
SESAME
SECURE AND SAFE MULTI-ROBOT SYSTEMS

**Project Number 101017258**

# D7.1 Runtime Safety and Security Concept - EDDI Runtime Model Specification

**Version 1.0**
**30 June 2022**
**Final**

**Public Distribution**

## Fraunhofer IESE and University of Hull

**Project Partners:** **Aero41, ATB, AVL, Bonn-Rhein-Sieg University, Cyprus Civil Defence, Domaine Kox, FORTH, Fraunhofer IESE, KIOS, KUKA Assembly & Test, Locomotec, Luxsense, The Open Group, Technology Transfer Systems, University of Hull, University of Luxembourg, University of York**

# PROJECT PARTNER CONTACT INFORMATION


| | |
|---|---|
| **Aero41**<br>Frédéric Hemmeler<br>Chemin de Mornex 3<br>1003 Lausanne<br>Switzerland<br>E-mail: frederic.hemmeler@aero41.ch | **ATB**<br>Sebastian Scholze<br>Wiener Strasse 1<br>28359 Bremen<br>Germany<br>E-mail: scholze@atb-bremen.de |
| **AVL**<br>Martin Weinzerl<br>Hans-List-Platz 1<br>8020 Graz<br>Austria<br>E-mail: martin.weinzerl@avl.com | **Bonn-Rhein-Sieg University**<br>Nico Hochgeschwender<br>Grantham-Allee 20<br>53757 Sankt Augustin<br>Germany<br>E-mail: nico.hochgeschwender@h-brs.de |
| **Cyprus Civil Defence**<br>Eftychia Stokkou<br>Cyprus Ministry of Interior<br>1453 Lefkosia<br>Cyprus<br>E-mail: estokkou@cd.moi.gov.cy | **Domaine Kox**<br>Corinne Kox<br>6 Rue des Prés<br>5561 Remich<br>Luxembourg<br>E-mail: corinne@domainekox.lu |
| **FORTH**<br>Sotiris Ioannidis<br>N Plastira Str 100<br>70013 Heraklion<br>Greece<br>E-mail: sotiris@ics.forth.gr | **Fraunhofer IESE**<br>Daniel Schneider<br>Fraunhofer-Platz 1<br>67663 Kaiserslautern<br>Germany<br>E-mail: daniel.schneider@iese.fraunhofer.de |
| **KIOS**<br>Maria Michael<br>1 Panepistimiou Avenue<br>2109 Aglatzia, Nicosia<br>Cyprus<br>E-mail: mmichael@ucy.ac.cy | **KUKA Assembly & Test**<br>Michael Laackmann<br>Uhthoffstrasse 1<br>28757 Bremen<br>Germany<br>E-mail: michael.laackmann@kuka.com |
| **Locomotec**<br>Sebastian Blumenthal<br>Bergiusstrasse 15<br>86199 Augsburg<br>Germany<br>E-mail: blumenthal@locomotec.com | **Luxsense**<br>Gilles Rock<br>85-87 Parc d'Activités<br>8303 Luxembourg<br>Luxembourg<br>E-mail: gilles.rock@luxsense.lu |
| **The Open Group**<br>Scott Hansen<br>Rond Point Schuman 6, 5th Floor<br>1040 Brussels<br>Belgium<br>E-mail: s.hansen@opengroup.org | **Technology Transfer Systems**<br>Paolo Pedrazzoli<br>Via Francesco d'Ovidio, 3<br>20131 Milano<br>Italy<br>E-mail: pedrazzoli@ttsnetwork.com |
| **University of Hull**<br>Yiannis Papadopoulos<br>Cottingham Road<br>Hull HU6 7TQ<br>United Kingdom<br>E-mail: y.i.papadopoulos@hull.ac.uk | **University of Luxembourg**<br>Miguel Olivares Mendez<br>2 Avenue de l'Universite<br>4365 Esch-sur-Alzette<br>Luxembourg<br>E-mail: miguel.olivaresmendez@uni.lu |
| **University of York**<br>Simos Gerasimou & Nicholas Matragkas<br>Deramore Lane<br>York YO10 5GH<br>United Kingdom<br>E-mail: simos.gerasimou@york.ac.uk<br>nicholas.matragkas@york.ac.uk | |

# DOCUMENT CONTROL

| Version | Status | Date |
|---|---|---|
| 0.1 | Initial outline | 10 May 2022 |
| 0.2 | Second version; Added HULL and IESE uncertainty monitoring content + dynamic event monitoring | 10 June 2022 |
| 0.3 | More content, restructured sections | 25 June 2022 |
| 0.4 | Final version for QA in project | 26 June 2022 |
| 0.5 | Review from FORTH | 28 June 2022 |
| 0.6 | Changes from FORTH review complete | 28 June 2022 |
| 0.7 | Comments and change proposals from LU review integrated | 29 June 2022 |
| 1.0 | Final version | 30 June 2022 |

# TABLE OF CONTENTS

**1. Introduction ..... 1**
**2. Runtime Dependability Monitoring Architecture With EDDI ..... 4**
**3. Dynamic Safety Capability Assessment ..... 7**
*3.1 Conditional Safety Certificates ..... 8*
*3.2 Application in SESAME ..... 9*
*3.3 Method and Algorithm ..... 10*
*3.4 Example ..... 10*
**4. Dynamic Reliability Assessment ..... 12**
**5. Dynamic Risk Assessment ..... 17**
*5.1 State of the Art ..... 17*
*5.2 Situation-Aware Dynamic Risk Assessment ..... 20*
*5.3 Example of SINADRA Model in SESAME Use Case ..... 23*
**6. Dynamic Perception Uncertainty Monitoring ..... 26**
*6.1 Background ..... 26*
*6.2 Uncertainty monitoring ..... 27*
*6.3 Uncertainty monitoring EDDI ..... 28*
*6.4 Application in SESAME ..... 28*
**7. Dynamic Event Monitoring ..... 29**
*7.1 EDDI Event Framework context ..... 30*
*7.2 Event monitoring ..... 31*
*7.3 Events ..... 32*
**8. Dynamic Security Management ..... 36**
**9. Conclusions ..... 37**
**10. References ..... 39**

# TABLE OF FIGURES

Figure 1 Runtime EDDI Functionality ..... 4
Figure 2 From single-system EDDI to MRS EDDI ..... 5
Figure 3 Relationship of D7.1 to other SESAME deliverables ..... 7
Figure 4 ConSert Composition Conceptual Overview ..... 8
Figure 5 Relation between safety concept and ConSert for an open adaptive system ..... 9
Figure 6 Example ConSert for the LOCOMOTEC use case ..... 11
Figure 7 The Proposed Fault Tree of a typical UAV ..... 14
Figure 8 Small FTA of a UAV with only three complex basic events – I) Processor Failure, II) Battery Failure and III) Propulsion System Failure -- considering failure symptoms and three different types of reconfigurations for propulsion system failure. ..... 15
Figure 9 Overall view on combining real-time monitoring and diagnosis with Fault Tree Analysis ..... 16
Figure 10 A sample view of SafeDrones integration with KIOS software. ..... 16
Figure 11 Difference between safety capability/reliability assessment and risk assessment ..... 17
Figure 12 Dynamic Risk Assessment Conceptual Overview [17] ..... 19
Figure 13 Components of the SINADRA Framework ..... 21
Figure 14 Example Qualitative Risk Model (Excerpt) ..... 22
Figure 15 Example Bayesian network for autonomous shuttle - pedestrian controllability estimation (Excerpt) ..... 23
Figure 16 Example qualitative risk model for human UV-C overexposure risks in disinfection use case ..... 24
Figure 17 Experimental determination of the influence of the robot-person distance on UV-C intensity ..... 25
Figure 18 Onion Layer model for uncertainty in ML components as described in [27] ..... 26
Figure 19 Uncertainty wrapper overview ..... 27
Figure 20 Application of SafeML ..... 27
Figure 21 The use of SafeML on LOCOMOTEC use case of SESAME ..... 29
Figure 22 The basic Executable Digital Dependability Identity architecture ..... 30
Figure 23 The Event-Action cycle ..... 32
Figure 24 Example workflow of IDS as part of runtime EDDIs ..... 37

# EXECUTIVE SUMMARY

Multi-Robot Systems (MRS) can be considered as a major driver of innovation in terms of engineering challenge, dynamicity of operation, and runtime collaboration. In the concepts presented in this deliverable, we make our preliminary attempt to address dependability through Executable Digital Dependability Identity (EDDI) runtime components.

Runtime EDDIs are responsible for attaching upon an MRS application's constituent robots, and enhancing both their awareness of their operational context in terms of dependability, as well as recommending behavioural adaptation when deemed relevant.

We should note that the specific components we discuss in the individual sections of the deliverable are hardly the only viable solution in terms of any MRS application, as there is significant existing and ongoing work that offers alternative components. However, the components included here present what we consider to be a composition that is appropriate for the MRS applications in the scope of the SESAME project.

That being said, the concepts that the runtime EDDI components realize are broader, and we also believe that they represent fundamental building blocks from which other MRS applications can compose solutions of equivalent effectiveness.

Summarizing the contents that follow in the remainder of the deliverable, we discuss further what the motivation for MRS applications and roles of runtime EDDI components are, and identify needs and solutions for runtime assessment of safety capabilities, reliability, situational risk, perception uncertainty, as well as runtime dependability event monitoring, and security management.

# LIST OF ABBREVIATIONS

| | | | |
|---|---|---|---|
| AI | Artificial Intelligence | MAS | Multi-Agent System |
| DDI | Digital Dependability Identity | ML | Machine Learning |
| ECDF | Empirical Cumulative Distribution Function | MRS | Multi-Robot System |
| EDDI | Executable Digital Dependability Identity | MTTF | Mean Time To Failure (also, MTBF: Mean Time Before Failure) |
| SINADRA | Situation-Aware Dynamic Risk Assessment | ODE | Open Dependability Exchange |
| SMILE | Statistical Model-agnostic Interpretability with Local Explanations | UAS | Unmanned Aircraft/Airborne System |
| FTA | Fault Tree Analysis | UAV | Unmanned Aerial Vehicle |
| DEIS | Dependability Engineering Innovation for Cyber-Physical Systems | UW | Uncertainty Wrapper |
| ConSerts | Conditional Safety Certificates | LIDAR | LIght Detection And Ranging |
| CBE | Complex Basic Event | RtE | Runtime Evidence |
| SESAME | Secure and Safe Multi-Robot Systems | EBNF | Extended Backus-Naur form |
| MAPE-K | Monitor-Analyze-Plan-Execute-Knowledge | IDS | Intrusion Detection System |

# 1. INTRODUCTION

The context introduced by the SESAME project — with its emphasis on multi-robot systems (MRS) — highlights several of the major challenges of modern safety engineering. While robots are not a new invention in themselves, the way they operate in increasingly interconnected, adaptive, and AI-driven ways poses distinct difficulties for engineering safe and secure systems and at the same time preserve a maximum level of performance. Conversely, Unmanned Aircraft/Airborne System (UAS)-like multirotor drones *are* a relatively new invention, and sufficiently different from manned aerial systems to demand a novel approach — especially when multiple units are operating together as part of a cohesive whole.

