# Computational Intelligence for Safety Assurance of Cooperative Systems of Systems

Sohag Kabir and Yiannis Papadopoulos

***Abstract*—Cooperative Systems of Systems (CSoS) including Autonomous systems (AS), such as autonomous cars and related smart traffic infrastructures form a new technological frontier for their enormous economic and societal potentials in various domains. CSoS are often safety-critical systems, therefore, they are expected to have a high level of dependability. Due to the open and adaptive nature of the CSoS, the conventional methods used to provide safety assurance for traditional systems cannot be applied directly to these systems. Potential configurations and scenarios during the evolving operation are infinite and cannot be exhaustively analysed to provide guarantees a priori. This paper presents a novel framework for dynamic safety assurance of CSoS, which integrates design time models and runtime techniques to provide continuous assurance for a CSoS and its systems during operation.**

***Index Terms*—Autonomous Systems, Safety Assurance, Reconfigurable Systems, Computational Intelligence**

## I. Introduction

Autonomous vehicles, unmanned aerial vehicles, distributed and cloud-controlled robotics, telehealth systems, smart energy grids, the internet of things and other technologies will define the smart cities and agriculture of this century. Such systems are often CSoS and making them safe is challenging. We can identify a number of challenges they pose in safety assurance.

A first challenge is caused by the distribution and the often heterarchical organisation of systems. A heterarchy is a system or organisation where the elements are unranked and non-hierarchical or can be ranked in different ways. CSoS are inherently distributed, loosely connected and non-hierarchical. Individual systems within a CSoS are produced by different stakeholders and there is no overarching specification or authority that can guarantee their dependability when they meet in various configurations. None of the systems typically has total control and authority over others. This means that the safety of the overall system cannot be interpreted as a set of goals that are related to the behaviour of one system and to which other systems contribute. The latter is possible in more conventional systems organised as hierarchies of subsystems. It is possible, for example, to express the safety requirements of a car as a set of integrity requirements that must be achieved by its components as dictated by safety standard ISO26262. However, it is not possible to use a single reference starting point from which one could express the requirements for safety in the totality of a transport system, which is composed of connected autonomous cars and smart infrastructure. A car comprises of a hierarchy of components, while the connected transport system is a heterarchy of systems where no system has priority or absolute control when safety is concerned. This heterarchical organisation poses a major challenge for the state-of-the-art on dependability. The challenge applies both to new standards as well as cutting edge research, e.g. on model-based safety analysis, model-checking or other formal methods. Indeed, both standards and current research mostly assume a hierarchical organisation of the system, decomposition of systems into subsystems, and clear hierarchical authority of control.

A second challenge is caused by the inevitable incompleteness of dependability models anyone would attempt to do a priori at design time for a CSoS. A traffic system of connected and autonomous cars and smart infrastructures does not have a finite set of configurations. Given the unpredictable nature of CSoS and the infinity of configurations, any a priori dependability models are likely to be incomplete. Indeed all state-of-the-art dependability analysis and assurance techniques assume a bounded system; which means that full a priori certification before operation using these techniques is impossible when the CSoS is unbounded and its configurations are impossible to enumerate. These systems operate in highly dynamic and unpredictable environments, where systems collaborate with other systems adapting their behaviour in response to the change in the context of operation, workload, physical infrastructure, and network topology.

Finally, there is increased uncertainty in CSoS. It can arise from many sources: a) limited observability of the system and its environment caused by lack of sensors or failure of sensors b) unreliability of measurements c) inaccuracy, indeterminism or probabilistic nature of the inferences drawn by AI components, e.g. machine learning algorithms d) limited knowledge concerning services and dependability-relevant properties of collaboration partners in a cooperative or open system e) limited knowledge regarding trustworthiness and quality of 3rd party information.

In works on safety assurance like [1]–[3], the dynamic nature of systems is recognised. In these works, at design time several contracts or assurance cases are defined for components for different foreseeable scenarios. It is assumed that information would be available at runtime to choose particular contracts or assurances cases for components to dynamically form the assurance case for the whole system to provide a safety guarantee for a particular operational scenario. That means these approaches are mostly applicable to cases where it is possible to foresee all the potential configurations and scenarios that a CSoS can avail during its operation. This

S. Kabir is with the Department of Computer Science, University of Bradford, Bradford, BD7 1DP, UK (e-mail: s.kabir2@bradford.ac.uk) and Y. Papadopoulos is with the Department of Computer Science, University of Hull, Hull, HU6 7RX, UK (e-mail: y.i.papadopoulos@hull.ac.uk)

is certainly not possible for a typical CSoS which constrains the application of these methods.

To address the challenges of CSoS identified above, we take a new approach to dynamic safety assurance of CSoS which is being developed in the DEIS EU project (http://deis-project.eu/). The approach uses a network of intelligent safety monitoring agents to deliver safety assurance within a CSoS. An agent for a system carries information about the safety assurance of the system in the form of dependability models and uses information shared by the agents of other cooperating systems in conjunction with information from the environment to provide dependability management at runtime, e.g., event monitoring, diagnostics, certification, risk prediction and action planning. The approach aims to facilitate the development of run-time artefacts to self-certify safe operation of CSoS even if the system design evolves during operation. The approach is outlined in the paper and illustrated with an example of an autonomous production cell system.

