A Paradigm for Safe Adaptation of Collaborating Robots

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

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

Emrah Tomur
Ericsson
Instambul, Turkey
emrah.tomur@ericsson.com

ABSTRACT

The dynamic forces that transit back and forth traditional boundaries of system development have led to the emergence of digital ecosystems. Within these, business gains are achieved through the development of intelligent control that requires a continuous design and runtime co-engineering process endangered by malicious attacks. The possibility of inserting specially crafted faults capable to exploit the nature of unknown evolving intelligent behavior raises the necessity of malicious behavior detection at runtime.

Adjusting to the needs and opportunities of fast AI development within digital ecosystems, in this paper, we envision a novel method and framework for runtime predictive evaluation of intelligent robots' behavior for assuring a cooperative safe adjustment.

KEYWORDS

Virtual Evaluation, Runtime Prediction, Building Trust, Safety-Critical Systems, Robots

ACM Reference Format:
Emilia Cioroaica, Barbora Buhnova, and Emrah Tomur. 2022. A Paradigm for Safe Adaptation of Collaborating Robots. In *17th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '22), May 18–23, 2022, PITTSBURGH, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3524844.3528061

1 INTRODUCTION

Supported by ultra reliable and low-latency communication of 5G and emergent 6G technology, independently cooperating networked robots greatly benefit from off-board fast computation and cloud-based knowledge sharing paradigms in disaster management scenarios [13]. Parallel development in the domain of teleoperation envisions the robots' capability to coordinate with a high degree of precision [1]. The emerging development in the domain of digital ecosystems [7] will further enable creation of autonomous robotic structures enhanced with AI (Artificial Intelligence) control received through runtime updates. It is expected that these robots will be required to form coalitions similar to living organisms, and will be characterized by mandatory hard real-time constraints on their safe coordination. For example, robots of different nature deployed in safety-critical scenarios, with different computational speeds, will be required to perform high-precision collaborative tasks, by relying not only on commonly shared information, but on safe physical synchronization as well. An illustrative example is the scenario of two robots of different nature that carry together a load while an obstacle is approaching and they need to synchronize on common maneuvers. In situations when the operational context of cooperative intelligent robots changes dynamically and unpredictably at runtime, safe adjustments also need to be planned during runtime and, in our opinion, need to account for evidence of robots' safe intended actions.

Further on, when AI control becomes embedded within software smart agents that control a robot, a major challenge in detecting possible malicious behavior arises from the non deterministic nature of the AI component. This happens because under the control of an AI, an ongoing adaptation of an intelligent robot can either indicate a behaviour enhancement or a malicious deviation. Concretely speaking, at the operational level, it is expected from an intelligent control that for the same set of inputs processed at different moments in time, different outputs might be provided. For intelligent behavior developed within digital ecosystems [7], it is very likely that sooner or later, intended faults will be injected into a system together with an update. 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.

In our work we address the latter, as it requires a security and safety co-mitigation strategies. In this paper, in particular, we are regarding the collaboration between two self-adaptive AI-controlled robots as an emergent single distributed self-adaptive system that exposes an emergent and unknown behavior. To address the challenge of safe collaborative synchronization between them, we introduce a method and a framework that supports the safe adaptation under the control of distributed software agents that can potentially contain malicious behavior.

In what follows, Section 2 provides a summary of the state of the art that emphasizes research and development trends which are currently emerging. Section 3 introduces our method for collaborative synchronization between heterogeneous robots by underlying our vision together with future research challenges. Section 4 presents the conceptual architecture of the framework that supports the implementation of the method together with the concept of a Domain Specific Language (DSL) that guides the definition of software behavior in a way that enables capturing of runtime artifacts used in the runtime decision making process.

2 EMERGING TRENDS

2.1 The need for speed

Digital ecosystems' facilitation for fast deployment of products and services capable to promptly meet users' requests and expectations brings tremendous business value [7]. Initially delivered with a quality right above the minimum required threshold, systems upgrades

SEAMS '22, May 18–23, 2022, PITTSBURGH, PA, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9305-8/22/05.
https://doi.org/10.1145/3524844.3528061

are then provided during operation. This philosophy, successfully applied for the development and maintenance of information systems that execute in stable environments, has recently emerged in the domain of safety-critical systems [30] through runtime deployments of AI components. However, such process implemented in the context of safety-critical systems operating in dynamic environments is currently raising considerable safety concerns [29].

It is envisioned that traditional systematic development of systems' safe reconfiguration will increasingly be replaced by an agile design-time & runtime co-engineering approach which will further develop into faster adaptation cycles under the control of AI components [11, 12].

