2020 28th INTERNATIONAL CONFERENCE ON SOFTWARE, TELECOMMUNICATIONS AND COMPUTER NETWORKS - (*SoftCOM 2020*)

SoftCOM 2020

Virtual Conference
September 17 - 19

Towards Creation of Automated Prediction Systems for Trust and Dependability Evaluation

Emilia Cioroaica
Fraunhofer IESE
Kaiserslautern, Germany
emilia.cioroaica@iese.fraunhofer.de

Stanislav Chren
Masaryk University
Brno, Czech Republic
chren@mail.muni.cz

Oum-El-Kheir Aktouf
Univ. Grenoble Alpes, Grenoble INP, LCIS
Valence, France
oum-el-kheir.aktouf@grenoble-inp.fr

Alf Larsson
Ericsson AB
Stockholm, Sweden
alf.larsson@ericsson.com

Ram Chillarege
Chillarege Inc.
Raleigh, NC, USA
ram@chillarege.com

Thomas Kuhn, Daniel Schneider, Christian Wolschke
Fraunhofer IESE
Kaiserslautern, Germany
{thomas.kuhn, daniel.schneider, christian.wolschke}@iese.fraunhofer.de

***Abstract*—We advance the ability to design reliable Cyber-Physical Systems of Systems (CPSoSs) by integrating artificial intelligence to the engineering methods of these systems. The current practice relies heavily on independent validation of software and hardware components, with only limited evaluation during engineering integration activities. Furthermore, our changing landscape of real-time adaptive systems allows software components to be dynamically included or re-distributed within a Cyber-Physical System (CPS), with mostly unknown implications on the overall systems integrity, reliability and security. This paper introduces an approach consisting of scientific and engineering processes that enable development of concepts for automated prediction systems for evaluating the dependability and trust of CPSoS. This significantly moves the security and reliability design process ahead by opening the doors for far more relevant design strategies and the opportunity to develop protocols, methods, and tools aimed at dealing with a wide variety of platforms with poorly calibrated reliability characteristics.**

***Index Terms*—CPSoS; Cyber-Physical System, Virtual Validation, Trust, Dependability.**

I. Introduction

A Cyber-Physical System (CPS) is formed by interconnected embedded and computational systems that communicate via communication networks in order to accomplish a defined objective [1]. For example, a vehicle can be considered a CPS since it consists of embedded systems communicating with each other via the vehicle bus. Multiple cooperating CPSs form a Cyber-Physical sytem of systems (CPSoS).

CPSs are frequently integrated into mission-critical systems, such as power grids, telecommunication or transportation. Such systems require a high level of dependability and trust. Dependability can be defined as the ability of a system to deliver a service that can be justifiably trusted [2]. Over the years, the dependability concept has evolved to integrate other qualitative attributes, such as availability, reliability, safety, integrity, maintainability and trust.

Different understandings of trust have been applied in research in the past years. These understandings have led to a number of definitions. Most of them focus on terms such as "firm belief", "decision making", "scoring", "ranking" and "behavior information". Scholars mainly differentiate between belief-based trust [3] and computation-based trust [4]–[6]. In this open position paper that invites the reader to join or follow this line of research, we consider the computation-based trust, resulted from the digital evaluation of a collaborator's trustworthiness. In the literature, the trustworthiness of a trustee is defined as the voluntary behavior of not taking advantage of a trustor's vulnerable position when making self-serving decisions that conflict with the trustor's objectives [7]. In our assessment of trust, a trustor is *the user of a service* who can trust a trustee who is *the provider of the service*, to satisfy its needs and expectations linked to a trustum which is *the provided service*. For example, the trustworthy hardware-software interaction is a safe and secure interaction between a software and a hardware that is free from software exploits of hardware resources and results from digital computations. Additionally, in this paper we consider trustworthy hardware-software interaction as safe and secure interaction between a software and a hardware resource that is free from software exploits of hardware resources.

The paper is structured as follows: SectionII sets the scene for the need of trust and dependability prediction. Section III details the steps of our approach. Section IV outlines the state of our current progress and Section V presents the conclusions.