## II. Intelligent Solution for Safety Assurance of CSoS

Fig. 1(a) shows the proposed intelligent solution framework for providing safety assurance for CSoS. The framework is formed as a distributed multi-agent system to provide safety assurance of autonomous CSoSs, where each agent will be responsible to observe and enforce the dependability of the individual physical system and the agents will cooperate to assure the safety of the whole system.

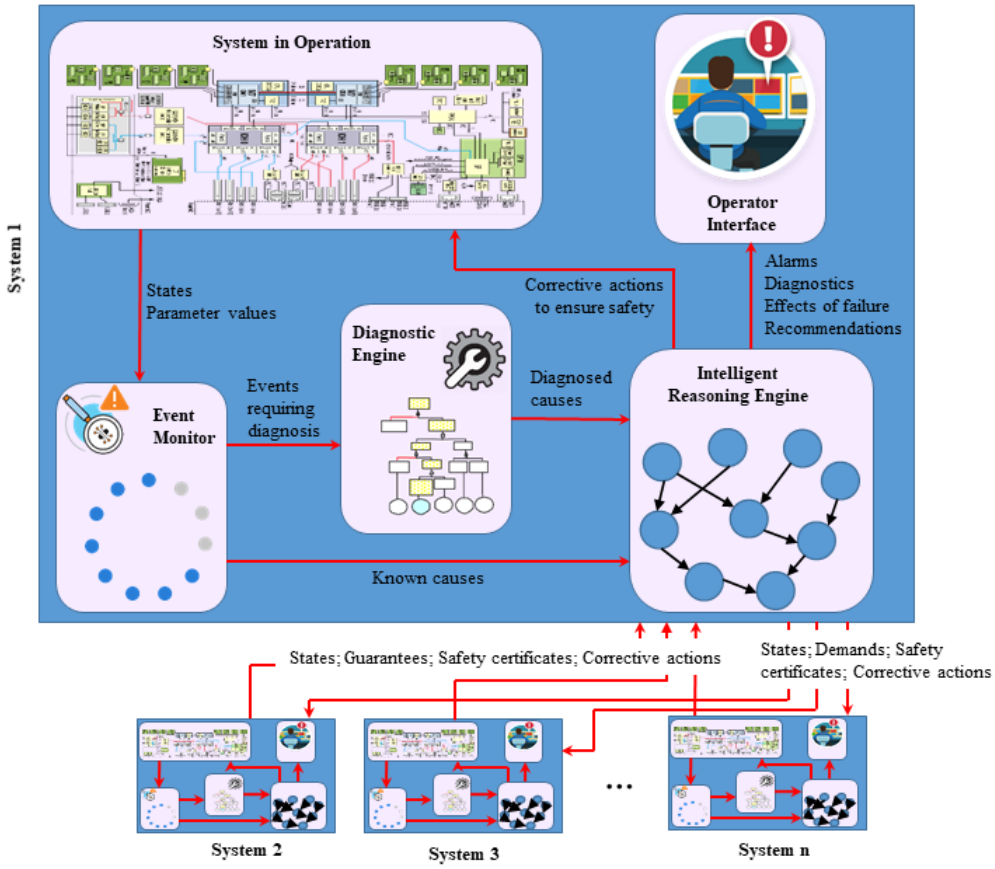
An intelligent agent, presented as intelligent reasoning engine (IRE) in Fig. 1(a), interfaces to its respective system, operators and the CSoS. As seen in the figure, the IRE of a particular system receives inputs from the system itself and also from the other systems that are part of the CSoS. The inputs received from its own system are based on real-time monitoring of the system. For monitoring, an Event Monitor is utilised, which determines the occurrence of events using real-time sensory data. These data are generated by the component or system, e.g. sensor readings, or maintenance events and can be stored for offline analysis. The modelling of event monitor may vary depending on the application area and a detailed explanation of such modelling is out of the scope of this paper. However, the use of techniques like a circular buffer or Complex Event Processing (CEP) [4] can be considered while modelling an event monitor. CEP can capture complex data streams, process a large number of events, and automatically correlate the events in the context of predefined constraints. This will help to identify events and their potential causes. If the causes of an event cannot be known with certainty, thus require diagnosis, the information will be passed to the Diagnostic Engine. For instance, while monitoring low-level events, a detected condition may reflect the symptom of failures, not underlying causes. Therefore, such symptoms are expected to be diagnosed before making a conclusion about the health of system components. Due to brevity, a detailed explanation on the modelling of fault diagnosis engine is not provided in this paper, however, interested readers can find more information on fault diagnosis in [5].

From the above discussion, we can see that with the help of an Event Monitor and Diagnostic Engine, an IRE can receive information about the state of the system and its parameters. An intelligent agent of one system can receive different information from the agents of other systems. This information may include, but not necessarily limited to, states, parameter values, dependability guarantees, safety certificates, any recommended actions etc. After processing the information received from both within the system and outside the system, an IRE can make decisions and communicate them as output to both its own system and other collaborating systems. For its own system, such outputs contain corrective measures to force the system to operate safely. Moreover, alarms are raised and recommendations are provided to the operator interface. The outputs provided to the outside systems may contain its own state information, dependability demands, safety certificates, corrective actions to assure safety, etc.