2.2 Unknown Behavior under Uncertainties

In the quest of enabling a highly trusted behavior of systems operating in safety-critical scenarios, approaches for devising runtime assurance cases have been proposed [31]. Runtime monitoring in particular [21], can indicate whether a system internal self reconfiguration is appropriate in a given technical setting or the activation of a fail-over behavior is needed [31]. Current approaches focus on assuring a known behavior in an uncertain environment [18] while in the future, it is expected that intelligent systems will not only operate in uncertain context, but will be controlled by intelligent software that evolves over time as well [9]. Such behavior is considered to be unknown at every moment in time because when faced with a similar situation, it is expected to display an improved behavior. Yet, a correct learning and knowledge-building process is still not guaranteed [18].

While the current efforts are directed towards designing a trusted safe and secure runtime operation, a systematic insertion of intended faults, such as logic bombs [3] can remain dormant and get activated at the "right moment" to support a planned attack that can lead to a wide range of a malicious behaviors.

2.3 Safe Reaction to Security Intrusions

Self-adaptive systems are known to adapt to internal dynamics with an autonomous structure that enables behavior reconfiguration at runtime [23, 32]. The internal dynamics of the system can be influenced by the runtime changing of goals or by detected internal faults handled in a variety of dynamic risk management schemes [22, 25]. However, as discussed previously, under intelligent control, a deviation is not necessarily an evidence of faulty behavior, it can as well be an evidence of adaptation of the intelligent control [9]. This aspect is receiving an increased attention within the safety-critical domain, leading to a current recognition of the risks of unknown behavior in uncertain context [20]. Tremendous effort has then been shifted towards understanding the intelligent control [2, 20]. However, without a real envisioned solution yet. In part, this is due to the fact that the process of assuring safety is driven by standards developments. And within standards, guidelines for the AI component integration are are still not given primary attention. The process of standard development is a much slower than the process of technological development that pushes innovation. Therefore, faster solutions for safeguarding the cooperative behavior of robots operating in safety-critical environments has to build on emerging engineering and research trends.

Looking back in the history of automatic computing, the engineering of single autonomic systems have provided the possibility of deploying self-management capabilities according to MAPE (Model, Analyze, Plan, Execute) [21] control architecture. Typically, the knowledge that accompanies the control loop deals with the dynamics of available resources and internal faults. Further research has then enabled the possibility to perform a collaborative learning and adjustment mechanism through coordination of multiple MAPE loops with specific patterns [33] followed by the provision of guarantees for the common adaptation decisions [10]. However, no engineering solution that can make the distinction between trusted sporadic adaptation of collaborative robots and sporadic software failure caused by intended malicious faults exists yet.

In our previous work we have proposed an approach for building trust in the unknown emergent behavior through runtime execution of Digital Twins (DTs) [16]. In this paper we elevate the concept of runtime prediction with a framework that supports the collaborative behavior of self-adaptive robots that need to execute synchronous maneuvers in uncertain environments. For this, we propose a new schematic of collaboration that enables evidence-based behavioral prediction within a runtime simulation environment. Traditional approaches based on monitoring techniques would require a detailed design of the system [8, 21] followed by detailed model deployment within a runtime virtual evaluation. In the context of predictive simulation, these approaches eliminate the possibility of fast behavioral execution which is a mandatory prerequisite in the process of runtime predictive simulation.

3 THE TRUST BUILDING METHOD

In this section we detail our vision for building trust in cloud-supported AI-controlled collaborating robots by regarding the collaboration as a distributed self-adaptive system with emergent and unknown behavior. In this context we propose the prediction of the computational control within a simulation environment that enables detection of unwanted effects. To this end we propose a runtime execution of specialized simulation models orchestrated in two directions of abstraction.

3.1 System Coordination under Uncertainties

Our proposed framework for assuring safe coordination between heterogeneous intelligent robots consists of nine steps grouped into three phases distributed on the system and in the cloud. Figure1 depicts the process of collaborative synchronization based on runtime execution of specialized simulation models. The derivation of those specialized simulation models is a design-time activity that will be detailed in subsection 3.2.