II. Motivation

In this section, we will discuss the challenges and trends in present CPSs and CPSoS addressed in our work.

A. Runtime coupling of hardware and software components

Engineers have a long tradition of designing CPSs as closed boxes. However, presently, CPSs are getting composed of HW and SW components developed disjointedly, which often interact with each other for the first time during operational time. Certain patterns of HW-SW interaction can trigger erroneous states that do not necessarily lead to the manifestation of a failure in the system that would result in the provision of incorrect service. This can happen because of the applied failure mitigation mechanisms. Furthermore, some interacting patterns of system components can be failure-prone in certain configurations and failure-free in others. Therefore, a system may become unsafe by showing inexplicable failures that cannot be predicted [8].

To enable collaboration between self-adaptive CPSs that form CPSoS such as car platooning, the hardware resources of a CPS can be enriched with software components on the fly. Given the fixed hardware resources of autonomous vehicles, ad-hoc CPSoS can only be formed through software updates.

However, since SW components and HW components are provided independently, their joint evaluation would be highly non-deterministic. This often makes it impractical to model and predict the behavior of the complete system at design time. Instead, the CPS should be verified at runtime, with a focus on time-bound behavior in the short-term future [9].

B. Challenges in HW-SW interaction

Seaborn and Dullien [10] have shown that specific HW-SW interaction patterns can be faulty and can lead to serious system failures that manifest into security threats. The DRAM row hammer effect, e.g. can be repeatedly induced by software exploiting the hardware structure of DRAM modules. In a row hammer attack, repetitive and specially crafted memory patterns can cause the memory cells to leak their charges and to interact electrically among each [11]. If it interacted with a hardware resource characterized by a different physical structure, the software alone would not have the same problem. This experience shows that there are situations in which HW-SW interaction can cause unpredicted failures. Because of the severe impact they cause, this type of failures needs to be addressed.

The challenge in analyzing faulty HW-SW interactions is caused by research and development activities that have, over time, evolved into two separate directions, namely FEF (Faults, Errors, Failures) analysis of hardware resources and FEF analysis of software components [12], [13]. Analysis of faulty HW-SW interactions prevents deployment of future platforms (combination of HW resource and SW component) likely to contain faulty behavior, with a major impact on the users and fixing costs. For example, a considerable amount of vulnerable DRAM resources have already been deployed in multiple systems when the hammer effect was discovered [10].

In the field of safety analysis, engineers need to evaluate safety concept of systems and systems of systems at early development stages. At the point of evaluation, decision about the concrete platforms to be used may have not been made yet or information is not disclosed. In these situations simulation engines are used to couple abstract representation of a platforms' behavior. Reasoning about the safety of a system based on simulated abstract behavior definition of composing platforms enables an efficient evaluation and fast decision making, keeping the cost of change low. However, in such evaluations, the overall platform behavior is considered. Faulty effects of software updates on the same hardware resources are not considered and therefore the scope of the evaluation is limited to a fix set of HW-SW combinations in fixed platforms. Such approaches at most scale towards interoperability during runtime, as they do not address openness concerns that enable reusability through software or hardware variations.

C. Emerging Trends

Technological advances enable the reuse and fast creation of hardware resources in innovative products. Thanks to the generic usage of GPUs, for example, hardware reuse is becoming more likely. Reusable hardware resources are enhanced with software components that make these products flexible. Application of specific software components enables reuse of the same hardware resource in different application domains. Software updates make it possible to fix hardware defects and realize enhanced functionality in the same application domain. In such situations, confidence in HW-SW interactions needs to be built up, for example by simulating the interacting components and subsequent conformity checks between the simulation models and the corresponding HW or SW components as presented in [14].

In the automotive domain, achievement of full autonomy is envisioned through usage of advanced driver assistance systems (ADAS) capable of assessing the environment through communication with interconnected vehicles from the same or even from different Original Equipment Manufacturers (OEMs). Flexibility comes with various growth opportunities for CPS and CPSoS, and evaluating the dependability and trust of these systems is of crucial importance in order to address the security threads. Security threats raise dependability concerns, which also have implications for safety.