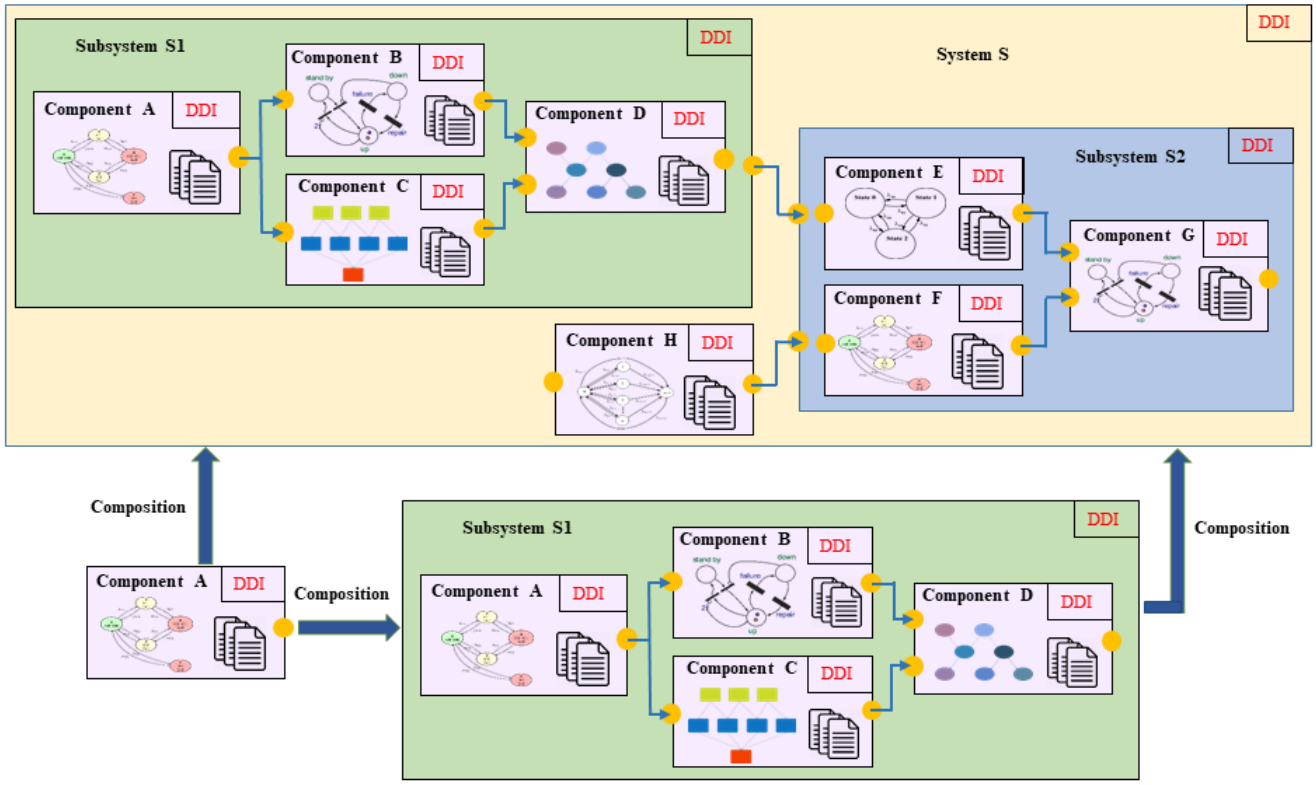
Following the primary detection and diagnosis of events, dependability-relevant events are handled by a set of model-based high-level IREs. The IREs can exist independently of each other together with their corresponding systems and assist them in dependability related tasks. However, as different systems meet or come together in a configuration, we expect that their respective IREs collectively form a parallel, distributed dependability certification system for the SoS as a whole. This distributed certification system is a multi-agent system incorporating several agents which operate locally on their models but also communicate and collaborate between them. Exchange of information about the state of other systems, their perception of the environment, and the reasoning from IREs of other systems help to reduce ambiguity and improve reasoning about dependability at the system level, e.g. to certify operations that involve many systems, assess collective risk or decide on corrective actions that may affect more than one system.

To provide safety certificates and further dependability management functions (e.g. recommended actions to avoid hazardous situations), the IREs operate on knowledge graph, which is an ontology that integrates metadata in a machine-interpretable representation. We incorporate appropriate IRE models into the knowledge graph to enable the executing of dependability algorithms at runtime. For this purpose, we use the concept of Digital Dependability Identities (DDIs) [6], which is a key innovation of the **H2020 DEIS project**. Due to brevity, full implementation details of DDI are omitted in this paper, however, interested readers can fetch the full implementation details from the publicly available GitHub repository of the DEIS project[1]. A DDI of a component or a system contains all the information required to uniquely define the dependability properties of the system or the component. The information can be captured in textual format and/or by using one or more safety artefacts such as in the form of fault trees, Conditional Safety Certificate (ConSerts), Bayesian Networks (BN), Markov chains, Petri nets, etc. Using these models, the key attributes that define the systems' or components' dependability behaviour are captured, for instance, in the form

[1]https://github.com/DEIS-Project-EU/




(a) Intelligent Solution Framework for Safety Assurance of Autonomous Systems of Systems




(b) Compositional formation of an Intelligent Reasoning model for a system based on DDIs

Fig. 1. The proposed intelligent solution framework

of faults and potential fault propagation models. Additionally, requirements on how the component/subsystem interacts with other components/subsystems of the CSoS in a dependable way is captured in terms of the level of trust and assurance.