The Paradigm for Collaborative Safe Adjustment. On a robot, a virtual and abstract representation of interacting system components, including the intelligent control (AI), is created by feeding specialized abstract models with real-time data gathered within the *Sensing* step part of the *Learning* phase. In the *Learning* phase, the specialized models are not executed yet, but information is gathered for the creation of a concrete technical situation that requires behavioral synchronization. Concretely, the *Framing* step creates the focus of the evaluation by selecting the set of models specifically designed for evaluating the major characteristic of a behavior such as timing synchronization, functional interaction or communication between components. When fed with real-time data, these specialized models will become Specialized Digital Twins (SDTs) as will be detailed next. Within the *Learn* phase, the *Sense* step supports knowledge formation for a single system, whereas the *Sharing* of concern-oriented real-time data, and runtime specialized model forms the prerequisite for creating digital twins that depict a collaboration in the next phase.

In the cloud, during the next *Prediction* phase, the previously selected abstract models of behavior that describe a concrete technical situation are executed within the *Execution* step performed in a simulated environment and is followed by the *Capture* step. Within the *Execution* step, the specialized simulation models of single systems are evaluated in relation with simulation models of interacting systems and system components. Knowledge of the overall cooperative behavior is then derived within a virtual simulation environment. Because the prediction requires execution of simulation models at a much faster rate than the wall clock, in this step, different types of behavioral abstractions are used specifically designed for the scope of the evaluation. After a runtime execution, the *Capture* step leads to the creation of runtime images that describe the intended reaction within a collaboration. Then, by feeding virtual images that represent the intended actions of collaborative systems to a *Safety Evaluator*, an evidence-based dynamic safety argumentation is performed. The safety evaluation is performed by comparing the runtime specialized images against a collection of safety claims and system safety goals. Only after executing an internal validation process, the *Safety Evaluator* passes the images to the next phase. In this way, the *Safety Evaluator* provides confidence of validated predictive claims that will lead to immediate commands triggered within *Adjust* step part of the *Collaborative Reaction* phase on the system itself. The *Sharing* step within the *Collaborative Reaction* phase permits sharing of information to the collaborating physical robots enabling the synchronous safe reaction.

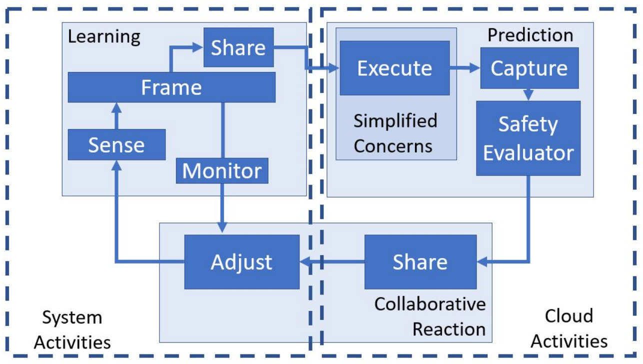
The collaborative safe adaptation patterns that have been learned, virtually evaluated and then shared between the robots can be triggered immediately on the system as a response to a new situation. The immediate reaction is depicted in Figure 1 by the arrow between the *Sense* and *Adjust* step. Future synchronous reactions rely on a Conformity *Monitoring* that evaluates the similarity between a newly encountered situation and a previously learned and virtually and virtually validated situations. The conformity monitoring process digital artefacts defined according to a specialized domain specific language, part of the framework as will be described in Subsection 4.2.

The Engineering Approach for Assuring Safety. Traditionally, the adaptation engine which is the controller of the self-adaptive system implements the goal-oriented adaptation logic within a feedback loop that adapts the managed system [21]. The feedback loop can reactively respond to a violation of the adaptation goals [5], or, by tracking the behavior of the managed system, can adapt its own behavior in response to the anticipation of possible violations [26]. In our approach, the predictive simulation builds on the proactive policy of a self-adaptive system feedback loop in an evidence-based dynamic safety argumentation that evolves over time. If no internal configuration can be matched to the fulfillment of the adaptation goals, then the controller triggers the fail-safe operating mode.

For assuring the safety of a planned collaborative reaction, in our approach the internal behavioral landscape of a system is reflected into multiple virtual images built and maintained over time. An image is a snapshot of a system's internal reactions to external factors evaluated as being trustworthy. This means that the implementation of a collaborative safe reaction relies on a system's access to an image formed by runtime abstract models that upon execution provide computed evidence of trust that feeds in a complex runtime safe argumentation process encoded within the *Safety Evaluator*. This component provides an assurance of the safe reaction. By feeding the abstract specialized simulation models with real-time data, the dynamic safety argumentation extends the considerations of a system's internal capabilities and includes environmental factors that influence its operation. For a single robot, the safety runtime dynamic argumentation maps the internal and external events from concrete technical situations to abstract functions and their relations in a runtime image within the internal behavioral landscape.