One of the most perplexing difficulties facing dependability engineers is the fact that only a portion of such systems and their environment are known at design time. This does not apply when considering systems which can learn and adapt during their operational lifetime. Neither does it apply when systems adapt their behaviour dynamically (and autonomously) at runtime in response to its environment or to cooperate with other systems. Thus, a simple, static, design-time analysis and an according system design is no longer sufficient by itself. Additionally, such systems need to understand the environmental context in a more sophisticated way to come to safe and performant adaptation decisions. That extends the required scope of the safety analysis beyond simple internal failure propagation to include interaction between the system and its environment, particularly in the case of safety issues that originate from the dynamic environment rather than the system itself.

These challenges can be broadly separated into three main categories:

- **Complexity** of dynamic systems in a dynamic environment
- **Intelligence** of systems driven by AI & machine learning
- **Autonomy** and unpredictability of open & distributed systems

One of the goals of the SESAME project is to develop a concept, methodology, and supporting tools to address these challenges. Central to this aim is the idea of an Executable Digital Dependability Identity (EDDI), which is a set of dependability-relevant models synthesised during design-time analysis, but is transformed into executable pieces of code that operate alongside the host system at runtime to monitor and respond to safety-relevant changes (e.g. failures or context changes) during operation.

MRS can be seen as cyber-physical systems of systems, which need to operate in highly dynamic environments to achieve their goals. From the SESAME use cases, this is already visible, as seen in the examples below:

- In the battery innovation centre use case, an MRS needs to react to thermal events and potentially human error (e.g. workers mistakenly obstructing MRS movement paths and/or incorrect task performance).

- In the hospital disinfection case, an MRS needs to once again react to unexpected human interaction, as well as to dynamic lighting conditions, which might impede accurate person detection.

- In the viticulture case, an MRS needs to adapt their movement and spraying patterns to changing weather conditions, as well as to the presence, absence, and movement of people in the vineyard.

- In the drone emergency surveillance case, an MRS needs to respond to potentially extreme environmental conditions, survey an area that might still be changing (falling debris, explosions) and potentially detect people in need of assistance (those who may or may not be immobile).

- In the factory plant case, an MRS needs to adjust to regular movement and interaction of human workers on the factory floor and avoid accidents when workers and/or objects are in irregular/unexpected positions.

What all of these use cases have in common is that both the safety and security of the MRS operation are affected by the high degree of dynamicity in system structure, (sub)system availability and operational situations the systems have to cope with. To react resiliently (i.e. safely, securely and efficiently) to changes in system and environment, sophisticated self-awareness and situational awareness are required on different levels to estimate the dynamic risk of MRS behaviours in a given situation and estimate the dynamic availability of system capabilities to mitigate and manage those risks. Traditional approaches to dependability engineering assume both system and environmental dynamics to be worst-case during design-time safety analysis and design the system dependability concept to be ready for the worst-case. Due to the mentioned high dynamicity, worst-case assumptions lead to poor performance, as the majority of the operational situations are non-worst-case situations and system states.

To overcome the limitations of existing approaches, this deliverable describes the SESAME runtime safety and security concept that is realized through the Executable Digital Dependability Identity (EDDI) and contains capabilities to dynamically (1) monitor dependability-relevant events in the system and environment, (2) estimate the uncertainty of machine-learning-based perception outputs, (3) create self-awareness through model-based inference of MRS reliability attributes as well as MRS safety capability availability, (4) create dependability-relevant situational awareness to dynamically determine the risk of potential MRS (mis-)behaviours in the current operational situation and (5) use dependability-related self- and situational awareness to take and execute optimal adaptation decisions, which keep the risk acceptable during operation and maximize operational performance.

A key feature distinguishing the EDDI from existing runtime dependability management schemes is its formal link to design time dependability artefacts. To that end, SESAME builds upon the holistic model-based dependability engineering methods that have been first introduced in the H2020 project "Dependability Engineering Innovation for Cyber-Physical Systems" (DEIS)[1]. The Digital Dependability Identity (DDI) concept in DEIS had a strong focus on providing a common integrated

[1] Project website: https://www.deis-project.eu/

metamodel for design-time dependability engineering artefacts and the assurance case, but left open the formal integration of the design-time artefacts with executable runtime dependability mechanisms. This gap is closed in SESAME and is indicated through the evolution from DDIs to **E**xecutable DDIs.

In contrast to runtime dependability approaches, where risk assessment and risk mitigation are often directly integrated into the nominal function of the system, the EDDI is executed in a control loop separated from the nominal function based on the Monitor-Analyse-Plan-Execute-Knowledge (MAPE-K) scheme loop [1]. EDDIs form the combined dependability 'knowledge' of the MRS, captured during development, and when deployed in the field, enhancing the underlying MRS MAPE-K loop with respect to dependability. Using a separate "dependability control loop" has significant advantages for reacting to unforeseen events, because the dependability logic is not scattered over the components of the nominal function. Thus, EDDIs can be updated more easily, the interfaces to the nominal system are slim and explicit, and assurance of the continuous evolution of EDDIs is simplified. Interfaces between EDDI dependability manager and the nominal system are required to get access to the perception stacks of all MRS systems and to trigger adaptation actions on the individual systems. As such, the EDDI dependability manager acts as the onboard dependability intelligence supervising risks and "asking" the MRS to adapt behaviour if necessary and possible.

This deliverable describes the EDDI runtime model specification, i.e. the methods and models needed for assuring MRS dependability at runtime, and it is structured as follows. In section 2, we provide an overview of our vision for EDDI usage at runtime, what we consider to be its main responsibilities, and how the remainder of the document is positioned to support this vision. In section 3, we describe the first of our runtime building blocks, which introduces to MRS the ability to assess safety capabilities dynamically via runtime EDDIs, allowing them to adapt their behaviour to their collaborative operational context. In section 4, our next building block allows MRS to perform runtime reliability assessment, such that the MRS can be aware of imminent or existing failures. Section 5 presents our approach for dynamic risk assessment, which allows MRS to evaluate the risk of a given situation, based on observed causal factors evaluated probabilistically. In section 6, our concept for dynamic perception uncertainty monitoring is responsible for assessing whether ML models (often used for perception) are uncertain in a given runtime situation, and might therefore be unreliable. In section 7, a concept for specifying and monitoring generic runtime events is presented, which can be form the basis of an EDDI-level communication protocol through the propagation of events (and eventually recommended actions to the MRS). In section 8, we discuss our preliminary concept for using intrusion detection, and how this can be exploited by runtime EDDIs. We conclude in section 9, summarizing the deliverable's main points and outlining the next steps in the context of the SESAME project.

The technical realization of runtime component generators to semi-automatically generate executable components of the runtime models described in this deliverable is described in Deliverable D7.2 (Figure 3). For the first iteration of the project, it was assumed that runtime EDDIs are executed in a centralized way. Deliverable D7.3 will elaborate on this aspect and consider distributed execution of runtime EDDIs.

# 2. RUNTIME DEPENDABILITY MONITORING ARCHITECTURE WITH EDDI

Assuring the dependability of multi-robot systems is usually performed almost entirely at design time, by anticipating the most critical environmental situations the system will find itself in, and by designing the system in a way that it can cope with these situations. This leads to poor system performance for highly dynamic systems in highly dynamic environments because the worst-case situation is almost never present and therefore the system acts more conservative than it needs to on many occasions. To account for this issue, runtime dependability monitoring architectures shift dependability knowledge and activities into the runtime and therefore enable the system itself to manage the dependability. EDDIs are the core elements of the SESAME runtime dependability monitoring architecture, which follows the Monitor-Analyse-Plan-Execute with Knowledge (MAPE-K) control scheme (Figure 1, left). The general idea is to execute a control loop, which is not contributing to the system's original functional purpose, but instead preserves an acceptable level of dependability throughout the operation. To achieve that, dependability sensing, reasoning and control activities are required. Based on perceive dependability-relevant events, the dynamic risk of the planned behaviour in the current situation is determined together with the capabilities to mitigate or control the risk. If the risk is low given the capabilities, the system can be adapted to less-constrained behaviour, e.g. increasing speed in areas, where non-presence of humans is reliably detected. If the risk is too high given the capabilities, the system needs to be adapted to more constrained behaviour, e.g. by degrading its functionality or even entering the safe state by performing a minimum risk manoeuvre. All of these activities are fed by models capturing the dependability engineer's knowledge along with algorithms, which can perform automated inference on the models given the current situation and the robot's system state.

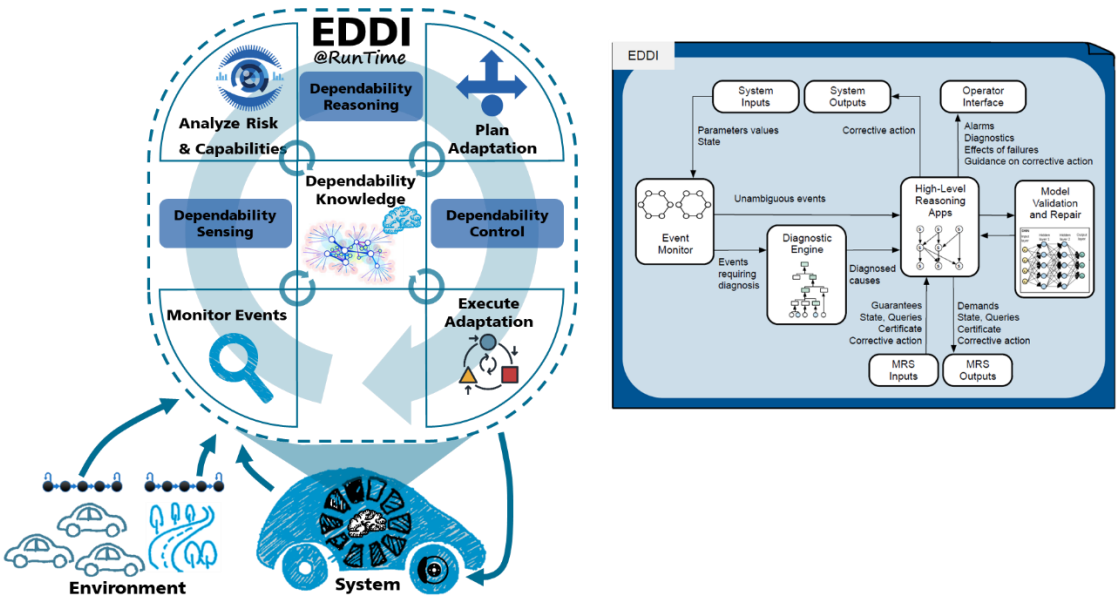


**Figure 1 Runtime EDDI Functionality**

EDDIs are triggered based on the observation of dependability-relevant events to either the MRS (per each constituent robot) or its operational environment. With respect to the MRS-internal events, dependability is concerned with system state changes that can either affect a robot's capability to perform its intended function safely or with a robot's

capability to execute safety functions properly when required. As a brief reminder, per the definition of dependability in [2], these changes may include the detection of robot errors or failures, which can impact safety (e.g. threaten the life of humans within the robot's vicinity), security and so on.