Note that, DDIs span over the lifecycle of a CSoS. They are produced during the design, issued during releasing the component/system and continuously maintained/adapted throughout the system lifetime. DDI models that provide the basis for runtime reasoning are modular and composable (see Fig. 1(b)). As seen in Fig. 1(b), DDIs of multiple components can be composed together to form the DDI of a subsystem. The DDIs of subsystems and other components can be integrated iteratively to form the DDI of the whole system. This integration requires algorithms and tools which include mechanisms for abstraction, modularisation, information hiding and formalisation. Currently, we are working towards developing a tool-supported process where the production of IREs as DDI models will be semi-automatic, where a component integrator would be able to compose component-level IREs into system reasoning models. IRE models will be thoroughly tested before deployment to systems, e.g. using appropriate testbeds and simulation. Once sufficient testing has been done, the models will be deployed on systems, each system carrying an intelligent agent responsible for system monitoring and managing its dependability.

## III. Illustrative Example

To demonstrate the proposed approach, we use the example of an autonomous production cell system presented in [7] and shown in Fig. 2(a). The function of the production system is to take a workpiece, drill a hole in it, insert a screw into this hole, and then tighten the screw. To achieve this functionality, the system uses three autonomous robots, which are connected with autonomous transportation carts. With the help of three different tools, each of the robots can perform three tasks: drilling, inserting and tightening. However, they can only perform one task at a time and if instructed they can reconfigure themselves to perform a different task. Note that switching from one mode of operation to another mode of operation, i.e., switching tools require some time. A workpiece must be processed in the order of Drill $\rightarrow$ Insert $\rightarrow$ Tighten, i.e., DIT. Although, a single robot can perform all three tasks, to meet the timing constraint a single robot cannot be used to perform all three tasks through multiple reconfigurations. Therefore, the workpiece should be processed by all three robots in order and the carts transport the workpiece to the robots according to the planned operation (see Fig. 2(a)).

In the conditions of break-down of one or more tools, a particular configuration of the whole system may not be able to provide the DIT processing of a workpiece. In such cases, a system reconfiguration may be possible to restore the DIT processing of the workpiece. For instance, in Fig. 2(b), a broken drill at Robot 1 seized its intended functionality to drill a hole on the workpiece, thus disrupts the DIT processing of the workpiece. This disruption is temporary in the sense that another Robot can be reconfigured to take over the drilling operation and the Robot 1 can be configured to perform another task than drilling. Fig. 2(c) shows the restoration of the DIT processing of the workpiece by switching tasks between Robot 1 and Robot 3. Note that, given the available functionality of each robot, many configurations can be possible to keep the system operational. However, the exact information about these configurations is not known at design time, i.e., how the robots will be configured in the future cannot be known at design time. Forming a new configuration to reassign tasks to robots and re-routing the carts accordingly is the responsibility of a controller. How the controller performs these tasks is out of the scope of this paper.

In this paper, we aim to demonstrate that the intelligent solution proposed in Section II can help to provide safety assurance of the above autonomous system even in the absence of the prior knowledge about any particular configuration the system will avail during its operation in the future. To do this, similar to [7] we define the correct functioning of the system as an invariant predicate as "a workpiece can be processed by the system in DIT order". Any violation of this variant can be considered as the violation of the safety requirement of the system operation. To detect such violations, we need to form an intelligent reasoning engine that can utilise evidences collected during the operation of the system to tell us whether the above-mentioned invariant can be satisfied by a particular system configuration at any point in time. If the invariant is not satisfied with the current system configuration, the intelligent reasoning engine should be able to tell us whether it is possible to satisfy the invariant via system reconfiguration. In the worst case, we will know that no reconfiguration is possible to satisfy the invariant, thus system operation should be halted.

To illustrate the idea of an intelligent reasoning engine in the form of DDIs, we formed the model shown in Fig. 3. For the illustration purpose, we use a Bayesian network as the model to represent DDIs. Due to space limitation, we only show the use of a BN model, however, as mentioned earlier, different other models such as ConSert and fault tree can be used for this purpose. Interested readers can find more information on such alternative modelling and applications of DDIs in different areas on the DEIS project's website[2]. From Fig. 3 it can be seen that the DDI of the whole production system is formed in a compositional fashion. Together with its own elements, it composed the DDIs of the three robots named as "DDI Robot X", where X=1, 2, 3. As the three robots are of the same specifications, their DDIs are modelled as the same. The DDI of each robot can communicate the functional status of the respective robot. For instance, at any point in time, a robot can be in one of the eight states: NoFunctionality (completely non-functional), DrillOnly (can perform drilling only), InsertOnly (can perform inserting only), TightOnly (can perform tightening only), DrillAndInsert (can perform drilling and inserting), DrillAndTight (can perform drilling and tightening), InsertAndTight (can perform inserting and tightening), AllFunctionality (fully functional). Within the DDI of the production system, the nodes ConfigurationOfRobot1, ConfigurationOfRobot2, and ConfigurationOfRobot3 represent the functionality expected from the Robots 1, 2, and 3 respectively