The runtime image must deal with a system's safe reaction by executing functions at an appropriate level of abstraction. Because the safe reaction has a direct impact in the physical world, the level of abstraction of used functions shall be carefully selected by considering a) the availability of resource computation on the system and b) the closest to real behavior representation. Generally, the level of runtime dynamic conception is correlated with the set of system's functional representation. Because the holistic evaluation of functional interaction is performed within the simulated environment, the set of real-world systems and software functions need to be delivered together with their abstract representations.

Figure 1: The Paradigm for Collaborative Safe Adjustment



3.2 Simplifying Concerns

Assuring a trustworthy prediction of the intelligent control in interaction with systems and system components within a simulated environment is crucial. The intelligent software should not detect it is under evaluation, otherwise it can hide its malicious behavior. In particular, because prediction of the behavior requires execution at a much higher speed than the wall clock, a malicious behavior can easily detect it is under evaluation if, for example, it can observe that it was in a given state more frequently than a real-world execution would permit. Typically, well designed safeguards, such as sand boxing [4, 19], are implemented in order to hinder the possibility of a software component to monitor the passing of time while executing a complex behavior. But, under the requirements of fast execution, such safeguards put a considerable heavy load on computation.

In our opinion, a better solution based on simplified concerns has the potential of preventing an intelligent software component to detect it is under evaluation. Initially designed for achieving a clear understanding of either functional or timing behavior of real-time control systems [6], we see great potential of this approach in enabling a concern-directed prediction of the trustworthiness of intelligent software behavior. By focusing the scope of the evaluation to either scheduling, function interaction or communication protocol between the intelligent software and interacting entities (such as software, hardware or subsystem) within a robot, specialized and faster evidence of trust can be achieved. As depicted in Figure2 runtime evidence of trust can be provided through execution of *horizontal abstractions* of a software component or systems behavior, which are directed towards a specific scope of the evaluation and can be executed at every level of *vertical abstraction.* From top to bottom, vertical abstractions can be defined with a range of details varying from a very high level where they take the shape of input/output tables or state charts towards very concrete levels when they are fully implemented. At every level of a robot's behavior vertical abstraction, the horizontal abstraction of the intelligent software behavior can be executed for providing the specialized evidence of trust. Further on, vertical abstractions can propagate evidence of trust between different horizontal levels for assuring the satisfaction of specific system-level goals and on-going coalitions [17].

3.3 Derivation of Specialized Digital Twins

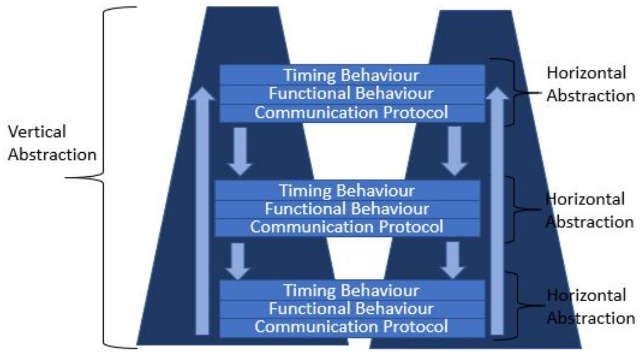
In the literature, digital twins are defined as a combination of real-world data and simulation models [27, 28]. In our work, we elevate the traditional concept of digital twin by performing a concern-driven abstraction process that leads to the creation of specialized abstract models which become specialized digital twins when they are fed with real-time data.

The current approaches for designing self-adaptive systems are based on simplified concerns that make the distinction between handling of domain concerns and adaptation concerns. In the case of heterogeneous autonomous collaborating robots, the domain concerns of one robot relate to the goals of the collaborating partner robot, whereas the adaptation concerns relate to the ego robot. In our approach we are leveraging the principle of concern simplification to the design of software functions, for creating simpler behavioral structures capable to avoid uncontrolled feature interaction. Concretely, we redirect the process of simplifying concerns to the definition of concern-oriented abstract models. For this, the simplified runtime models need to be a result of complex design process and a deep technological perspective needs to be employed in order to capture the core aspects to be evaluated in concrete technical situations within the virtual execution environment. In the process of deriving abstract models, it is important to identify the main concerns and to leave apart the ones that are subordinated to them. Desirably, the subordinated and auxiliary concerns are left apart with the support of a rigorous process designed to guarantee the conceptual integrity of the evolving simplified model. The model derivation process that fits into the methodology introduced in this paper needs to adhere to the following aspects:

(1) Firstly, the creation of simplified models needs to rely on an engineering process that is based on the description of services that the respective component provides at the interface with another component. Because the behavior of the physical system imposes timing constraints that must be handled by the intelligent control, corresponding timing specifications need to be assigned at the communication interface between components as well. In this way, besides guaranteeing the mere functional requirements, the abstraction of the intelligent control can guarantee the specified temporal constraints and proper handling of various environmental effects.
(2) Secondly, the realization of a simplified runtime model for a system needs to be given by an abstract conceptualization of the internal operation of the robot. The abstract conceptualization encompasses an engineering understating and relies on the technical description captured while designing system models.
(3) Third, the abstract conceptualization needs to be an engineering process that builds on human understanding of the system and its component's behavior in an explainable and predictable manner. The incorporation of the engineering knowledge and human prediction in the derivation of abstract models forms the basis for the runtime behavioral prediction. This is further on enhanced by the runtime execution of abstract models within the *Execute* step of the *Prediction* phase.

Shortly, the design process of abstract specialized models captures characteristics of a system's and its components behavior at a specific level of abstraction. This process varies in complexity in accordance with the nature of the system to be abstracted.

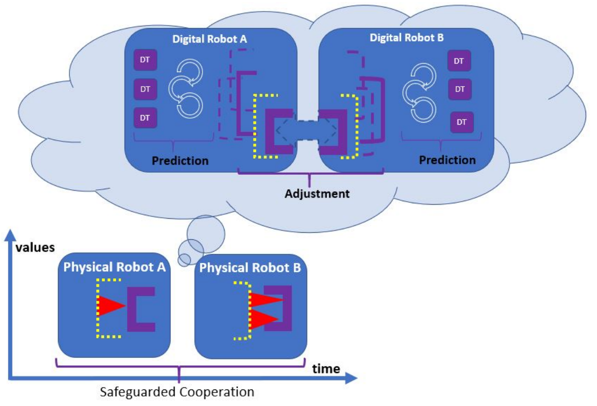
Figure 2: Envisioned execution of parallel abstractions



3.4 Off-board / On-board Distribution

Relying on cloud support, in the edge, a virtual framework performs behavioral predictions through execution of simplified behavioral models of each robot's intelligent control in interaction with designated abstractions of the interacting platform, hardware resources or other software components. When fed with real-time data, these models become specialized digital twins that are executed at a faster rate than the wall clock and provide evidence of trustworthy behavior for specific concerns.

Figure 3: Distribution of the paradigm in an On-board and Off-board computation system



In Figure3, the square brackets depicted with interrupted line represent predicted trusted range of possible paths that feed into planned synchronous maneuvers. By accounting of each robot's safe reaction, a synchronization algorithm outputs the virtually trusted cooperative behavior, which is depicted with square brackets in dotted lines. These predictions are then sent to the real systems. Because one actuation plan may be subject to more severe or less severe risks depending on the immediate operational circumstances, it is on the real systems where the ultimate safe operational decision is taken. For assuring safety, each value range that has been predicted as trustworthy in the cloud has associated a minimum set of events that can be triggered on the real system in case of sudden obstacles approaching. The range of safe values is depicted with thick square brackets on the physical system and on the cloud. The mapping between range of predicted values and the safe events is depicted with triangles on the physical systems.

Overall, our method for assuring a cloud-based safe synchronization of safety critical systems requires multiple challenges to be addressed including (a) specification and design of appropriate levels of vertical abstractions, (b) specification and design of vertical abstraction models at different levels, (c) definition of corresponding horizontal behavioral abstraction for every layer of the vertical abstraction, (d) instrumentation of dedicated abstractions for framing trust evidence in holistic evaluation scenarios, (e) definition of safe events that can be mapped to runtime predictions, (f) derivation of minimal cut set for the tuple (adjusted behavior, safe behavior), and (g) aggregation and perpetuation of specialized evidence of trust at every level of vertical abstraction. In the following section we propose the conceptualization of a framework which aims to address challenge (d) related to the instrumentation of different vertical and horizontal abstractions.

4 THE INSTRUMENTATION FRAMEWORK

In this section, we present the components of the framework designed for enabling the execution of the paradigm for the safe collaborative adjustment between heterogeneous robots.

4.1 Architectural View

As depicted in Figure 4 the framework accounts for real-world and virtual-world elements. Within the *Virtual World*, a *Virtual Platform* which can be a co-simulation framework such as the one presented in [24] can couple and execute different *Specialized Digital Twins*, which are a combination of *Abstract Specialized Models* and *Real-Time Data*. The *Virtual Platform* contains the *Dynamic Safety Evaluator* that evaluated and validates the virtual *Images* resulted from the execution of different digital twins.