In addition to robot-internal events, the operational context may also change. Which system-internal changes are dependability-relevant depends on each robot's intended function, and which operational context changes are dependability-relevant depends on the specifics of the application domain of the MRS. As indicated in the examples mentioned from each of the SESAME use cases in section 1, careful consideration of the mission scope must be included to make this determination.

Architecture-wise, the EDDI component is connected to the systems forming the MRS. The EDDI has access to the perception stacks of the systems and thus all inputs required by the EDDI come from there (Figure 1, right). Based on the perception stacks' outputs, the event monitor evaluates the dependability of event presence. Sometimes, measurable events are symptoms, but the correct system reaction can only be chosen, if the causes of a symptom are known. Therefore, a diagnostic engine is used to determine the likelihood of causes concerning a symptom. Based on events and causes, high-level reasoning apps use various techniques (e.g. dynamic risk/reliability/capability/perception uncertainty assessment) and reasoning models (e.g. Boolean, Bayesian, Markovian, Statistical) to determine the current high-level dependability properties of the MRS. On the output side, the EDDI results in a corrective action, which is realized by one or more systems through adaptation. Note that risk reduction actions can be targeted towards the system or the system context. A situation, where a collision risk between a robot and a human is present, can be resolved either via the robot taking a path, which is non-intersecting with the human, or it can be resolved by actively influencing the human's behaviour, e.g. externally indicating the robot intent and hoping the human changes his path. In addition, there might be MRS, where humans are part of the system, e.g. as operators. In these cases, an EDDI action could also be a notification to a human operator or a call to action.

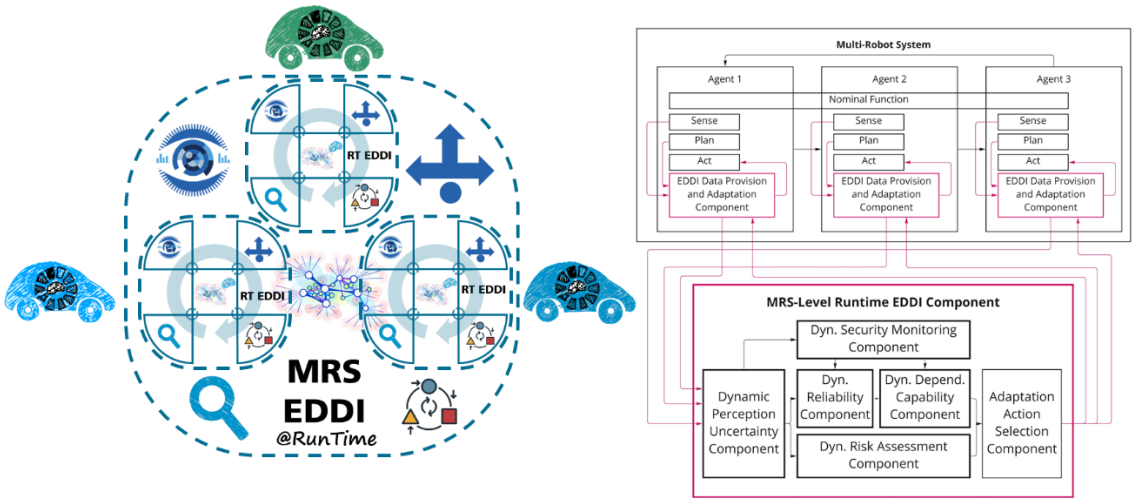


**Figure 2 From single-system EDDI to MRS EDDI**

Having explained the principal functionality of an EDDI for an individual system, the idea can be easily expanded toward MRS following a common mission goal (Figure 2).

In the MRS scenario, the EDDIs of single systems operating on their own knowledge base are connected and combined into an MRS EDDI operating on the whole knowledge of the MRS and triggering corrective actions affecting the whole MRS behaviour or mission.

To realize MRS EDDI operation, models need to support modularity and algorithms are required for dynamic model composition and online model inference. This deliverable represents the first iteration of the runtime dependability management concept. For the EDDI concept described in this iteration, an assumption is that the EDDI is executed in a centralized way by a "master" node, either by one robot or a cloud system, and the single systems operate as "slave" nodes, providing access to their perception stacks and action interfaces. This assumption is planned to be dropped in the second iteration, which will result in the EDDI distribution concept described and realized in deliverable D7.3.

The authors in [3] propose a conceptual framework to support decision-making in potentially novel conditions. Their framework classifies five possible context scenarios, each of which incentivizes different abstract decision strategies, depending on the degree of information and familiarity with the circumstances the subject decision-makers find themselves in. The context scenarios range from the fully-informed and well-understood, where one must simply identify in which situation they are in and then apply a known rule, to the alien and chaotic, where careful but immediate action and close supervision of an evolving situation might be needed. MRS may similarly need to cope with varying decision contexts and feature an added requirement: their response should be, to some degree, designed a priori. As such, for contexts where MRS operate under full information (e.g. full visibility of surroundings) and 'known knowns', a ruleset can be directly encoded and followed. As context conditions deteriorate, the MRS may be required to adapt accordingly to accommodate uncertainty and lack of experience. With the above framing in mind, the following sections describe the specific methodological parts for each used technique. These aspects include, for instance, how the models look like, what are their contents, what are suitable inference algorithms, and how is modularity and distribution of MRS supported.

Having established our different concepts for runtime EDDI models, including their responsibilities, functionality, and semantic interfaces still leaves open the issue of developing runtime components that can be deployed in MRS and perform the intended EDDI behaviour. This is the responsibility of SESAME task T7.2, which follows the ideas outlined in the conceptual stage here, and implements equivalent EDDI runtime components. To support scalability and interoperability, among other benefits, the generation of runtime components is supported via tools and is further described in D7.2.

We should also note that the concepts described here do not materialize themselves at runtime; they are expected to be the result of rigorous engineering activities performed during development. For instance, the dynamic safety capability assessment models are expected to be the product of the safety assessment phase of the development lifecycle. Similar statements hold true for other models and properties specified to support the other concepts. As such, for a deeper understanding of what is involved in the development of EDDIs, the work described in SESAME WP4 and WP5 tasks, and

specifically deliverables D4.1 to D4.4 and D5.1 to D5.3 are relevant to safety and security respectively.

With respect to the interaction between safety and security, the work described in D4.3 can also impact the runtime; as mentioned earlier, monitoring security at runtime (in our specific concept through the use of intrusion detection) is a prerequisite for MRS awareness of intrusion attempts or attacks, and an enabler for corresponding MRS response to said intrusions e.g. via performance degradation. Thus, during development, the safety assessment influences the security assessment, leading to the rules specified for intrusion detection, and the MRS inherit this knowledge distilled through the generation and deployment of runtime EDDIs.

D5.4 is also notable for providing additional detail into mechanisms for enhancing EDDI tailorability both during development and for deployment in different MRS applications. The latter mechanisms affect the technical realization of the concepts in this deliverable but are further detailed in D7.2. The above relationships are summarized in Figure 3.

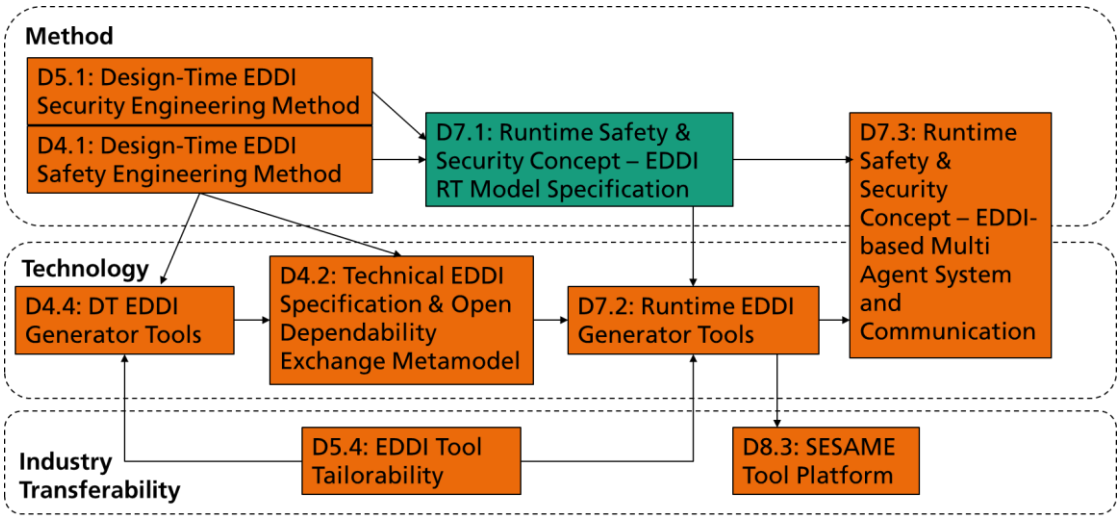


**Figure 3 Relationship of D7.1 to other SESAME deliverables**

# 3. DYNAMIC SAFETY CAPABILITY ASSESSMENT

In the use cases of the SESAME project, autonomous multi-robot systems operate in complex environments by cooperating with various other systems. Such cooperation is only possible if certain information and services are shared. Unfortunately, the lack of knowledge regarding external services and their safety properties typically leads to worst-case assumptions, which in turn severely constrain performance, or even lead to the decision not to use external services or information at all. A straightforward solution to this problem can be to enable constituent systems of MRS to explicitly negotiate their safety-related properties at runtime. This implies that we establish runtime safety models describing these properties for a (constituent) system and standardize a protocol for their negotiation. *Conditional Safety Certificates* (ConSerts) are an approach to do exactly that.

## 3.1 CONDITIONAL SAFETY CERTIFICATES

*Conditional Safety Certificates* [4] [5] (ConSerts) operate on the level of safety requirements. They are specified at development time based on a sound and comprehensive safety argumentation (e.g. an assurance case). They conditionally certify that the associated system will provide specific safety guarantees. Conditions are related to the fulfilment of specific demands regarding the environment and the fulfilment of the conditions is checked during runtime. In the same way as "static" certificates, ConSerts shall be issued by safety experts, independent organizations, or authorized bodies (depending on the respective application domain) after a stringent manual check of the safety argument. To this end, it is mandatory to prove all claims regarding the fulfilment of provided safety guarantees by means of suitable evidence and to provide adequate documentation of the overall argument – including the external demands and their implications.

Conditions within a ConSert manifest in relations between potentially guaranteed safety requirements ("guarantees") and the corresponding demanded safety requirements ("demands"). Demands always represent safety requirements relating to the environment of a component, which cannot be verified at development time because the required information is not available yet. These demands might directly relate to required functionalities from other components.

On the other hand, evidence can be required beyond that, since safety is not a purely modular property and it cannot be assumed that a composition of safe components is automatically safe. To this end, ConSerts support the concept of so-called Runtime Evidences (RtE) as an additional operand of the conditions. RtEs are a very flexible concept. In principle, any runtime analysis providing a Boolean result can be used. RtEs might relate to properties of the composition or to any context information, e.g. a physical phenomenon such as the temperature of the environment that is safety-relevant. Other RtE requires dynamic negotiation between components.