[2]http://www.deis-project.eu/dissemination/

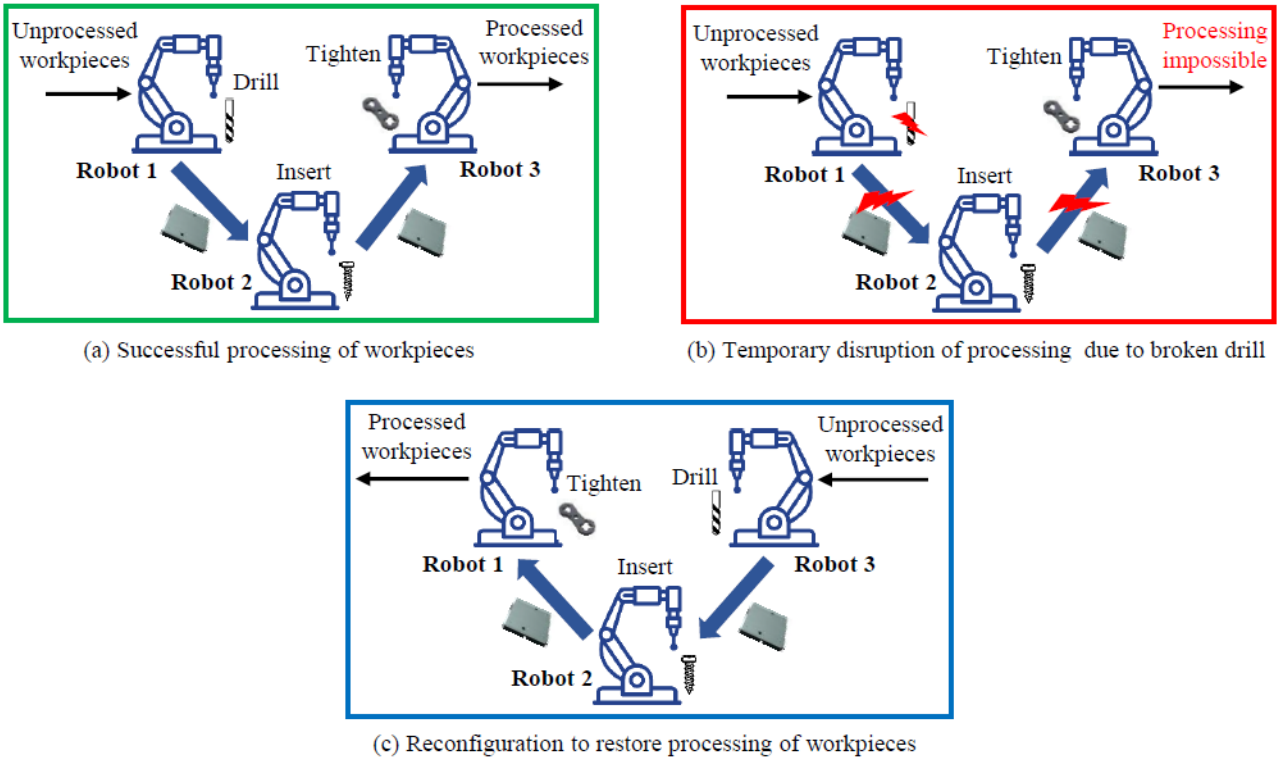



Fig. 2. Production automation system with three autonomous robots

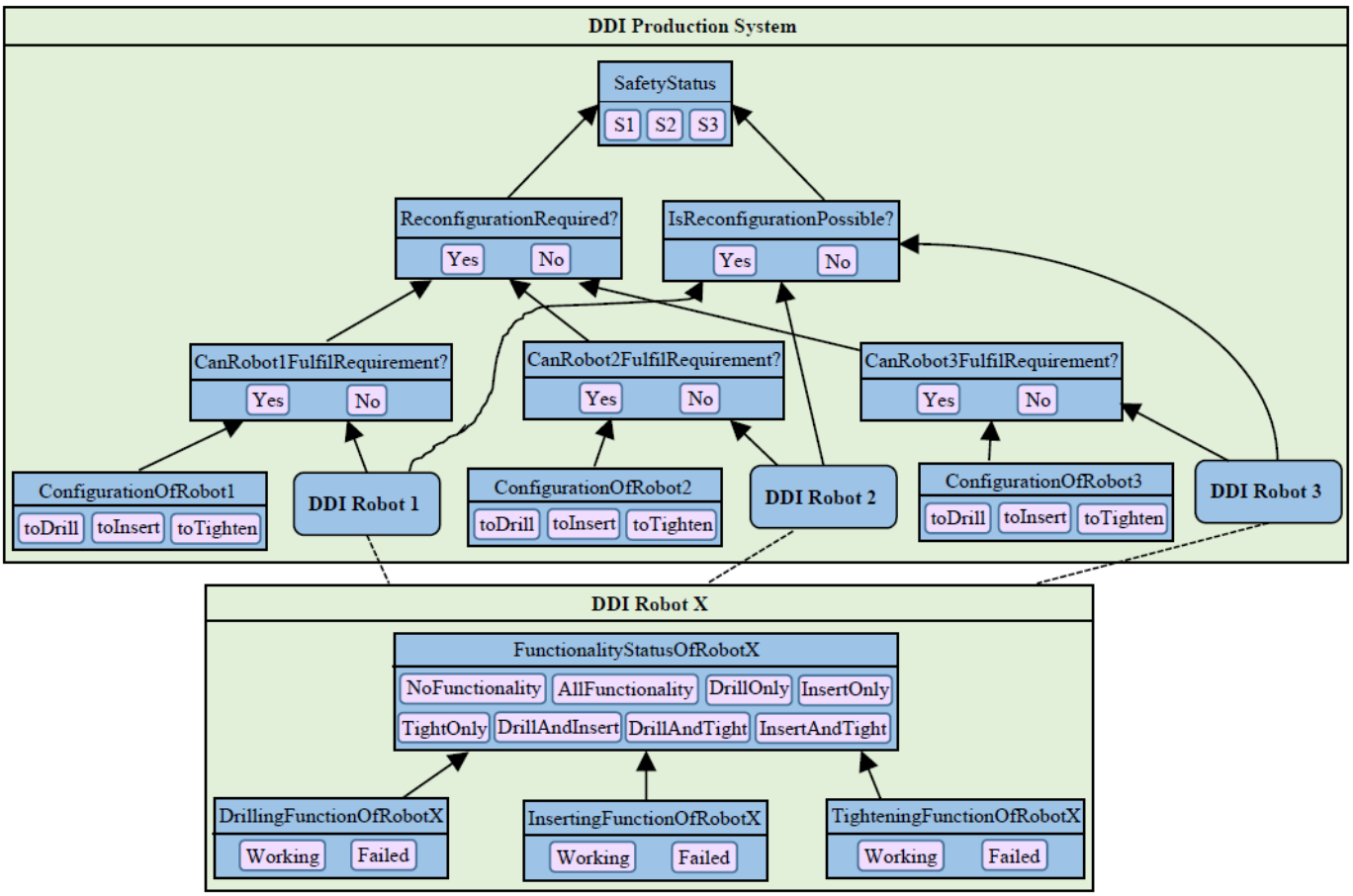



Fig. 3. Intelligent reasoning engine for the autonomous system as DDIs

in the current system configuration. The controller, which is not shown in this paper, decides the expected functionality of the robots, i.e. the value of these nodes. Whether a robot can fulfil its operational requirement in the current operational state (represented as nodes CanRobotXFulfilRequirement? in the model) is determined based on its operational state (obtained from its DDI) and the expected function of the robot. The node SafetyStatus within the DDI of the production system can tell us the status of the system with regards to the invariant defined earlier. This node has three values: S1, S2, and S3. S1 means that the invariant can be satisfied with the current configuration, i.e., safety is guaranteed in this state. S2 means that invariant is not satisfied in this current configuration, however, the invariant can be satisfied by reconfiguring the system. That means the controller has to perform the required reconfiguration to assure safety in this state. S3 means that invariant is not satisfied in this current configuration and no other configuration can be formed to satisfy the invariant. Therefore, system operation should be stopped immediately.

To test whether the intelligent reasoning engine can detect different scenarios based on the inputs received from the robots during the operation, we generated different hypothetical scenarios and evaluated the reasoning capability of the model. Due to limited space, in Table I, we reported the results of six of those tests (Test 1 to Test 6). For instance, Test 1 depicts the scenario shown in Fig. 2(a), where all robots are fully functional and Robot 1, 2, and 3 are assigned to drill, insert, and tighten respectively. Therefore, as expected the model determined the state of the system as S1. In Test 2, although all the robots are partially functional, still the system is determined to be in state S1 because the available functionalities of the robots are sufficient to meet their requirements. In Test 3 and 5, the reasoning engine determines the system to be in S2, meaning a reconfiguration