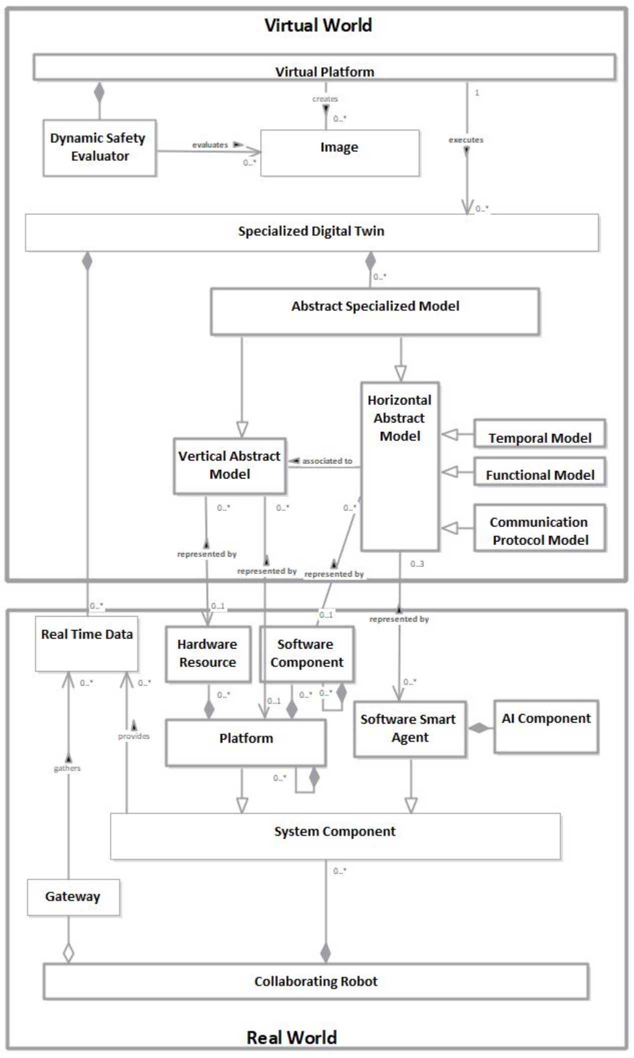
The *Specialized Digital Twins* are digital representations of real-world systems depicted in the figure by *Collaborating Robot. The Real-time Data* that supports the derivation of the Specialized Digital Twins is gathered from the *Collaborating Robot* via a *Gateway*. Typically, a real-world robot is composed of multiple subsystems or *Platforms* which can further be composed of *Hardware Resources*, *Software Components* and/or other *Platforms*. For hardware resources and for platforms that are the atomic combination of hardware resources and software components, *Vertical Abstraction Models* are defined. As discussed before, the execution of vertical abstraction models enables fast behavior evaluation in simulated environments and provide specialized evidence of trust through the execution of associated *Horizontal Abstract Models*. For enabling trust evaluation at software levels, the simple *Software Components* as well as the *Software Smart Agents* characterized by the integration of *AI Components*, are represented by a set of three *Horizontal Abstract Models* in the form of (a) a *Temporal Model* that provides the timing representation of the software behavior used for runtime scheduling evaluation, (b) a *Functional Model* which is a representation of the functional behavior of a software used for evaluating functional interactions between different interacting entities, including platforms, hardware resources or software components, and (c) a *Communication Protocol Model* executed for the evaluation of messages sent between interacting components (hardware, software, platform).

4.2 Conformity Monitoring

When they are delivered as black boxes, the internal states of software components, including AI components, will not be known during execution. However, for the detection of malicious deviations, an approach that enables the monitoring of effects produced by state changes can support the trust building process. These effects can be observed and analyzed by recording the order, type and name of the events triggered at the communication interface. Implicitly, for enabling the observation of behavioral execution, the execution of specialized digital twins need to provide these events. For this, in our approach we propose the design of a domain-specific language for guiding the specification of the digital twins temporal and functional behavior by making a distinction between *normal events* and *decision events*. While normal events describe the receiving of input values, sending output values or interaction with other components, the decision events are these events that cross the architectural boundaries of a component. For example, the sending of a command from one component to another one for increasing speed is a decision event that crosses the architectural boundaries of the initial software component.