D. Need for Change and Speed

Technology is advancing at a much faster speed than standards can be adapted in order to safely formalize validation procedures. By missing out on technology disruptions, businesses are in jeopardy of fading away and missing growing opportunities. Well-established car manufacturers, for example, are way behind innovative companies that welcome technological advances and exploit the flexibility of software to create competitive goods.

In the automotive domain in particular, technological progress and sustained effort put into optimizing logistics and improving citizens' lives lead to fundamental business changes [15]. The traditional Business-to-Customer model is changing into a Business (B1) to Business (B2) to Consumers (C) Model. With B1 delivering products to B2 and B2 offering services to C on the basis of these products. B2 is extending a hardware product delivered by B1 with software updates to enhance or customize functionality and achieve increased performance. In this way, B1 enables the provision of products that can be individually tailored to the needs of the consumers. Competitive advancement of B2 over other service providers is achieved through software function updates.

Existing methods and tools can capture dependability information about systems or system components. However, their use is limited, and they do not address the runtime coupling of CPSs in the context of a CPSoS and dynamic interaction of HW-SW components within a CPS. Schneider et al. [16] argue that in industrial practice, tools and methodologies can reliably capture dependability information about systems and system components and even integrate such information in order to derive information about the dependability of the resulting system of systems. However, the usage of these tools is limited to design time evaluation, as they are unable to capture the dynamic HW-SW interaction of components at runtime. Moreover, tools that support dependability evaluation need to perform automatic reasoning on failure propagation between components and between systems as well.

When systems components couple at runtime, evaluation of the resulting system's dependability needs to be performed at runtime as well.

E. Trust Prediction

Proposed models for trust prediction focus on the different properties of trust and their resistance to different attacks [17]. However, weaknesses against failures and malicious intrusions are commonly encountered when dealing with CPSoS and not yet well handled, especially when these systems experience emergent behavior and properties. Different than existing approaches for trust prediction presented in [12], our approach considers the effects of faulty HW-SW interactions and its propagation at the system and system of systems level.

III. Creation of Automated Prediction Systems

In this section, we introduce our approach which comprises a research process and an engineering process (see Fig. 1). Together, they support the development of systems for predicting trust and dependability of CPSoS. The overall workflow is shown in Fig. 1. The activities in our approach can be grouped into five building blocks: The *Artificial Intelligence* and *Ontology Definition* blocks are part of the research process. The engineering process consists of the *Catalog Creation*, *Simulation*, and *Evaluation* blocks. The research process provides input to the *Catalog Creation* block. Additionally, with the *Catalog Creation* it provides input to the *Simulation* block. The *Evaluation* block incorporates visualization techniques and methods.

A high level of system autonomy can be realized through provision of software updates on the fly. However, when software components integrate with each other and with hardware resources dynamically at runtime, prediction of consequences for systems' dependability as well as trust evaluation need to be done at runtime as well. This can be achieved through deployment of automated predictive systems that can synthesize dependability information from different sources and enable system reconfiguration in case of foreseen failures.

In this context we foresee the necessity of reasoning about faults, errors and failures (FEF) of constituent platforms from a new perspective that we express in a conceptual model. From the conceptual model we derive a taxonomy and an ontology. The taxonomy enables a hierarchical classification of FEF and of behavioral models of platforms from abstract to concrete. The ontology models the relationships between the elements of taxonomy and enables automatic retrieval of information, thus making the first bridge between scientific results and engineering activities. The formal methods employed at design time, define the compliance model between ontology and the simulation models used for prediction and it forms the second bridge between scientific results and engineering processes. The ontology and the taxonomy together provide a structure for catalogues of FEF to be deployed on systems and populated with information about faulty HW-SW interactions of platforms. This information is retrieved from the field through monitoring. Information from the catalogue is then send to Deep learning algorithms that decide for the best fitting software patches to existing hardware resources. Additionally, by analyzing the faulty HW-SW interaction, intelligent algorithms derive guidelines for software applications and in this way support the engineering methods of systems. Guidelines are defined in terms of software behavior that is not allowed to execute on hardware resources that have been identified as sensitive.