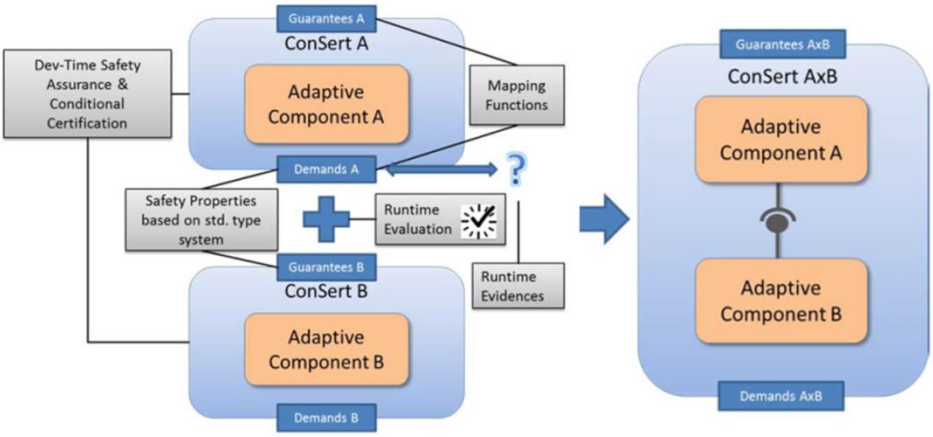


**Figure 4 ConSert Composition Conceptual Overview**

In any case, ConSerts must be available at runtime in a machine-readable representation and the systems need to possess mechanisms for composing and analysing runtime models. Based thereon, a valid safety certificate for the overall system of systems can be established. ConSerts are a relatively lightweight runtime safety approach and they are not far from traditional safety engineering. The main difference is that unknown

context is structured into a series of foreseen variants, which are then specified in a runtime model to be resolved at runtime.

ConSerts capture modular, conditional, and pre-assured safety concepts that enable dynamic reconfiguration of the underlying system (of systems) based on observed changes in the operational context (Figure 4). This relates to runtime fault diagnosis and dynamic risk assessment in the following way: Dynamic risk assessment dynamically determines MAS or constituent system safety goals along with a dynamic rating of the safety goal integrity derived from the risk estimate. This represents the set of top-level safety goals that need to be fulfilled in the *current* operational situation.

Having these dynamic safety goals, the system has to have a safety concept in place that is able to address those dynamic safety goals. Since different safety concepts are conceivable to address different safety goals and variable integrity demands, the need for specifying variable safety concepts arises. Such a safety concept is operationalized by so-called safety capabilities that the system uses during operation. Examples of such capabilities are fault diagnosis mechanisms to establish runtime evidences, fault tolerance mechanisms to achieve safety guarantees or mechanisms for the transition into a safe state. As such, ConSerts are formal representations of dynamic safety concepts expressing the dynamic parts of a safety concept necessary to distinguish between the availability of different safety capabilities and in consequence different safety guarantees that a system can give at runtime. Runtime diagnosis techniques are related to ConSerts in that they provide the basis for observing runtime evidences that represent the leaf nodes in a ConSert tree.

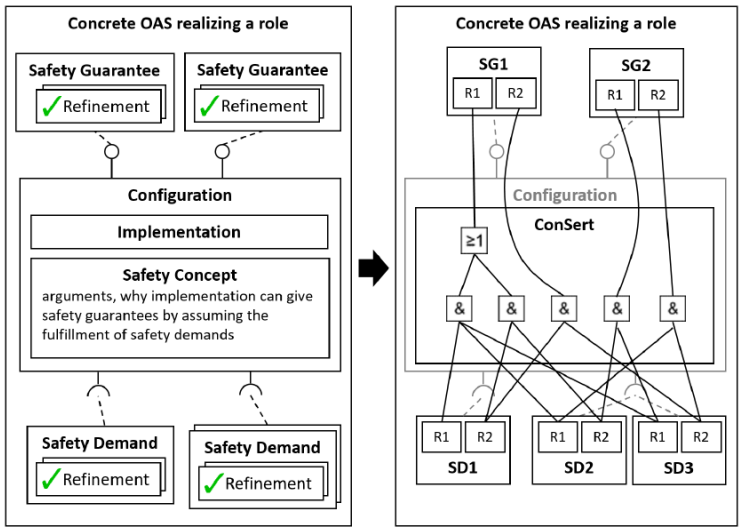


**Figure 5 Relation between safety concept and ConSert for an open adaptive system**

Figure 5 highlights the relationship between a design-time safety concept and its transformation into a Boolean model representing the dynamic parts of the safety concept.

## 3.2 APPLICATION IN SESAME

The multi-robot systems such as the disinfection robot of LOCOMOTEC operate in complex environments such as hospitals or, generally, general public buildings. During operation, the robot must guarantee that it will not harm any humans, animals or

property. For example, the disinfection robot must turn off the light when a human is in close vicinity. In order to fulfil its task, the robot uses services from other components or systems. For example, a component is used that handles the detection of humans in close vicinity. This service is safety-critical because it directly impacts whether the disinfection lights are turned on or off. According to the safety requirement, high confidence in the human detection service is required. In this example, a ConSert can first determine whether cooperation between these systems is feasible. The ConSert checks whether the demanded safety requirements can be guaranteed by the other system. If cooperation is feasible, the ConSert evaluates during runtime whether the guarantees are still provided and respectively recommends controlling the disinfection lights.

Similar safety requirements also exist for the other use cases. Drones e.g. which must have precise localization and object detection use different components and might communicate with other systems to fulfil their task. Further, Industrial plants with cooperating robots can ensure safety for reconfigurations through ConSerts.

## 3.3 METHOD AND ALGORITHM

Within SESAME various components and subsystems are developed which support the robots of the use case partner to fulfil their task. ConSerts are used to ensure safety for these robots and especially for the composition and cooperation which evolves during this project. During operation, a robot uses services of various (sub-)systems such as trajectory planner, human detection, other robots, etc. The majority of these systems are safety-critical. In order to fulfil their task, external information and services are used that must fulfil specified safety requirements. ConSerts have formally specified which safety requirements are required (Demands) for external services and which safety requirements are provided (Guarantee) for their own provided services. The ConSerts can then check whether an external service fulfils the demanded safety requirements. In case the external services are feasible cooperation between the systems is possible.

After cooperation between two systems is established, data is exchanged. Further, the safety guarantees are updated continuously. In case some internal or external parameters change a guarantee could potentially no longer be provided. This directly impacts the evaluation of any higher level ConSert, i.e. any ConSert that receives this guarantee as a demand.

## 3.4 EXAMPLE

The LOCOMOTEC disinfection robot must detect humans relative to its position to determine whether it can fulfil its disinfection task. If a human is too close to the robot, the UV lights must be turned off to prevent harm to the human. In this example, it is assumed that the robot has two operation modes “fast disinfection” and “slow disinfection”. The most efficient “fast disinfection” mode has the highest safety requirements. In order to disinfect at the highest possible speed, humans must be at least 20m away and the position of the robot must be accurate. In case humans are detected at a distance higher than 5 meters but less than 20 meters and the localization is accurate, the robot can still disinfect but at a slower speed. On the other hand, if the localization is inaccurate the robot can only disinfect at a slow speed if humans are at least 20 meters or more away. This logic is visualized in a ConSert in Figure 6. While the human detection is here realized as runtime evidence and thus as an internal property, the

accuracy of the localization is displayed as demands. Demands are safety requirements that must be fulfilled by external systems or components. In this example, a localization module provides a localization service and two guarantees according to the accuracy of the localization. Depending on the available technologies, the localization of the localization module is more accurate. Here e.g., the localization is more accurate when both the sonar sensor and the infrastructure sensors are working correctly. On the other hand, only with the Sonar-based sensors, only an inaccurate localization can be guaranteed. In case, the sonar sensors are not working, no localization can be guaranteed.

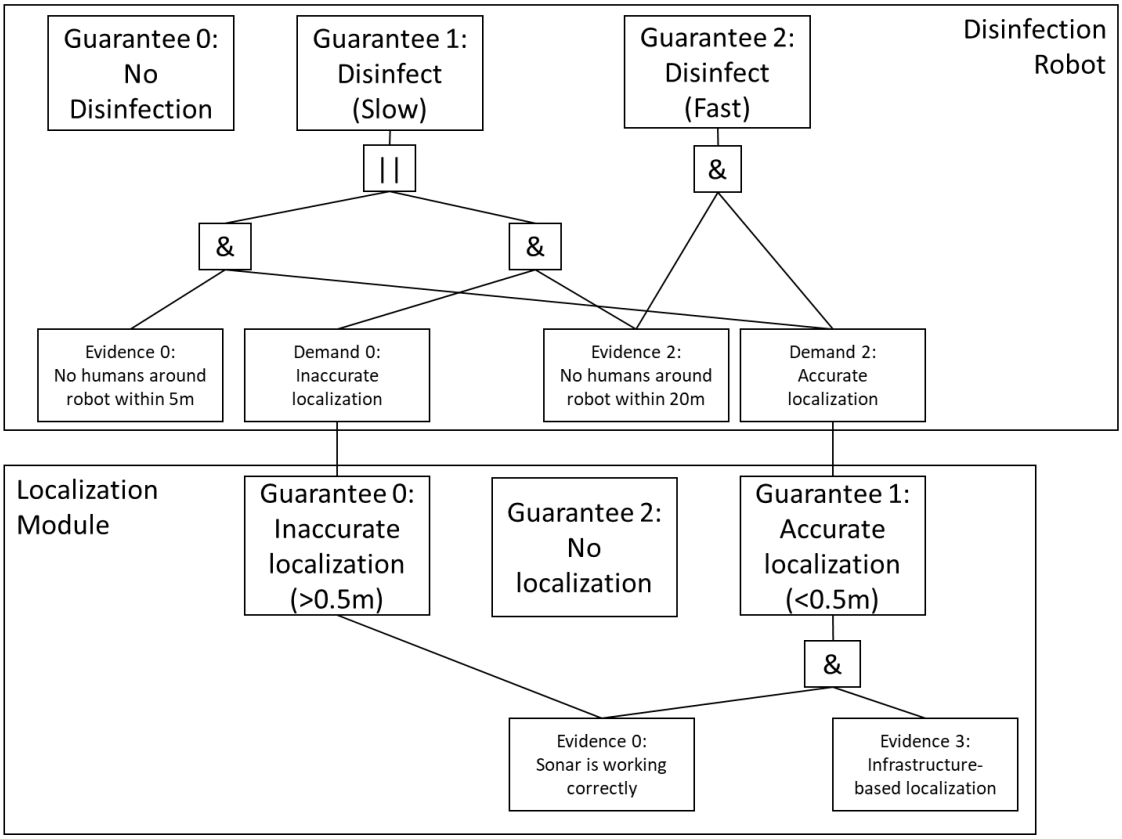


**Figure 6 Example ConSert for the LOCOMOTEC use case**

Now, the localization module could be swapped during runtime. The new localization module would only provide the inaccurate localization guarantee or no localization guarantees at all. Then, the ConSert would evaluate, that composition is possible because the demand of the disinfection robots matches the guarantee provided by the new localization module. However, the robot could only operate in the degraded mode and disinfect at a slower speed because the better fast disinfection mode would require a guaranteed accurate localization.