For enabling provision of artefacts to the interconnected components of the runtime framework, the domain specific language need to guide the explicit declaration of events that are triggered during

Figure 4: Metamodel of the Framework



an execution along with their types. In this way, the execution of the digital twins will provide a precisely formulated sequence of events that enables the runtime prediction to output artefacts that can be monitored on the system. By instrumenting the definition of the software behavior of the managed system in a way that it exposes observable artefacts, trusted behavior signatures can be derived. Then the monitoring component can check the conformity between the real-world execution of the software component and the virtually trusted valid synchronous behavior and detect deviations. These deviations are indications of a change in the internal and/or the external environmental conditions. In case of unwanted deviations, a reactive feedback loop can be triggered on a single system.

4.3 State of the Work and Preliminary Results

Our framework is based on FERAL co-simulation platform [24] and comes as an extension of the framework introduced in [16]. The platform already supports testing of collaborative control algorithms directed towards avoiding static obstacles [15] in a design-time & runtime co-engineering framework [14]. The method and platform presented in this paper enhance the level of trust during execution in uncertain environments through runtime prediction based on synchronous execution of specialized simulation models.

5 DISCUSSION AND CONCLUSION

By bringing the possibilities of hosting heavy computations in the cloud, the fast development of communication technology is pushing forward the overall business landscape and the opening of safety-critical systems' interface to accommodate software applications developed by third parties. The incentive for immediate business gains leads to another parallel technological progress in AI development that overpasses standardization procedures typically reviewed only in case of already occurred major accidents.

Keeping up with the demands of fast technological progress and emergent trends of accommodating AI components in the control of safety critical systems, in this paper we have proposed a method and a framework for enabling a trusted safe adaptation of intelligent robots collaborating in safety-critical scenarios, by using the concept of runtime predictive execution of behavioral models defined on various levels of horizontal abstraction.

We propose the design of collaborating heterogeneous systems that can be regarded as a complex decentralized system for which a prediction of global behavior can be performed at runtime with the scope of avoiding unwanted emergent behavior that can have severe safety implications. In this way we pave the way towards safe collaborative adaptation which can be based on contractual agreements that are provided to the ecosystem orchestrator. With the support of 5G and 6G development, the orchestrator can perform the prediction in the cloud.

The scope of the entire concept encompasses many challenging research questions for further investigation, some of which we have already outlined at the end of Section 3. The predictive simulation approach deployed in the context of self-adaptive collaborating systems requires an infrastructure that manages the messages exchanged during the adaptation phase. The message protocol needs to handle the "before" and "after" states of adaptation for ensuring consistency. Besides these, we are currently working on the implementation of the framework and on early validation of the method. Because the application of our proposed method relies on delivery of real-world functions with corresponding abstractions for a set of safety-critical functions, our immediate strategy for validation implements a qualitative study regarding of its feasibility and will be followed by a planned quantitative study.

ACKNOWLEDGMENTS

The work was supported This work was supported by the project BIECO (www.bieco.org) that received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952702 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

[1] Alperen Acemoglu, Jan Krieglstein, Darwin G. Caldwell, Francesco Mora, Luca Guastini, Matteo Trimarchi, Alessandro Vinciguerra, Andrea Luigi Camillo Carobbio, Juljana Hysenbelli, Marco Delsanto, Ottavia Barboni, Sabrina Baggioni, Giorgio Peretti, and Leonardo S. Mattos. 2020. 5G Robotic Telesurgery: Remote Transoral Laser Microsurgeries on a Cadaver. *IEEE Transactions on Medical Robotics and Bionics* 2, 4 (2020), 511–518. https://doi.org/10.1109/TMRB.2020.3033007

[2] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilovic, et al. 2020. AI Explainability 360: An Extensible Toolkit for Understanding Data and Machine Learning Models. *J. Mach. Learn. Res.* 21, 130 (2020), 1–6.

[3] Algirdas Avižienis, Jean-Claude Laprie, and Brian Randell. 2004. Dependability and its threats: a taxonomy. In *Building the Information Society*. Springer, 91–120.

[4] Stanley Bak, Karthik Manamcheri, Sayan Mitra, and Marco Caccamo. 2011. Sandboxing controllers for cyber-physical systems. In *2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*. IEEE, 3–12.

[5] Cyril Ballagny, Nabil Hameurlain, and Franck Barbier. 2009. Mocas: A state-based component model for self-adaptation. In *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 206–215.

[6] Mario R Barbacci and Jeannette M Wing. 1987. Specifying functional and timing behavior for real-time applications. In *International Conference on Parallel Architectures and Languages Europe*. Springer, 124–140.

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

[8] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. 2009. Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*. Springer, 48–70.

[9] Adam Bry and Nicholas Roy. 2011. Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE international conference on robotics and automation*. IEEE, 723–730.