is required to satisfy the safety invariant. In Test 3, Robot 1 and 3 fail to satisfy their requirement and Robot 1 fails to satisfy its requirement in Test 5. System state S3 has been determined in Tests 4 and 6. In Test 4, a common single available functionality (TightOnly) for Robot 1 and 3 push the system to state S3, where further reconfiguration is not possible to satisfy the invariant. This is because either of Robot 1 or Robot 3 has to be able to perform a different function than tightening. In Test 6, the reason behind the system is determined to be in state S3 is the non-functionality of Robot 2.

TABLE I
RESULTS OF THE TESTING OF THE REASONING ENGINE OF FIG. 3

| **Parameters** | **Test 1** | **Test 2** | **Test 3** | **Test 4** | **Test 5** | **Test 6** |
|---|---|---|---|---|---|---|
| FunctionalityStatusOfRobot1 | AllFunctionality | DrillAndTight | DrillAndTight | TightOnly | DrillAndInsert | InsertAndTight |
| FunctionalityStatusOfRobot2 | AllFunctionality | InsertOnly | AllFunctionality | DrillAndInsert | DrillAndTight | NoFunctionality |
| FunctionalityStatusOfRobot3 | AllFunctionality | InsertAndTight | TightOnly | TightOnly | InsertOnly | AllFunctionality |
| ConfigurationOfRobot1 | toDrill | toDrill | toInsert | toInsert | toTighten | toTighten |
| ConfigurationOfRobot2 | toInsert | toInsert | toTighten | toDrill | toDrill | toInsert |
| ConfigurationOfRobot3 | toTighten | toTighten | toDrill | toTighten | toInsert | toDrill |
| **SafetyStatus** | **S1** | **S1** | **S2** | **S3** | **S2** | **S3** |

## IV. RELEVANT WORK AND COMPARISON WITH STATE-OF-THE-ART

There has been recent interest in the concept of dynamic assurance cases that proactively compute the confidence in, and update the reasoning about, the dependability of ongoing operations. Current research on dynamic assurance has considered the need for greater automation [8]; mathematical formality [9]; and quantification of confidence based on real monitored data [10]. This has been combined with modular approaches to reasoning about safety and security of cooperating systems in applications such as farming and automotive [11]

and industry 4.0 [12]. Rushby introduced some initial ideas for runtime certification based on formal analyses, enabling the verification of component runtime behaviour according to its specification [13].