Further, during runtime, whenever a runtime evidence or demand changes, the ConSert reevaluates its guarantees and thus adapts its operation mode if needed. For example, let's consider the situation where a person approaches the robot and the distance falls below 20 meters. The former fast disinfection guarantee can no longer be provided and the robot would change its operation mode to disinfect at a slower speed. The same applies to changing demands. If the localization module does not receive any localization data from the infrastructure and its Lidar sensor is damaged, then only the

inaccurate localization guarantee can be provided. This also changes the evaluated guarantee within the disinfection robot, because one demand is no longer fulfilled and fast disinfection can no longer be executed.

# 4. DYNAMIC RELIABILITY ASSESSMENT

Of the different SESAME use cases, two use drones — the Power Station Inspection use case and the viticulture use case. Drones, and Unmanned Aerial Vehicle (UAVs) in general, have a wide range of uses across multiple domains in addition to the emergency response and agriculture applications considered in this project. However, one of the major barriers to widespread acceptance of UAVs is ensuring their safety during operation, particularly as they are typically expected to operate in highly dynamic environments where operating conditions cannot easily be assessed a priori in a design-time model.

To that end, as part of the project, we have been developing a prototype real-time safety monitoring approach for UAVs called "SafeDrones". This approach is intended to illustrate the utility of the EDDI concept by creating a dynamic, onboard safety monitor that can provide real-time information that is used to inform decision-making during operation.

While not specifically limited to drones, we have chosen to begin with them because they are a common robotic platform in many MRS/MAS applications with many generic similarities in failure behaviour. This overall failure behaviour can be captured in a higher-level safety model such as a fault tree at design time, providing a sample structure to assess the ways in which the drone might fail and establishing how such failures are interconnected.

At runtime, however, this model can be connected to symptoms of failures that are detectable during operation (e.g. via readings from onboard sensors etc). While fault diagnosis is not explicitly considered at this stage, such data can be used to perform real-time estimation of the drone's safety and reliability attributes.

Because this assessment is performed based on real-world data obtained from the drone itself, it is also adaptive to the operational environment. For example, if the drone is operating in a high-temperature environment (such as a sunny day in a hot climate), this would be reflected in the readings of e.g. its motor, processor, or battery temperature sensors, yielding a potentially higher estimated risk of failure than if the drone was operating in a cooler environment.

With this information, either the operators or the drone itself can make decisions about how to respond to anticipated problems. If the data suggests an increased probability of failure for a particular aspect of the system that the higher-level model (such as a fault tree) indicates as being at high risk, e.g. because it is a single point of failure, then appropriate actions can be taken in response.

Such a capability could be well suited to experimentation with the two SESAME UAV-based case studies and may pave the way to more sophisticated EDDI-based dynamic dependability management.

In this stage of the project, to provide an early prototype of EDDI functionality, we have focused on one of the drone partners (KIOS) and proposed a fault tree for their drone. Figure 7 illustrates the proposed fault tree consisting of nine failure categories, including:

1. Communication system failure
2. Navigation system failure
3. Computer system failure
4. Environment detection systems
5. Propulsion system
6. Energy system
7. Obstacle avoidance system
8. Security system
9. and Landing system.

To provide a dynamic reliability assessment, the concept presented in [6] is utilized to consider larger reliability functions as special "complex" basic events (CBEs) in the fault tree.

Figure 8 shows a small version of the proposed fault tree with only three complex basic events —Processor Failure, Battery Failure and Propulsion System Failure. Each chosen complex basic event is chosen in particular to demonstrate the different capabilities of the proposed approach. For the processor failure, the Arrhenius equation [7] is used to update the probability of failure and Mean Time To Fail (MTTF) based on monitored temperature. This basic event shows how a complex equation can be used as the basis for a complex basic event.

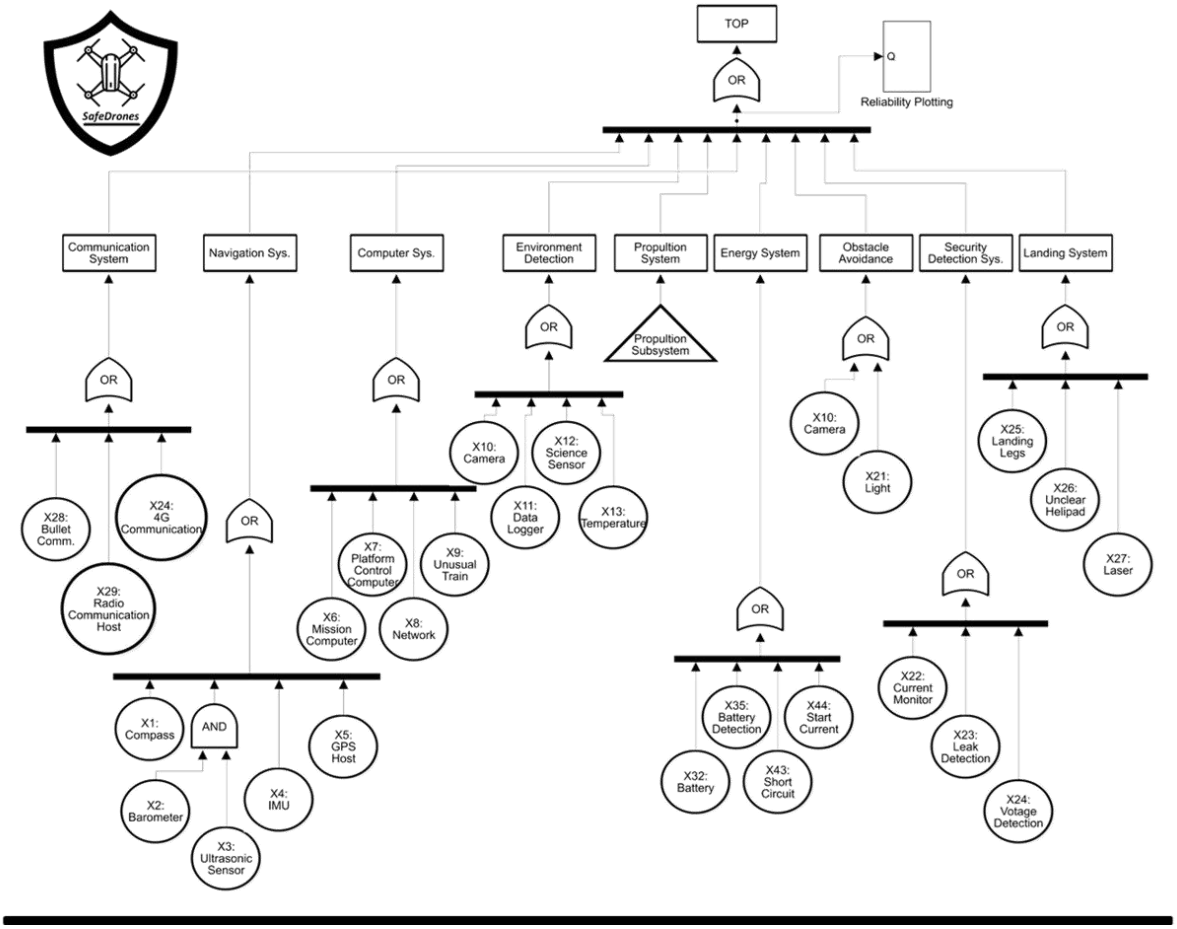


**Figure 7 The Proposed Fault Tree of a typical UAV**

The second complex basic event is the battery failure. In this event, a semi-Markov process is used to model battery degradation and failure [6]. In this model, it is assumed that the input symptom is battery level and based on that the current state in the semi-Markov process will be selected and the probability of failure will be calculated from that state at run time.

The third complex basic event is provided to show reconfiguration capabilities. In this complex basic event, one quadcopter and two hexa-copter reconfigurations are considered. The executable function receives the motor configuration and motor status as a symptom. Based on the motor configuration, the right semi-Markov model is selected and based on the motor status the right state in the model will be selected. From the selected state, the probability of failure and MTTF will be calculated and will be passed to the parent fault tree for higher-level reliability calculations [8]. It should be noted that all the reliability functions are executable and can be run on the drone's processor to update the reliability during the mission at runtime.

We call this overall concept "SafeDrones". The SafeDrones idea as an early prototype of EDDI functionality is illustrated in Figure 9. A conventional Fault Tree Analysis (FTA) tree has a top layer, many intermediate levels, and a basic events layer. However, there is a new layer in our suggested technique termed the symptoms layer. The safety expert(s) should identify probable observable events in the system and establish the relationship between symptoms and fundamental events in the symptoms layer. For

instance, in Figure 8, the symptoms are temperature, battery status, and motor status along with motor configuration. In Figure 8, it is assumed the temperature symptom only affects the processor and has no effect on the others. It is recommended in this proposed reliability modelling technique to employ CBEs to link with the symptoms. A CBE can take many different forms, such as a multi-state Markov chain in which the symptom influences the current state, a Bayesian Network in which a symptom can create a belief or any other reliability function in which a symptom can be a parameter.