[10] Radu Calinescu, Simos Gerasimou, and Alec Banks. 2015. Self-adaptive software with decentralised control loops. In *International Conference on Fundamental Approaches To Software Engineering*. Springer, 235–251.

[11] Maria Casimiro, David Garlan, Javier Cámara, Luis Rodrigues, and Paolo Romano. 2021. A Probabilistic Model Checking Approach to Self-Adapting Machine Learning Systems. (2021).

[12] Maria Casimiro, Paolo Romano, David Garlan, Gabriel A Moreno, Eunsuk Kang, and Mark Klein. 2021. Self-Adaptation for Machine Learning Based Systems. In *Proceedings of the 1st International Workshop on Software Architecture and Machine Learning (SAML)*. Springer.

[13] Wuhui Chen, Yuichi Yaguchi, Keitaro Naruse, Yutaka Watanobe, Keita Nakamura, and Jun Ogawa. 2018. A study of robotic cooperation in cloud robotics: Architecture and challenges. *IEEE Access* 6 (2018), 36662–36682.

[14] Emilia Cioroaica, Stanislav Chren, Alf Larsson, Ram Chillarege, Thomas Kuhn, Daniel Schneider, Christian Wolschke, et al. 2020. Towards creation of automated prediction systems for trust and dependability evaluation. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 1–6.

[15] Emilia Cioroaica, Thomas Kuhn, and Thomas Bauer. 2018. Prototyping Automotive Smart Ecosystems. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE.

[16] Emilia Cioroaica, Thomas Kuhn, and Barbora Buhnova. 2019. (Do not) trust in ecosystems. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 9–12.

[17] Emilia Cioroaica, Akanksha Purohit, Barbora Buhnova, and Daniel Schneider. 2021. Goals within Trust-based Digital Ecosystems. In *2021 IEEE/ACM Joint 9th International Workshop on Software Engineering for Systems-of-Systems and 15th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (SESoS/WDES)*. 1–7. https://doi.org/10.1109/SESoS-WDES52566.2021.00006

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[19] Chris Greamo and Anup Ghosh. 2011. Sandboxing and virtualization: Modern tools for combating malware. *IEEE Security & Privacy* 9, 2 (2011), 79–82.

[20] Ronan Hamon, Henrik Junklewitz, and Ignacio Sanchez. 2020. Robustness and explainability of artificial intelligence. *Publications Office of the European Union* (2020).

[21] Jeffrey O Kephart and David M Chess. 2003. The vision of autonomic computing. *Computer* 1 (2003), 41–50.

[22] Faisal Khan, Seyed Javad Hashemi, Nicola Paltrinieri, Paul Amyotte, Valerio Cozzani, and Genserik Reniers. 2016. Dynamic risk management: a contemporary approach to process safety management. *Current opinion in chemical engineering* 14 (2016), 9–17.

[23] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17 (2015), 184–206.

[24] Thomas Kuhr, Thomas Forster, Tobias Braun, and Reinhard Gotzhein. 2013. FERAL—Framework for simulator coupling on requirements and architecture level. In *2013 Eleventh ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2013)*. IEEE, 11–22.

[25] Fábio L Leite, Daniel Schneider, and Rasmus Adler. 2018. Dynamic risk management for cooperative autonomous medical cyber-physical systems. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 126–138.

[26] Gabriel A Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2015. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*. 1–12.

[27] Roland Rosen, Georg Von Wichert, George Lo, and Kurt D Bettenhausen. 2015. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine* 48, 3 (2015), 567–572.

[28] Mike Shafto, Mike Conroy, Rich Doyle, Ed Glaessgen, Chris Kemp, Jacqueline LeMoigne, and Lui Wang. 2012. Modeling, simulation, information technology & processing roadmap. *National Aeronautics and Space Administration* (2012).

[29] Jack Stilgoe and Missy Cummings. 2020. CAN DRIVERLESS VEHICLES PROVE THEMSELVES SAFE? *Issues in Science and Technology* 37, 1 (2020), 12–14.

[30] techcrunch.com. 2021. Tesla has activated its camera. Retrieved June 23, 2021 from https://techcrunch.com/2021/05/27/tesla-has-activated-its-in-car-camera-to-monitor-drivers-using-autopilot/

[31] Mario Trapp and Daniel Schneider. 2014. Safety assurance of open adaptive systems–a survey. In *Models@ run. time*. Springer, 279–318.

[32] Danny Weyns, M Usman Iftikhar, Sam Malek, and Jesper Andersson. 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)*. IEEE, 89–98.

[33] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M Göschka. 2013. On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*. Springer, 76–107.