The top most priority of trust evaluation of systems operating in the field is their ultimate safety. And in the safety domain, a wide range of all possible deviations and formalization of system functions have been defined. Therefore, at the operational level, we quantify achievements of trust and dependability based on evidence derived from runtime system execution. A trusted, dependable behavior is the one that does not exceed a threshold of allowed deviations. The threshold for accepted deviations is under the limits defined by anti-goals [18]. For example, a runtime predicted behavior of a software smart agent shows an activation of speed limits within 30 seconds. Actuators are designed to guarantee timely activation if no more than 3 seconds of delay is reached. A timing behavior executed in the threshold between 30s and 33s is considered trusted. In this example, when timing deviations exceed the threshold, fail-over behavior is executed instead. At the operational level, fail-over behavior can be activated when faults in the field are discovered. Fail-over behaviors can bring a system into a safe state in face of hazardous events [19].

Therefore, the development of such prediction systems need to extend the existing state of the art and practice. The following processes represent our vision for development of such systems:

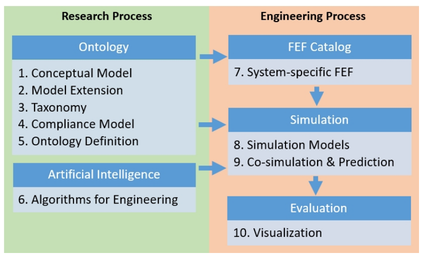

Fig. 1. Workflow for the creation of automated prediction systems.

A. Research Process

The purpose of the research process is the development of an extensible ontology capable of capturing knowledge about the faults, errors, and failures (FEF) of platforms within a CPS. The development of an ontology is based on the methodology by Stanislav et al. [20]. We further enrich it with more detailed conceptual modeling and by considering its extensibility to various platforms. The interactions between the activities in the research process are shown in Fig. 2 and consist of multiple artifacts:

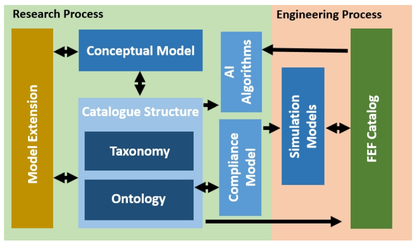

Fig. 2. Research process activities and their mutual interaction.

- A *Conceptual Model* that defines the main concepts and captures the interactions between hardware resources and software components as well as the information about FEF and their dependencies. Our conceptual model makes use of concepts from existing models, such as Failure Mode and Effects Analysis (FMEA) [21], Laprie's taxonomy [2] and ODC [22]. In the construction of the conceptual model similarities between platforms like CPUs, sensors, and memory for the basics for creating abstract simulation models. Continuous iteration on the conceptual model targets accommodation of a large amount of platforms and starts by looking at the specifics of platforms and platform components and indentify commonalities between these platforms. . This step forms the basis for *Step 3* and *Step 4*.
- A formalized *Model Extension*, which enables further extension of current concepts to provide support for the integration of new platforms.
- A *Taxonomy* emerged from the conceptual model that allows the description of dependability concepts for platforms at different levels of abstraction via a *generalization - specialization* relationship. Being able to capture a platform's FEF at different levels of abstraction offers the possibility to balance the trade-off between speed and accuracy of the analysis and enables simulation based on fault injections at different levels.
  On one hand, it gives the possibility to balance between efficiency and effectiveness of analysis. For example, when evaluation needs to be performed efficiently, then reasoning can be achieved by using simulated abstraction models of platforms. Effective evaluation can be performed using simulation models with higher levels of details, however it requires more time.
- An *Ontology* that supports the classification of FEF by formalizing the relationships between the elements of the taxonomy. It provides a knowledge base that is accessible by humans and processable by machine algorithms.
- A *Compliance Model* between the ontology and platform simulation models that resembles the system behavior and describes the platform simulation models at different levels of abstraction. Formal methods are applied to define the properties of the models used in the simulation activities.
- *Deep Learning Algorithms* that support the engineering methods for CPSs and CPSoSs, and are used to analyze data monitored on the field and stored in the FEF catalog. By analyzing SW-HW interaction patterns that may be faulty, deep learning algorithms can generate guidelines for SW component behavior that has a trustworthy interaction with hardware resources. For example, a guideline could recommend the maximum number of writes per minute in a memory row.