A more practical refinement of this general idea is the concept of the ConSert which facilitates the modular definition of safety certificates using a contract-like approach for components and systems [11]. ConSerts have a runtime representation, enabling their use during in-the-field execution and dynamic reconfiguration of systems at runtime. ConSerts do not describe fixed contracts, but support variability. Depending on which runtime demands can be fulfilled, the guarantees offered by the system can be updated and adapted accordingly. This provides flexibility during system integration, as it is very unlikely that independently-developed components provide the exact safety properties fitting to static assumptions. ConSerts have been applied in different settings (e.g. industrial prototypes) and they have been applied and adapted in research projects such as EMC2 [3]. ConSerts have been integrated within the concept of the DDI presented in this paper.

Dynamic risk management is a rather new concept. Dynamic dependability management capabilities can be used as a basis for dynamic monitoring. In the automotive domain, there are approaches potentially allowing dynamic risk management. Essential ingredients are models of system capabilities and the dependability-relevant environment [14]; and scenarios combining the former at a specific point in time, where risk shall be assessed dynamically. Risk control requires models about safety capability variants [15] that can be selected based on the assessed risk. Other recent approaches to dependability of CSoS use run-time application of formal methods [16] and model checking [17]. Johnson *et al.* [18] proposed a multi-agent framework for the dependable adaptation of evolving system architectures. This approach used an Agent Verification Engine, which can construct evolvable Belief-Desire-Intention (BDI) agents that verify a system architecture when something changes in the architecture. Getir *et al.* [19] provided a set of model transformation rules for ensuring the co-evolution of software architecture and fault tree models. This approach has been applied to factory automation examples and for a limited number of known evolution scenarios. Similar to ConSerts, this approach can provide safety assurance for reconfigurable systems but under the assumption that there are a known number of system configurations. Good overviews of dynamic assurance approaches and the many open research challenges are given by [16].

Overall, work on dynamic assurance cases has addressed to some extent the challenges of openness. We draw for this work, ConSerts in particular. However, we also look to create a flexible framework which can employ models and techniques to address the challenges of uncertainty, unpredictability and the need to repair dependability models that in practice may deviate from reality in a complex CSoS.

Our multi-agent system of DDIs facilitates dynamic dependable integration and operation of systems into intelligent "systems of systems". We currently incorporate uncertainty in models including concepts from relevant literature: multi-valued logics and Bayesian inference. This is early work but could lead to significant step forwards; to our knowledge, no other dynamic assurance method has addressed the important issue of uncertainty at runtime certification.

**For additional reading:**

1. E.E. Alves, D. Bhatt, B. Hall, K. Driscoll, A. Murugesan and J. Rushby, "Considerations in assuring safety of increasingly autonomous systems," NASA Langley Research Center, Hampton, Virginia, Tech. Rep. No. NASA/CR–2018-220080, 2018.

2. J.A. McDermid, Y. Jia, and I. Habli, "Towards a Framework for Safety Assurance of Autonomous Systems," in Artificial Intelligence Safety, 2019, pp. 1-7.

3. S. Müller, and P. Liggesmeyer, "Safety assurance for emergent collaboration of open distributed systems," in International Symposium on Software Reliability Engineering Workshops (ISSREW), IEEE, 2016, pp. 249-256.

4. J. Swanson, M.B. Cohen, M.B. Dwyer, B.J. Garvin, and J., Firestone, "Beyond the rainbow: Self-adaptive failure avoidance in configurable systems," in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 377-388.

5. E. Denney, G. Pai, and I. Habli, "Dynamic safety cases for through-life safety assurance," in IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2. IEEE, 2015, pp. 587–590.

6. D. Schneider and M. Trapp, "B-space: dynamic management and assurance of open systems of systems," Journal of Internet Services and Applications, vol. 9, no. 1, pp. 1–16, 2018.

7. R. De Lemos, D. Garlan, C. Ghezzi, H. Giese, J. Andersson, M. Litoiu, B. Schmerl, D. Weyns, L. Baresi, N. Bencomo et al., "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," in Software Engineering for Self-Adaptive Systems III. Assurances. Springer, 2017, pp. 3–30.

The highly dynamic nature of CSoS means that any models used to observe and enforce dependability are likely to deviate from reality. To address this problem, DDI agents self-monitor at run-time the integrity of their models in terms of correctness and completeness. When problems are detected then the limitations of runtime certification can become evident to those who oversee the systems. Runtime certification can be temporarily suspended in full knowledge of the stakeholders involved. In addition, we explore techniques for fixing DDI models at runtime, e.g. by correcting or augmenting them with new causal relationships and paths, using, for example, machine learning to establish those new relationships. The main goal here is the ability for non-monotonic reasoning about certification or safety. In this mode, the system could recover from reasoning failure enabling a DDI-based certifying system to continue to operate in degraded mode when this is necessary, still providing advice to systems and operators.

Early work on model-repair has been presented in [20]. We are not aware of any dynamic assurance technique that offers such capabilities.

## V. Conclusion

The unstoppable development of CSoSs has the potential to provide significant benefits to society by facilitating productivity enhancements and creating new markets. While such developments have created huge expectations and investments in AI, they give rise to fundamental challenges to safety assurance of such systems. CSoSs' safety assurance is a relatively new research area and has gained increasing attention from industry, academia, governments and regulators around the world. To provide safety assurance for the CSoSs while taking into account the challenges posed by the evolving and black-box nature of such systems, researchers are putting their efforts to develop approaches to shift some of the design time assurance activities to the runtime. The framework presented in this paper is one such effort to provide an intelligent solution for the safety assurances of CSoSs. The approach presented in this paper proposes to deploy intelligent agents in the systems to monitor different parameters and the behaviours of the systems. The agents of different collaborating systems are designed to share their knowledge about the parameters related to the dependable operation/collaboration of the systems. This knowledge is utilised within the intelligent reasoning engines to perceive the safety status of the CSoSs. Based on the perceived safety state of a system, different level of variable guarantees can be provided. Moreover, the recommended actions can be suggested to improve the overall safety of the system.