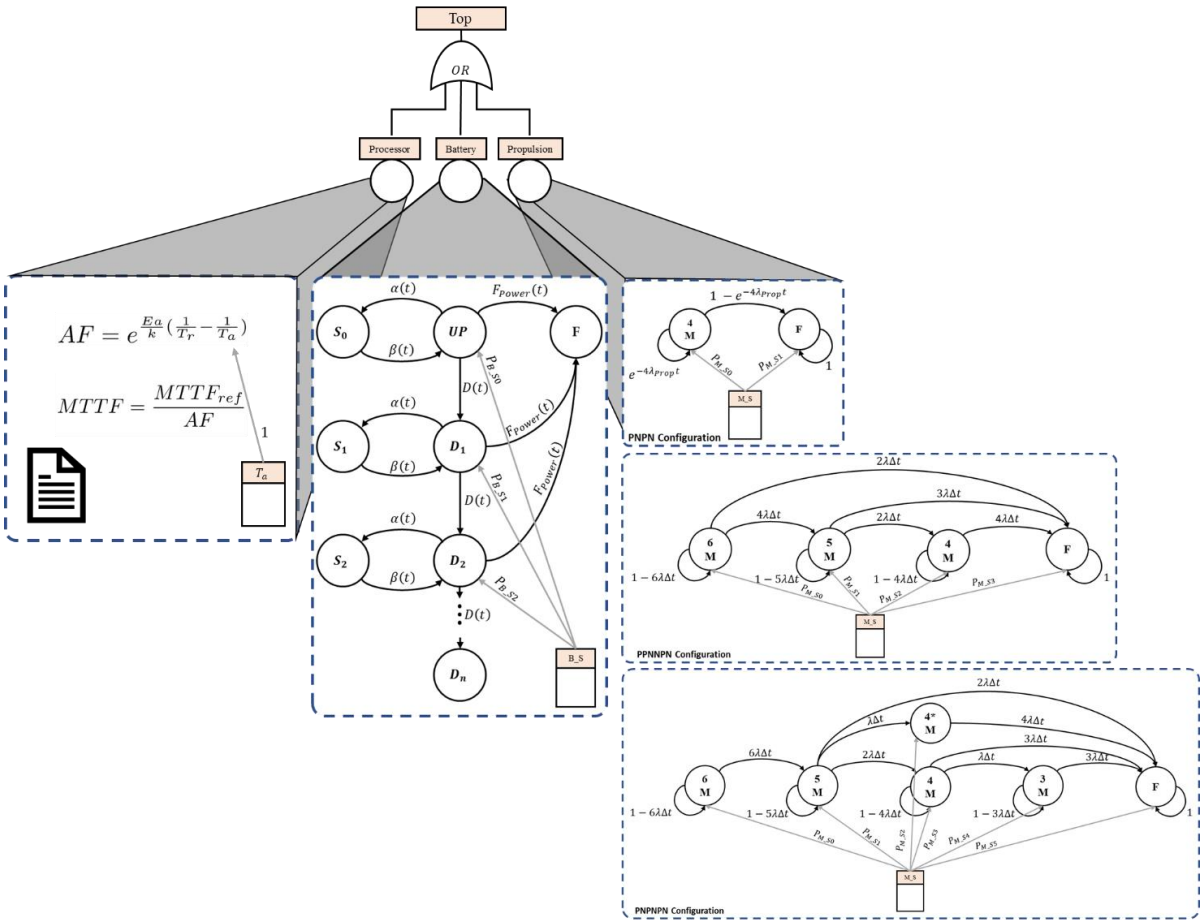


**Figure 8 Small FTA of a UAV with only three complex basic events – I) Processor Failure, II) Battery Failure and III) Propulsion System Failure -- considering failure symptoms and three different types of reconfigurations for propulsion system failure.**

Figure 10 shows a sample view of the KIOS software for controlling the UAVs. In this figure two orange and light blue boxes display the real-time monitoring values for the UAVs. With SafeDrones integration, each drone can calculate its real-time reliability and MTTF. As well as showing these values for informational purposes, they can also be used as the basis for dynamic decision-making; for example, if the MTTF drops below the expected mission duration, the drone could automatically be instructed to perform a safe emergency landing. In this way, the SafeDrones approach serves as a test case to demonstrate the EDDI dynamic safety assessment and response concept.

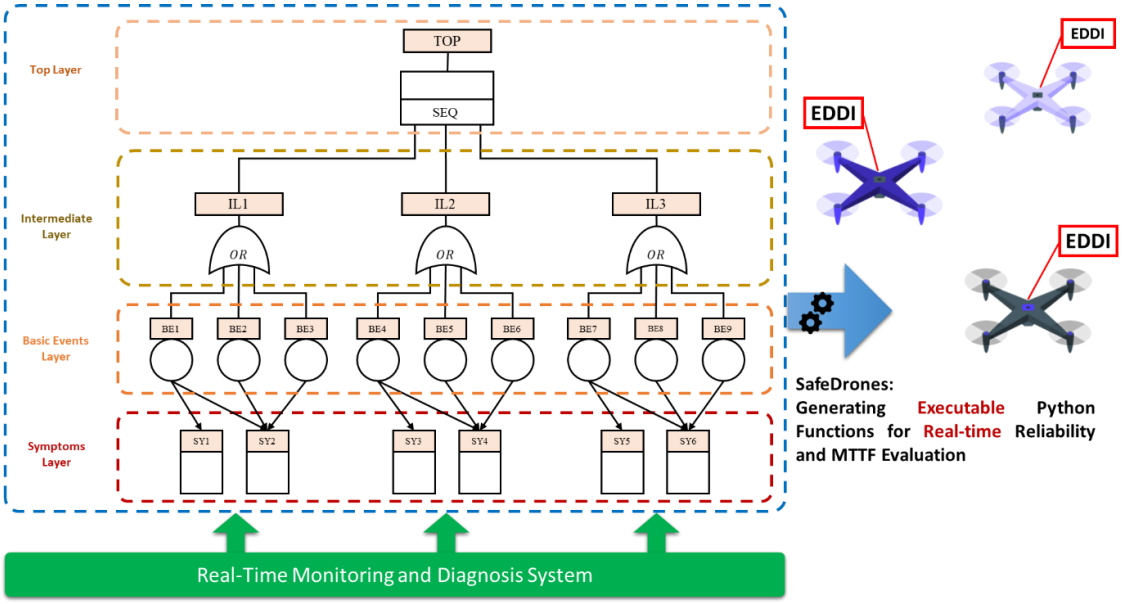


**Figure 9 Overall view on combining real-time monitoring and diagnosis with Fault Tree Analysis**

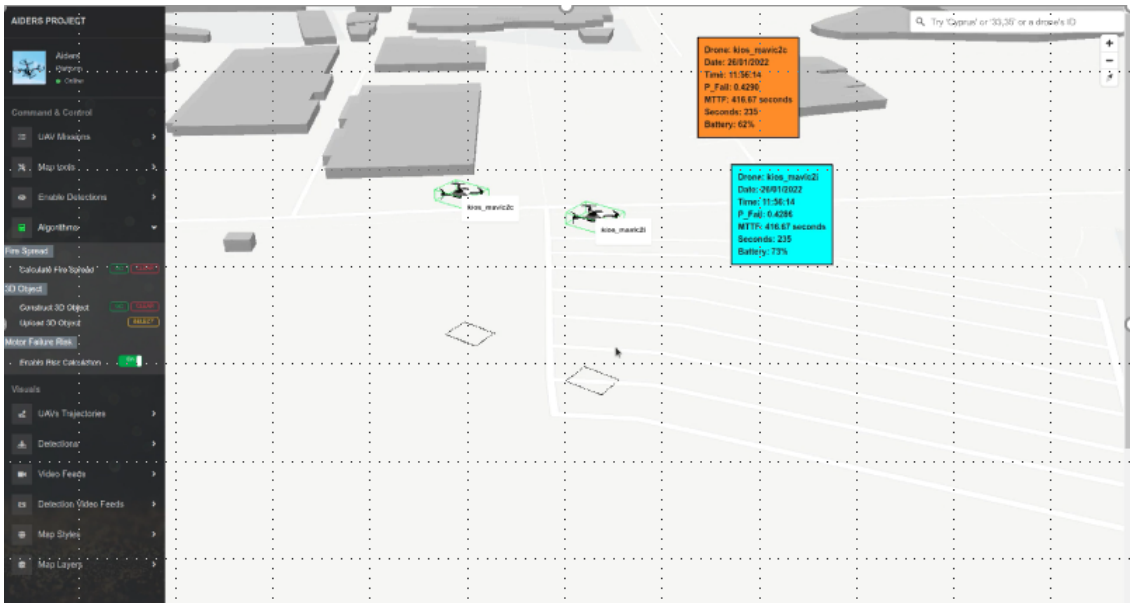


**Figure 10 A sample view of SafeDrones integration with KIOS software.**

Detailed calculations and experimental results are provided in the following paper that is expected to be published in 2022.

| **Project Publication** |
| --- |
| Koorosh Aslansefat, Panagiota Nikolaou, Martin Walker, Mohammed Naveed Akram, Ioannis Sorokos, Jan Reich, Panayiotis Kolios, Maria K. Michael, Theocharis Theocharides, Georgios Ellinas, Daniel Schneider and Yiannis Papadopoulos, (2022) SafeDrones: Real-Time Reliability Evaluation of UAVs using Executable Digital Dependable Identities. (Submitted for the 8th International Symposium on Model-Based Safety Assessment (IMBSA 2022)). |

To improve the research reproducibility, code, functions, demo notebooks, and other materials supporting the SafeDrones project are published online at GitHub: https://github.com/koo-ec/SafeDrones.

# 5. DYNAMIC RISK ASSESSMENT

Dynamic safety capability assessment as well as dynamic reliability assessment described in the previous sections focus on detecting causes for observed errors or failures *within* the multi-robot system. Therefore, variability is considered that affects the MRS capabilities to react dependably in a critical situation. However, whether a particular error or failure is safety-critical and poses an actual risk depends on the current operational situation the system finds itself in during runtime (Figure 11). For instance, if a planned trajectory of a drone differs from the specification due to a fault in the system, a collision with other dynamic or static objects may only occur if those objects are present in the current operational situation. Thus, hazardous events and their associated risk, i.e. the likelihood of a transition from a system hazard to an accident as well as the potential severity of this accident, are always conditioned on the operational situation. Since the operational environment of MRS in the use cases considered in SESAME is highly dynamic, determining the impact of the current situation on the risk parameters at runtime can increase system performance, as the MRS may continue operation in situations, where it would have stopped in conventional worst-case assumed environmental situations.

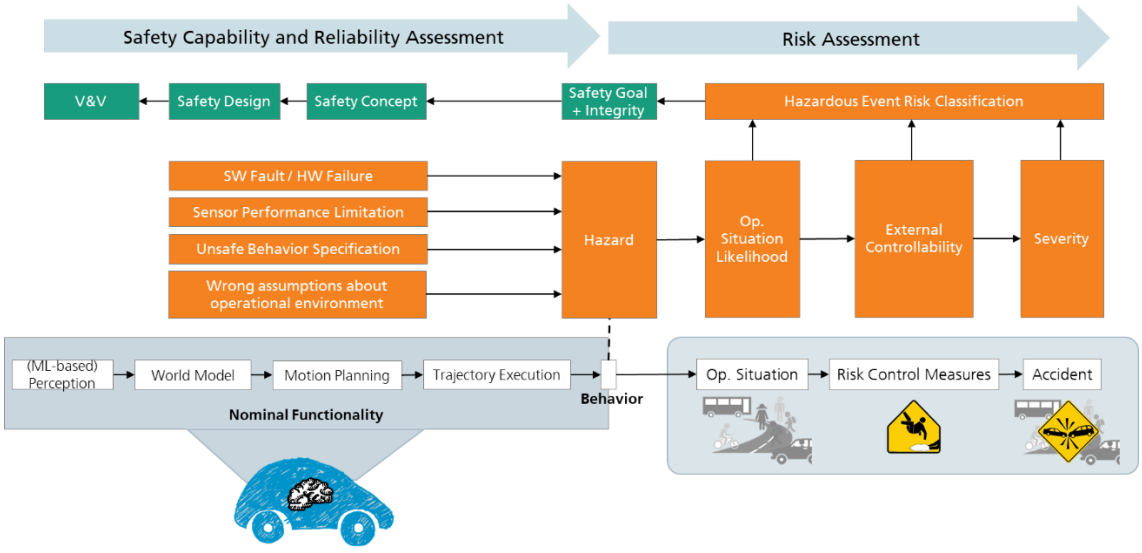


**Figure 11 Difference between safety capability/reliability assessment and risk assessment**

## 5.1 STATE OF THE ART

Current safety standards address this issue by designing the system in a way that safety goals and their integrity will lead to safe behaviour in all operational situations based on worst-case assumptions. Put differently, the system always expects the worst-case situation to happen. This approach indeed leads to safe behaviour and cost-effective safety assurance, as the situation space needs to be analysed only for the identification of worst-case situations. In reality, however, worst-case situations rarely occur and in

the majority of operational situations, the risk is low. This results in situations where diagnosed faults lead to the execution of minimum risk manoeuvres or a transition to the safe state, although the actual risk would not demand it. Consequentially, if we monitor the presence of risky/non-risky operational situations, we can increase MRS performance by a) being able to tolerate certain faults and failures if their associated risk is low in the current situation and b) actively reducing particular risk parameters with tactical decisions, where severity, controllability or the operational situation itself is changed.