B. Engineering Process

The engineering process wraps around the research approach. Its activities are shown in Fig. 3 and consist of following spets:

- Creation of *FEF Catalogs* that contains up-to-date information about FEF for specific platforms and enables the definition of relevant fault models. System-specific catalogs are instances of the catalog structure specified by the ontology. FEF catalog are populated with monitored data about the HW-SW interaction within a CPS.
- Creation of *Simulation Models* at different levels of abstraction. Different behaviors of simulation models are mapped to the entities in the taxonomy and respect the rules of the ontology. The classification captured in the taxonomy offers the possibility to create a balance between efficiency and effectiveness of the analysis. For example, when an evaluation needs to be performed efficiently, simulation models at a higher level of abstraction are used. Effective evaluation can be performed using simulation models with higher levels of details at the cost of increased computation time. Simulation models are defined according to the Compliance Model. Furthermore, the faults stored in the FEF catalog are injected into the simulation model to allow simulation and prediction of failure behavior.
- *Prediction of HW-SW interaction* through simulation. We use a co-simulation framework to couple simulation models. Prediction of trust at runtime is performed through implementation of novel trust method we introduced in [14].
- *Evaluation* of the results of dependability analysis of CPSs and CPSoSs through meaningful visual representation. Humans are visual beings and the results of a virtual evaluation are easily understood when their effects are displayed in a meaningful way. This can be achieved by coupling with gaming engines such as Unity 3D [23] or Unreal [24].

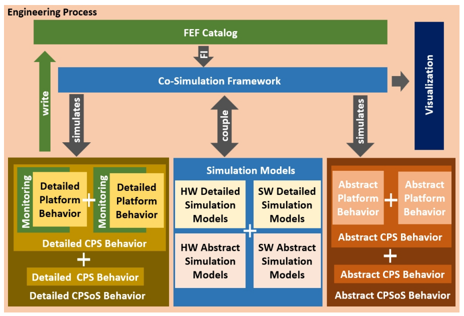

Fig. 3. Engineering process of creating automated prediction systems.

C. Application Scenario

This section presents an application scenario demonstrating the usage of our approach in the automotive domain. The approach is universally applicable, but we illustrate it in the context of the automotive domain, We envision an automated system that supports optimization and self-adaptation of CPSs within a CPSoS by predicting a system's dependability in specific situations encountered in the field at runtime. The deep learning algorithms within the system determine the failure mitigation activities based on insights gained from monitoring the system operation at runtime. We use the concept of Runtime Models as described by Blair et al. [25] to enable continuous evolution of a CPS by deriving design-time guidelines from runtime activities. A CPS is represented by an autonomous vehicle. A group of vehicles communicating with each other form a CPSoS, e.g, a vehicle platoon formed on highways to optimize traffic and reduce fuel consumption. A platoon is a convoy in which vehicles move closely to each other, thereby benefiting from reduced air friction and lower energy consumption.

In a fully autonomous scenario, download of software components enrich existing hardware resources and form platforms that can enable a vehicle to lead or join a platoon. In this process different hardware resources are enriched with software updates in order to enable specific functions. A software component can enable reading of sensor measurements, and provision of information to another software control function in charge of keeping a safe distance between the vehicles. Platform failures in such situations can have disastrous effects not only for the vehicle itself, but for the whole platoon. If the safe distance is violated, accidents can happen.

In order to prevent accidents in the platoon, a system conceptualized following engineering processes we have presented is capable of predicting dependability and trust in vehicles and vehicle platoons. Fig. 4 depicts the deployment scheme of such a system. The dotted arrows represent wireless transmission of SW applications from the cloud to a vehicle and the exchange of information between vehicles. In order to achieve a satisfactory degree of autonomy, different OEMs need to share a minimum amount of information. In our example, the vehicle on the far left provides an abstract simulation model of itself to the other vehicle.

