DT. The system needs sufficient time to react to the decisions of the smart agent and to possibly override them. Therefore, the platform evaluates the smart agent's DT with faster simulation speed during

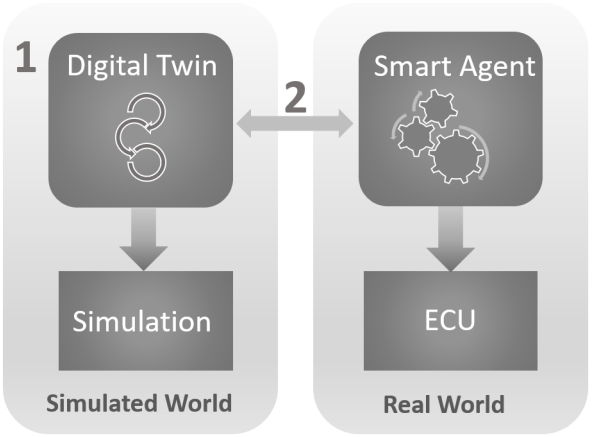

Fig. 1. Phases of the Method

vehicle operation. If the DT decides to continue driving at full speed, the outcome of this decision needs to be evaluated in advance in order to safely override the behavior of the smart agent, in case it leads to hazardous situations.

For evaluating the outcome of a control algorithm, a holistic scenario of the situation in which the system operates needs to be created. When evaluation happens at runtime (i.e., linked predictive simulation), the situation encountered at runtime needs to be represented accurately. This is accomplished by using DTs of systems that are part of the ecosystem and form the technical situation in which the smart agents are evaluated.

IV. Testing Platform

Our test platform evaluates the trustworthiness of a smart agent before it is fully admitted to a smart ecosystem. For this reason, it must ensure that the control algorithm under evaluation cannot detect that it is under evaluation, i.e., that it cannot distinguish whether it is interacting with a real system, or with a simulation model. This includes addressing the uncertainties of real and simulated sensors, as well as the creating realistic scenarios that accurately reflect the technical situation encountered at runtime or that are anticipated in the near future. Based on this, our platform needs to deploy mechanisms that accurately compute the level of trust in an agent.

The metamodel that describes the logical structure of our test platform is depicted in Fig. 2. Smart agents are software components that are dynamically deployed on systems. Digital Twins represent the behavior of smart agents with bounded accuracy. DTs are necessary because when a smart agents trustworthiness is evaluated, its DT must not detect whether it is interacting with real or virtual entities. Consequently, malicious behavior will be visible in the virtual world before it takes effect in the real world.

A. (Do Not) Trust the Virtual Evaluation

One possible sandboxing approach is to ensure that the smart agent cannot detect that it is under evaluation in a simulated environment. In our opinion, such a sandbox is almost impossible to create in a safe and secure manner if the smart agent can implement complete Turing behavior. Therefore, we propose reducing the expressiveness of the DT for the simulation. This assures that the DT cannot implement detection algorithms.

The simulation is a virtual representation of a vehicle that is composed of multiple *Software Components* that implement for example control software. *Virtual HW Components* are specialized software components that implement hardware models, for example sensors, actuators, microprocessors, memories, etc.

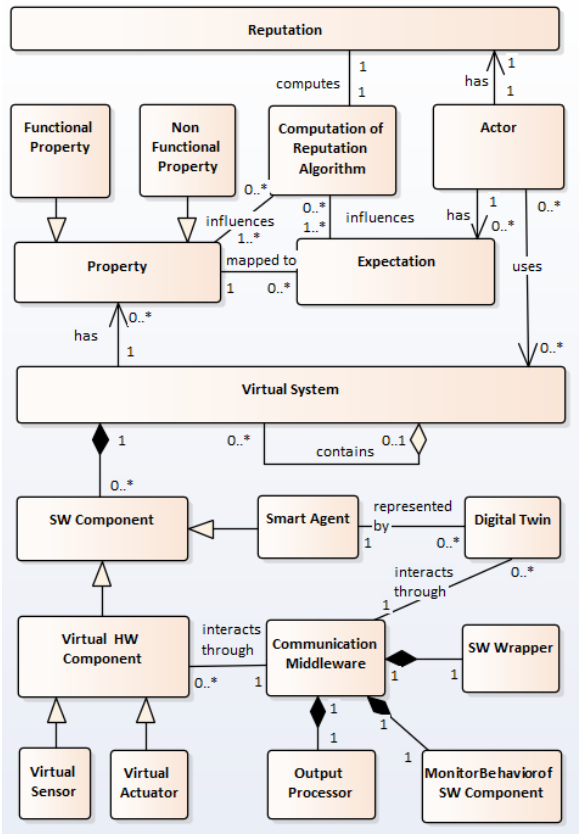

Fig. 2. Metamodel of the Test Platform@Runtime

Given the abstracted behavior of the virtualized hardware resources, the frequency and value range of the output signals, as well as signal quality can differ considerably from real-world components. This offers an opportunity to detect that the evaluation is happening in a virtual environment. Therefore, *Output Processors* as the interface between the DT and the hardware resource implement signal preprocessing.