Dynamic risk assessment (DRA) techniques thus treat the MRS or a constituent system as a black box and provide means to analyse the consequences of MRS behaviour deviations or present system hazards on the risk in the current operational situation. Note that DRA can be both performed for constituent systems and on the MRS as a whole. The difference lies in the definition of hazards resulting from deviating behaviour. Such hazards can be analysed for MRS collaborative behaviour or single system behaviour.

For autonomous systems, in particular in the automotive domain, DRA techniques have been applied to address the trade-off between performance and safety risks. The existing approaches can be classified broadly into three categories.

1. DRA is incorporated into motion and trajectory prediction frameworks by specifying behavioural or kinematic constraints that are *input* to the trajectory planner. Such constraints come either in the shape of an augmented map, which treats risk as occupied spaces the planner has to avoid, or they come in as boundary conditions the trajectory planner needs to respect, e.g. speed less than a particular threshold. The result of these approaches is usually a planned trajectory that is claimed to be safe. Representatives of this class are [9] and [10].

2. DRA is performed during the online verification of an already planned yet potentially unsafe trajectory. Thus, this class of approaches uses the *output* of the trajectory planner and checks afterwards whether it may lead to unsafe behaviour in the current operational situation. If the safety criteria of the verifier are violated, a transition to a safe state or the degraded operation mode is triggered. Representatives of this class are [11] and [12].

3. DRA is not performed in direct relation to a planned trajectory, but instead monitors influence variables that enable distinction between the presence/absence of a hazardous event and influence its criticality through risk parameters such as exposure, external controllability and accident severity. Such DRA approaches are closely related to design-time hazard and risk assessment models, as the monitors capture the observable variability in these models. The output of these approaches is usually a list of safety goals that are relevant to the *current* operational situation along with the dynamic risk rating of these safety goals based on a dynamic assessment of the risk parameters. Representatives of this class are [13] [14] [15] [16] and [17].

Although different architectures for incorporating DRA into autonomous systems exist, all of them need to decide which particular risk-influencing situation features are to be observed in the present, project the evolution of these features into the future by using a

set of assumptions, and rate the risk of the projected future situation (Figure 12). For each of these DRA sub-tasks, respective design time engineering activities are required.

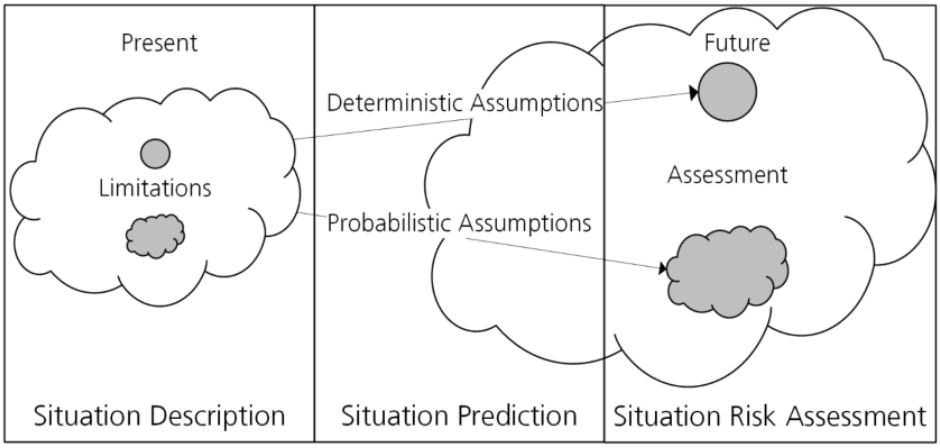


**Figure 12 Dynamic Risk Assessment Conceptual Overview [17]**

*Situation Description* demands a risk-driven situation space decomposition. This task is already performed during hazard analysis and risk assessment (HARA) at design time, where those operational situations are sought for indicating the highest risk. In contrast, DRA needs to identify risk-variable operational situations and thus extend current design-time HARA activities with a more fine-grained situation space analysis. This analysis requires an understanding of the intended operational domain and may consider situation features of dynamic and static objects, environmental conditions as well as interactions between actors.

Based on a selection of situation features being risk-relevant in the intended operational domain, *Situation Prediction* predicts the future state of the selected situation features based on different assumptions and prediction models. For instance, for autonomous vehicles, such behaviour prediction models may assume traffic rule adherence or constraints on the kinematic state such as constant speed or acceleration. The assumptions can be either deterministic or probabilistic.

Having a predicted state of situation features, finally, their relation to risk needs to be established during *Situation Risk Assessment*. For this purpose, risk metrics are used. Such risk metrics are typically highly domain- and even application-specific. Although on a high level, risk metrics always combine the likelihood of an unwanted event with its impact severity, the concrete relation between the predicted situation features and those risk parameters needs to be explicitly modelled. Examples of DRA metrics are Time-To-Critical-Collision-Probability (TTTCP) [18] and Deviation-From-Expectation [19].

Further information on situation prediction techniques and concrete risk metrics can be found in relevant literature reviews (see [17], [20], [21], and [22]).

Up to this point during design time, situation features have been selected based on a risk-driven analysis of the operational domain (possibly through a HARA analysis with a more fine-grained situation analysis), and prediction models for those situation features have been selected based on a set of deterministic or probabilistic assumptions

and domain-specific risk metrics have been selected to quantify the situation risk based on the predicted situation feature state.

To enable machines to perform DRA, modelling formalisms are required to technically capture the relationship between situation features and risks. For this purpose, different classes of models can be used, which are model-based (e.g. structural equation models), rule-based (Boolean models), generative (Gaussian Mixture Models, dynamic Bayesian networks, Hidden Markov Models), discriminative (Decision Tree, Random Forest, Support Vector Machines) or deep learning-based (Multi-Layer Perceptron, Convolution Neural Network – CNN, Recurrent Neural Network – RNN, Long-Short-Term-Memory Network – LSTM). Some of them are developed purely based on expert knowledge, and some of them can be adapted to new operational domains with machine learning techniques. Depending on the concrete modelling formalism selected, support for deterministic or probabilistic assumptions may be given as well as the consideration of uncertainties during feature perception is possible.

## 5.2 SITUATION-AWARE DYNAMIC RISK ASSESSMENT

At Fraunhofer IESE, DRA for autonomous systems was considered in a recent dissertation [115]. The work contributed a conceptual taxonomy of DRA and provided a concrete application of DRA using concrete instances of controllability and severity metrics to rate collision risks for automated driving. In order to be applicable in many different operational situations, one explicit requirement in [115] is that the risk metric is supposed to be situation-agnostic, i.e. it is only allowed to use the kinematic state of dynamic actors. Situation-specific features such as road structure, lighting and weather conditions, traffic rules, and actor interactions were consequentially not considered. To account for the fact that risk is often influenced by exactly those situation-specific features, the work in [115] was extended to the *Situation-Aware Dynamic Risk Assessment* (SINADRA) framework [23]. The approach uses Bayesian networks as a modelling formalism and explicitly considers the mentioned situation-specific features as risk influences. A proof-of-concept and tool implementation of the approach is presented in [24]. The design-time method for building Bayesian situation prediction models with machine learning techniques is presented in [25]. The SINADRA framework belongs to the third class of DRA approaches mentioned in section 5.1 and is the approach that has been integrated into the SESAME runtime concept to address dynamic risk variability.

The conceptual components of the SINADRA framework are shown in Figure 13. The core artefact of the framework is the *DRA Monitor Runtime Component*. It is an executable software component that is deployed to an MRS along with the nominal functionality. Functionally, it contains a model that captures the variability of risk parameters with respect to risk-relevant situation features. Thus, the output is a dynamic risk classification of particular (unsafe) behaviour intentions of interest, e.g. the current behaviour plan of a robot-like “Turn on disinfection and disinfect the next 10 meters” in the LOCOMOTEC use case. Based on the runtime perception of relevant situation features and the inference determining the effect of the dynamic input feature vector on risk, the basis for a risk-informed decision is given. Such a decision can be continuing operation as risk is determined low or can involve various types of adaptation, e.g. adapting the robot’s behaviour to eliminate the hazard or change the behavior or trying to adapt to the external environment by actively influencing the external controllability of people at risk or reducing accident severity.

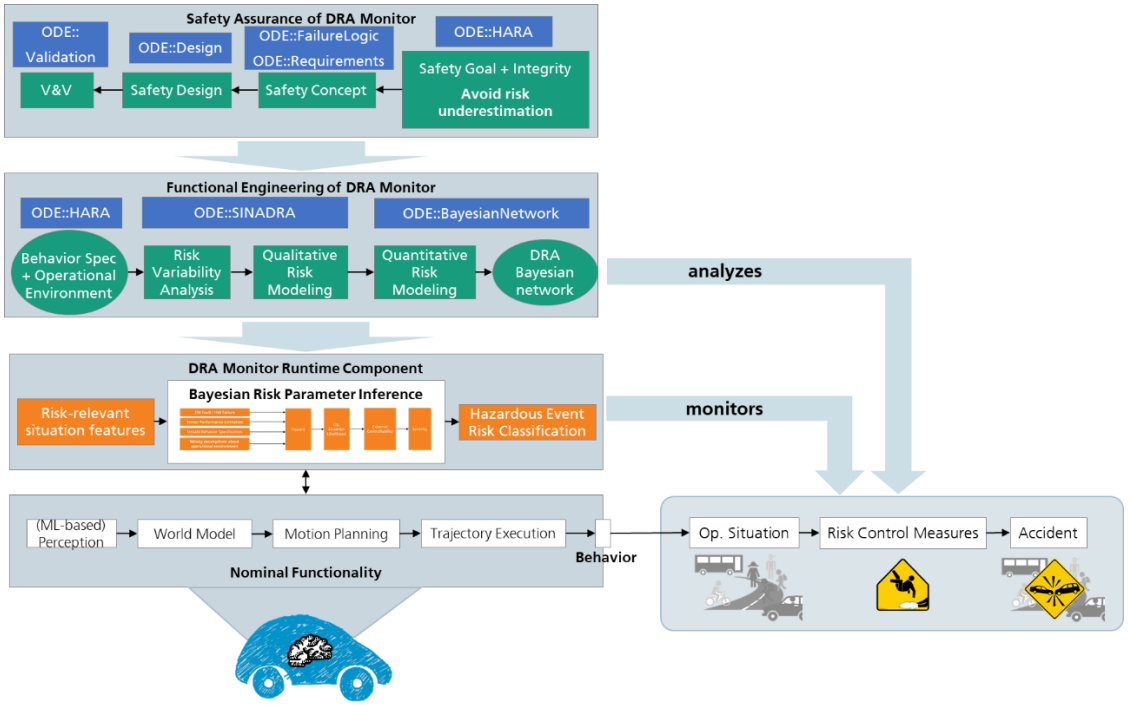


**Figure 13 Components of the SINADRA Framework**

Since the relevant contents of the DRA model vary for different applications and more importantly different operational environments, a systematic functional engineering method is required to come up with a *DRA Bayesian network*, which properly fits the system's behaviour and the intended target operational environment. The method consists of three steps:

The first step is the *risk variability analysis*. This analysis is comparable to conventional hazard analysis and risk assessment (HARA), except that it uses a much more detailed situation analysis than conventional HARA methods. The reason for this is that conventional HARA methods have the goal to assess worst-case risks for particular behaviour intentions. Thus, the situation analysis needs to efficiently find exactly those situations posing the highest risk. In SINADRA, a *situation-aware* dynamic risk assessment model shall be engineered, therefore the situation analysis needs to result in those situation features, which represent risk parameter variability. To that end, the ODE::SINADRA package (see deliverable D4.2 for more details on the contents of the refined situation analysis model) captures detailed elements to explicitly analyse and express situation features affecting the controllability parameters of people at risk. The output of this stage is a comprehensible mapping of risk-relevant situation features and their hypothesized effect on the risk parameters for the analysed behaviour intentions in the operational environment.