The operation of such a system consists of two phases. In the first phase (marked as 1 in Fig. 4), the co-simulation framework executes the SW component together with the targeted HW resource as soon as a SW component is downloaded on the vehicle. Next, the resulting platform is checked against the errors and failures that can emerge from faulty HW-SW interactions. Faults for the emerging platform are retrieved from the catalog. By monitoring the HW-SW interactions in detail, new information about faulty interactions can be discovered. This information is written back to the catalog. Data from the catalog is then fed into Deep Learning Algorithms. These algorithms can be used to derive guidelines for software development towards assuring that the software component doesn't have faulty interactions with a hardware resource. Additionally, for faulty interaction patterns identified in the field, software patches can be selected and downloaded to the vehicle. We envision intelligent algorithms to perform:

- Derive guidelines for app development. Such guidelines can check for dangerous coding patterns of an application that can be faulty in combination with certain HW resources.
- Derive guidelines for the design of improved HW resources.
- Support the selection of platforms during the engineering stages of CPS development.
- For faulty interaction patterns found, identify software patches that can be downloaded to the vehicle to prevent these faults.

In this way, functional design will be free from identified faults of CPSs emerges from identifying the best-fitting platforms and is backed up by information derived from data collected from the field through monitoring activities. By considering the effects of such platforms on the guarantees and demands of interacting platforms and on the guarantees and demands of other CPSoSs, functional correctness extends to the level of CPSoSs. All these phases happen in the part ”1” of the framework depicted in Fig. 4 that can be deployed on the vehicle and in the cloud.

In the second phase of the system's operation (marked as 2 in Fig. 4), the abstract models of CPS platforms are executed for deriving an abstract behavior of a CPS in the context of a CPSoS. The level of abstraction is targeted towards the scope of the evaluation. Virtual evaluation based on the execution of abstract models is efficient in terms of response time. Such models are used for performing fast evaluation of platforms, CPSs and CPSoSs. In order to avoid latency, this phase of the system operation needs is executed on the vehicle.

IV. State of the Work and Preliminary Results

The conceptual model, artefact of the research process is under development together with an initial definition of the taxonomy. The system implementation is supported by preliminary results. For example, the co-simulation platform presented in [26] enables coupling of different simulation models, with the possibility of visualizing failure effects using a Unity 3D game engine, as presented in [27]. The taxonomy

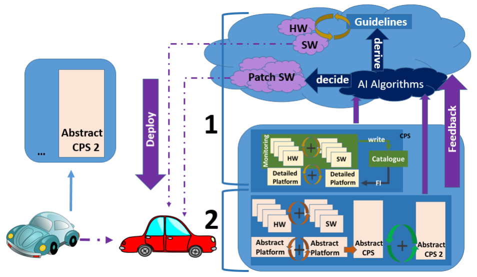

Fig. 4. Deployment and execution scheme

applied in DAGGTAX [28] enhances the possibility to identify reusable design solutions and eliminate unfeasible feature combinations. A novel method for predicting trust at runtime using simulation has been introduced in [14]. Initial work on fault analysis and fault models in CPSs has been done by Barnier et al. [29].

Acknowledgments

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 FE: Technologieorientierte Kompetenzenfelder -MWVLW "Neue Erprobungskonzepte fuer sichere Software in hochautomatisierten Nutzfahrzeugen" and from ERDF/ESF CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence (No. CZ.02.1.01/0.0/0.0/16019/0000822.

V. Conclusions

To improve the dependability of CPSs and CPSoSs, it is necessary to extend the current state of the art by addressing current trends and challenges related to the development of CPSs in various domains. For that we need to bring together two worlds that have been kept apart, namely analysis of software failures and analysis of hardware failures. In this paper, we presented an approach for conceptualizing systems for predicting the dependability and trust in CPSs characterized by the independent provision of HW resources and SW components. It involves building a general taxonomy that captures the complex relationships between faults, errors and failures by looking at the HW-SW interactions within a platform. Our approach uses deep learning algorithms supported by an ontology and simulation models to predict faulty behavior of components that have never interacted before.