## Acknowledgements

This work was supported by the Dependability Engineering Innovation for Cyber Physical Systems (DEIS) H2020 Project under Grant 732242.

## References

[1] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly, "Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1039–1069, 2018.

[2] S. Kabir, I. Sorokos, K. Aslansefat, Y. Papadopoulos, Y. Gheraibia, J. Reich, M. Saimler, and R. Wei, "A Runtime Safety Analysis Concept for Open Adaptive Systems," in *International Symposium on Model-Based Safety and Assessment*. Springer, 2019, pp. 332–346.

[3] T. Amorim, D. Ratasich, G. Macher, A. Ruiz, D. Schneider, M. Driussi, and R. Grosu, "Runtime safety assurance for adaptive cyber-physical systems: ConSerts M and ontology-based runtime reconfiguration applied to an automotive case study," in *Solutions for Cyber-Physical Systems Ubiquity*, 2018, pp. 137–168.

[4] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," in *Proceedings of the ACM SIGMOD International Conference on Management of data*. New York, New York, USA: ACM Press, 2006, pp. 407–418.

[5] J. Chen and R. J. Patton, *Robust model-based fault diagnosis for dynamic systems*, 3rd ed. Springer Science & Business Media, 2012.

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

[7] M. Güdemann, F. Ortmeier, and W. Reif, "Safety and Dependability Analysis of Self-Adaptive Systems," in *Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (isola 2006)*. IEEE, 2006, pp. 177–184.

[8] E. Denney and G. Pai, "Tool support for assurance case development," *Automated Software Engineering*, vol. 25, no. 3, pp. 435–499, 2018.

[9] J. Rushby, "Trustworthy self-integrating systems," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2016, pp. 19–29.

[10] P. J. Graydon and C. M. Holloway, "An investigation of proposed techniques for quantifying confidence in assurance arguments," *Safety science*, vol. 92, pp. 53–65, 2017.

[11] D. Schneider and M. Trapp, "Conditional Safety Certification of Open Adaptive Systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 8, no. 2, pp. 1–20, 2013.

[12] O. Jaradat, I. Sljivo, I. Habli, and R. Hawkins, "Challenges of safety assurance for industry 4.0," in *13th European Dependable Computing Conference (EDCC)*. IEEE, 2017, pp. 103–106.

[13] J. Rushby, "Runtime certification," in *International Workshop on Runtime Verification*. Springer, 2008, pp. 21–35.

[14] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 982–988.

[15] I. Colwell, B. Phan, S. Saleem, R. Salay, and K. Czarnecki, "An automated vehicle safety concept based on runtime restriction of the operational design domain," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1910–1917.

[16] B. Cheng, K. Eder, M. Gogolla, L. Grunske, M. Litoiu, H. Müller, P. Pelliccione, A. Perini, N. Qureshi, B. Rumpe, and D. Schneider, "Using models at runtime to address assurance for self-adaptive systems," in *Models@ run. time*. Springer, 2014, pp. 101–136.

[17] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic QoS Management and Optimization in Service-Based Systems," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 387–409, 2011.

[18] K. Johnson, R. Sinha, R. Calinescu, and J. Ruan, "A Multi-agent Framework for Dependable Adaptation of Evolving System Architectures," in *41st Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, 2015, pp. 159–166.

[19] S. Getir, L. Grunske, A. van Hoorn, T. Kehrer, Y. Noller, and M. Tichy, "Supporting semi-automatic co-evolution of architecture and fault tree models," *Journal of Systems and Software*, vol. 142, pp. 115–135, 2018.

[20] Y. Gheraibia, S. Kabir, K. Aslansefat, I. Sorokos, and Y. Papadopoulos, "Safety + AI: A Novel Approach to Update Safety Models Using Artificial Intelligence," *IEEE Access*, vol. 7, pp. 135 855–135 869, 2019.

**Sohag Kabir** received the Ph.D. degree in computer science and the M.Sc. degree in embedded systems from the University of Hull, UK, in 2016 and 2012, respectively. He is currently working as a Lecturer (Assistant Professor) in the Department of Computer Science at the University of Bradford, UK. Prior to that, he was a research associate in the Dependable Intelligent Systems (DEIS) Research Group at the University of Hull. He has worked in EU projects on safety, including MAENAD and DEIS. His research interests include model-based safety assessment, probabilistic risk and safety analysis, fault tolerant computing, and stochastic modelling and analysis.

**Yiannis Papadopoulos** is a professor and leader of the Dependable Intelligent Systems research group at the University of Hull. He has pioneered work on model-based dependability assessment and evolutionary optimisation of complex engineering systems known as Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS). He co-authored EAST-ADL, an emerging automotive architecture description language working with Volvo, Honda, Continental, Honeywell, and DNV-GL, among others. He is actively involved in two technical committees of IFAC (TC 1.3 & 5.1). He is also working on new metaheuristics inspired by the hunting behaviour of penguins and developing technologies for self-certification of cyber-physical and autonomous systems. He is interested in digital art and various aspects of philosophy and its interactions with science.