Abstract platform simulations create the opportunity for malicious software algorithms to detect whether they are interacting with virtual components or real systems, e.g., by sending commands to challenge the platform and provoke known responses. For example, by monitoring response times, or monitoring the consistency of sensor signals, a DT can detect a simulation. Therefore, besides reducing the expressiveness of the language for programming a DT, we also need

to reduce its abilities to monitor its surroundings. The *Software Wrapper* component provides mechanisms that assure that DTs are not able to retain their state, neither monitor nor observe the passing of time. The platform interface of the *Communication Middleware* permits only very specific and limited interactions between DTs and platforms. Furthermore, the *Monitor Behavior of SW Component* monitors the behavior of software components in order to detect suspicious interaction patterns.

B. (Do Not) Trust the Smart Agent

Trust in a smart agent is built up over time via the concept of reputation. Our platform enables predictive simulation, which involves the execution of algorithm behavior at much higher speeds than wall clock time, as well as simulation of the same situations over and over again. This detects, for example, hidden behavior whose activation is quite unlikely. The reputation of a smart agent is influenced by monitoring *Functional Properties* and *Non Functional Properties*, such as reaction times, of the smart agent behavior. Actors, i.e. organizations that provide the system or components and users define *Expectations* towards actors. The reputation of an actor is therefore also liked to the Expectations that another actor has in the context of a collaboration. Therefore, the reputation of an actor introducing a system to the market needs to be considered as well.

Smart agents start with an initial level of trust when they join the ecosystem. This level is updated according to evidence continuously derived from the virtual evaluation of the smart agents Digital Twins. This includes multiple behavior executions in a virtual environment, uncommon situations, reference behavior, or situations that have been recorded before. This increases the probability that malicious behavior will show up in simulations and not when controlling real systems.

V. Discussion and Conclusion

Technological progress opens up opportunities for business growth but despite the existence of safety standards it also leads to higher safety and security risks. With every new major accident, standards are reconsidered and improved. However, the process of improving the standards takes much longer than the deployment of methods and tools that can reduce these risks. Deployment of platforms that enable virtual verification of smart agents at runtime represents our vision tackling risks and building trust in safety-critical ecosystems.

A. State of the Work and Preliminary Results

Our platform is based on FERAL simulator [13] and comes as an extension of the platform we describe in [5], which enables testing of automotive smart ecosystems. The platform already supports the testing of control algorithms to avoid obstacles. Virtual entities and real world entities collaborate with each other within the platform to jointly avoid obstacles. Our method and platform extension presented in this paper enhance the level of trust in smart agents with special attention on the importance of performing trustworthy virtual evaluation.

B. Future and Ongoing Work

The scope of the entire concept encompasses many interesting research questions for further investigation such as (a) checking conformity between the DT and the smart agent and (b) creating Digital Twins that accurately represent the behavior of the smart agent. In addition, we are currently researching a DT language with reduced expressiveness for functional models. The language must not be Turing complete so that detection of simulated environments can be prevented, and it should enable identification of malicious smart agents based on their Digital Twins. Furthermore, our platform could identify situations in which malicious behavior is exposed. Future work will include the implementation of phase 2 which guarantees conformity of smart agents with DT behavior and implements countermeasures in case of non conforming behavior.

Acknowledgment

This work has been funded by the German Ministry of Education and Research (BMBF) through the research project CrESt (Collaborative Embedded Systems). The contribution of B. Buhnova was supported by ERDF/ESF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16$_0$19/0000822).

References

[1] "AUTOSAR," https://www.autosar.org/, [Online; accessed 03-September-2018].

[2] K. Manikas, "Revisiting software ecosystems research: A longitudinal literature study," *Journal of Systems and Software*, vol. 117, pp. 84–103, 2016.

[3] K. Manikas, K. Wnuk, and A. Shollo, "Defining decision making strategies in software ecosystem governance," *Department of Computer Science, University of Copenhagen*, 2015.

[4] "Office of the deputy under secretary of defense for acquisition and technology, systems and software engineering. systems engineering guide for systems of systems," *Version 1.0. Washington, DC, ODUSD(A and T)SSE*, 2008.

[5] E. Cioroaica, T. Kuhn, and T. Bauer, "Prototyping automotive smart ecosystems," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2018.

[6] K. Manikas and K. M. Hansen, "Reviewing the health of software ecosystems–a conceptual framework proposal," in *Proceedings of the 5th International Workshop on Software Ecosystems (IWSECO)*, 2013, pp. 33–44.

[7] K. M. Popp, "Goals of software vendors for partner ecosystems–a practitioner s view," in *International Conference of Software Business*. Springer, 2010, pp. 181–186.

[8] J. Bosch and H. H. Olsson, "Ecosystem traps and where to find them," *Journal of Software: Evolution and Process*, p. e1961, 2018.

[9] J. Bosch and P. Bosch-Sijtsema, "From integration to composition: On the impact of software product lines, global development and ecosystems," *Journal of Systems and Software*, vol. 83, no. 1, pp. 67–76, 2010.

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

[11] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Modeling, simulation, information technology & processing roadmap," *National Aeronautics and Space Administration*, 2012.

[12] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–572, 2015.

[13] T. Kuhn, T. Forster, T. Braun, and R. Gotzhein, "Feralframework for simulator coupling on requirements and architecture level," in *Formal Methods and Models for Codesign (MEMOCODE), 2013 Eleventh IEEE/ACM International Conference on*. IEEE, 2013, pp. 11–22.