References

[1] B.-H. Schlingloff, H. Stubert, and W. Jamroga, "Collaborative embedded systems-a case study," in *Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC), 2016 3rd International Workshop on*. IEEE, 2016, pp. 17–22.
[2] 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.
[3] 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.
[4] 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.
[5] G. Yin, D. Shi, H. Wang, and M. Guo, "Repcom: Towards reputation composition over peer-to-peer communities," in *2009 International Conference on Computational Science and Engineering*, vol. 2. IEEE, 2009, pp. 765–771.
[6] B. Qureshi, G. Min, and D. Kouvatsos, "Collusion detection and prevention with fire+ trust and reputation model," in *2010 10th IEEE International Conference on Computer and Information Technology*. IEEE, 2010, pp. 2548–2555.
[7] Ö. Özer and Y. Zheng, "Trust and trustworthiness," 2017.
[8] E. A. Lee, "Cps foundations," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*. IEEE, 2010, pp. 737–742.
[9] L. Bu, Q. Wang, X. Chen, L. Wang, T. Zhang, J. Zhao, and X. Li, "Toward online hybrid systems model checking of cyber-physical systems' time-bounded short-run behavior," *ACM SIGBED Review*, vol. 8, no. 2, pp. 7–10, 2011.
[10] M. Seaborn and T. Dullien, "Exploiting the dram rowhammer bug to gain kernel privileges," *Black Hat*, vol. 15, 2015.
[11] "Ars technica." [Online]. Available: https://arstechnica.com/information-technology/2016/10/using-rowhammer-bitflips-to-root-android-phones-is-now-a-thing/
[12] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computing Surveys (CSUR)*, vol. 42, no. 3, p. 10, 2010.
[13] "Differentiating between a hardware problem and a software problem." [Online]. Available: https://crubsy.com/differentiating-between-a-hardware-problem-and-a-software-problem/
[14] E. Cioroaica, T. Kuhn, and B. Buhnova, "(do not) trust in ecosystems," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2019, pp. 9–12.
[15] J. Arbib and T. Seba, "Rethinking transportation 2020-2030: The disruption of transportation and the collapse of the internal-combustion vehicle and oil industries," *San Francisco, United States: RethinkX*, 2017.
[16] D. Schneider, M. Trapp, Y. Papadopoulos, E. Armengaud, M. Zeller, and K. Höfig, "Wap: digital dependability identities," in *Software Reliability Engineering (ISSRE), 2015 IEEE 26th International Symposium on*. IEEE, 2015, pp. 324–329.
[17] Y. Ruan and A. Durresi, "A survey of trust management systems for online social communities–trust modeling, trust inference and attacks," *Knowledge-Based Systems*, vol. 106, pp. 150–163, 2016.
[18] P. Fenelon, J. A. McDermid, M. Nicolson, and D. J. Pumfrey, "Towards integrated safety analysis and design," *ACM SIGAPP Applied Computing Review*, vol. 2, no. 1, pp. 21–32, 1994.
[19] 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.
[20] T. Sanislav, G. Mois, and L. Miclea, "An approach to model dependability of cyber-physical systems," *Microprocessors and Microsystems*, vol. 41, pp. 67 – 76, 2016.
[21] D. H. Stamatis, *Failure mode and effect analysis: FMEA from theory to execution*. ASQ Quality Press, 2003.
[22] R. Chillarege, "Odc measurement and analysis-industry applications," *Technical report*, 2007.
[23] "Unity 3d engine." [Online]. Available: https://unity3d.com/
[24] "Unreal engine." [Online]. Available: https://www.unrealengine.com
[25] G. Blair, N. Bencomo, and R. B. France, "Models@ run. time," *Computer*, vol. 42, no. 10, 2009.
[26] 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.
[27] 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.
[28] S. Cai, B. Gallina, D. Nyström, C. Seceleanu, and A. Larsson, "Design of cloud monitoring systems via daggtax: a case study," *Procedia Computer Science*, vol. 109, pp. 424–431, 2017.
[29] C. Barnier, A. Mercier, O.-E.-K. Aktouf, and J.-P. Jamont, "Toward an embedded multi-agent system methodology and positioning on testing," in *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2017, pp. 239–244.