The second step is the *qualitative risk modelling*. In this step, the elements identified in the risk variability analysis are modelled in a directed cause-effect graph with the leave nodes representing the measurable situation features of interest and the target node representing the risk parameters. In addition, intermediate nodes are added that represent intermediate concepts required for comprehension of the cause-effect chain. This is required because a single situation feature might affect multiple different risk parameters at the same time in different ways. By adding the explicit reasoning, of why and how a situation feature affects a risk parameter causally, proper quantification of

those effects gets possible and, even more important, validation and verification of the effects can be done in a modular way by step-wisely finding evidence for each pair of nodes along the cause-effect chain. This is in particular important for the argumentation, of why the DRA component does not underestimate the risk. Figure 14 shows an example qualitative risk model for an automated driving shuttle in university environments.

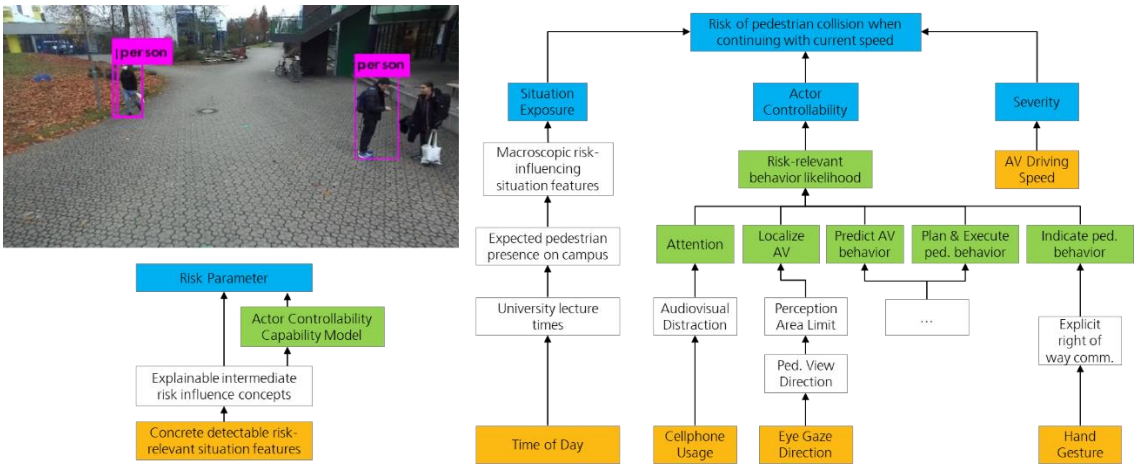


**Figure 14 Example Qualitative Risk Model (Excerpt)**

The last step of the engineering method is *quantitative risk modelling*. Here, the goal is to create a machine-processable model suitable for runtime inference. For this purpose, Bayesian networks (BN) have been selected for SINADRA. They combine the advantage of automatically parameterising quantitative parameters of cause-effect chains with machine learning techniques with the advantage to add expert knowledge to the network structure (based on our qualitative risk model). The first is necessary to parameterize the risk model for different operational environments and the second is necessary for the verification and validation of the BN, not only on the black box level like with other ML techniques such as Deep Neural Networks but also on the inner network structure. The output of this step is a BN capturing the quantitative risk variability of the MRS application in the target operational environment. At design time this BN is modelled via the elements of the ODE::BayesianNetwork package.

The DRA monitor component determines the risk of a behaviour intention in the current situation to inform risk-based decision-making at runtime. Therefore, wrong inference potentially leads to safety-critical consequences and, as with every other safety-critical functionality, evidence needs to be provided for why these critical consequences do not occur with defined integrity confidence. In Figure 13, this is represented via the safety engineering process at the top. The Open Dependability Exchange Metamodel contains respective packages to systematically perform the model-based safety engineering, by identifying causes for risk underestimation and deriving appropriate requirements to mitigate these causes. Within the DRA monitor, the two high-level causes for risk underestimation are the unreliable perception of situation features and the quantitative risk model leading to risk underestimation compared to reality. The perception issue can be tackled by using redundancy, e.g. sensor fusion, to mitigate critical misclassification. The risk model wrong inference can be tackled via the validation and verification of the Bayesian network itself, either via performing extensive testing or by performing expert-based sensitivity analysis on the inner structure of the BN.

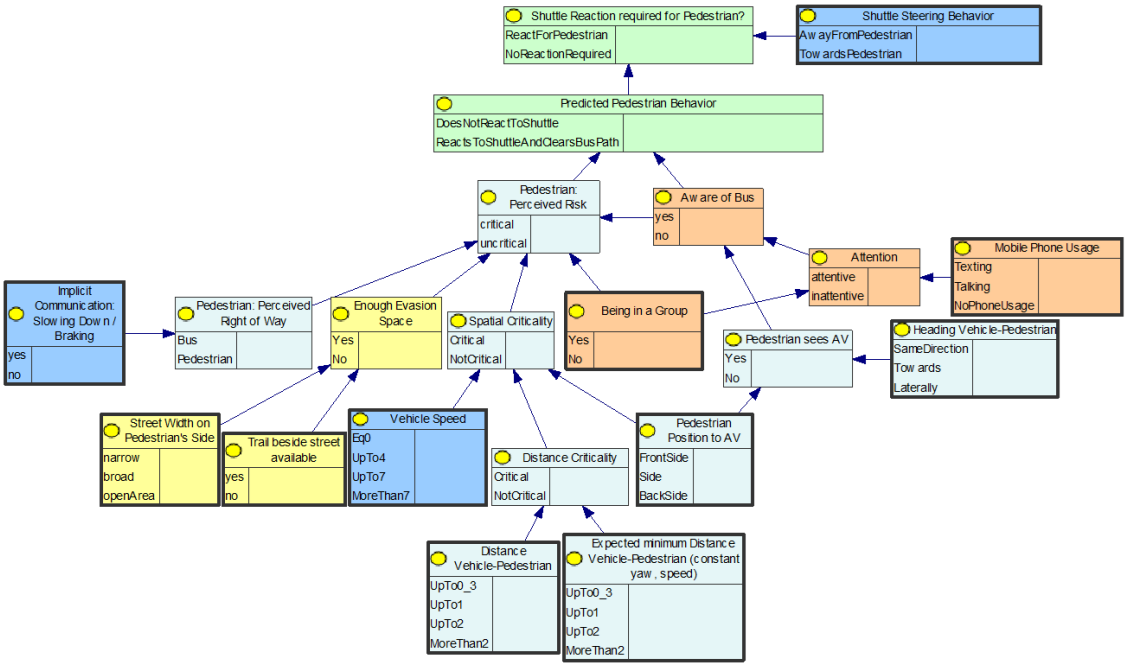


**Figure 15 Example Bayesian network for autonomous shuttle - pedestrian controllability estimation (Excerpt)**

In summary, SINADRA enables a system or MRS to automatically assess the risk of (malfunctioning) behaviour in the *current* operational situation. Based on this dynamic risk estimate, the unavailability of safety capabilities can be potentially tolerated in low-risk situations or tactical decisions can be enacted to actively lower the risk to an acceptable level. For this purpose, situation features need to be selected, their future state predicted, and a risk metric is calculated on the future state to get a qualitative or quantitative risk rating. Apart from a fully formal inference mechanism at runtime, quite some effort needs to be put into the safety engineering of the DRA mechanism for a particular use case in a particular domain. To come up with a meaningful trustworthiness argument for a DRA monitor, the selection of considered situation features needs to be grounded in a systematic HARA, the validity of the assumptions behind the situation prediction needs to be demonstrated and the adequacy of selected risk metrics for the concretely considered risks needs to be argued.

## 5.3 EXAMPLE OF SINADRA MODEL IN SESAME USE CASE

In order to demonstrate the principal applicability of SINADRA in SESAME, this section contains a preliminary example on how SINADRA can be beneficially used within the collaborative hospital disinfection use case provided by LOCOMOTEC. The use case including business context, system description and relevant validation scenarios are documented in deliverable D1.2. In short, a multi-robot system equipped with ultra-violet (UV-C) lamps is tasked to disinfect defined areas in a hospital on a regular basis to relieve hospital personnel from having to perform this task manually. UV-C radiation is safety-critical for persons in the vicinity of the robots, therefore UV-C over-exposure needs to be prevented by means of a safety concept.

Without SESAME technologies, the robots get a disinfection task consisting of waypoints to be navigated with UV-C lamps turned on. A ML-based person detection component realized with camera sensors perceives people in the sensing range and

triggers a robot stop and UV-C lamp turnoff as soon as persons are detected. After no person is detected again, the robot continues the disinfection task. Through these safety-motivated stops, the key performance indicator "time required for disinfection task" is increased. The hypothesis behind the SINADRA application in this use case is that the robot stops unnecessarily often, as the risk parameters associated with the accident "UV-C over-exposure" vary with the presence of features in the concrete operational situation. An example qualitative risk model for the disinfection application and in particular the UV-C overexposure risk is shown in Figure 16.

The severity of the accident is determined by the intensity of received UV-C radiation doses over time. The intensity of received UV-C radiation varies with:

- the distance between the UV-C lamp and person
- the angle between UV-C beam and person
- the target area on the human receiving the radiation (Radiation on eyes is more critical than on other body parts)
- the power of deployed UV-C lamps

The controllability of people in the hospital potentially exposed to UV-C radiation might depend on their knowledge about the danger associated with the robots. As such, assumptions about "training" could be used in dedicated hospital areas, e.g. intensive care units, where usually no visitors are allowed. Thus, trained personnel could work in the same areas in parallel to the robot disinfection at suitable distances without risk of UV-C over-exposure.

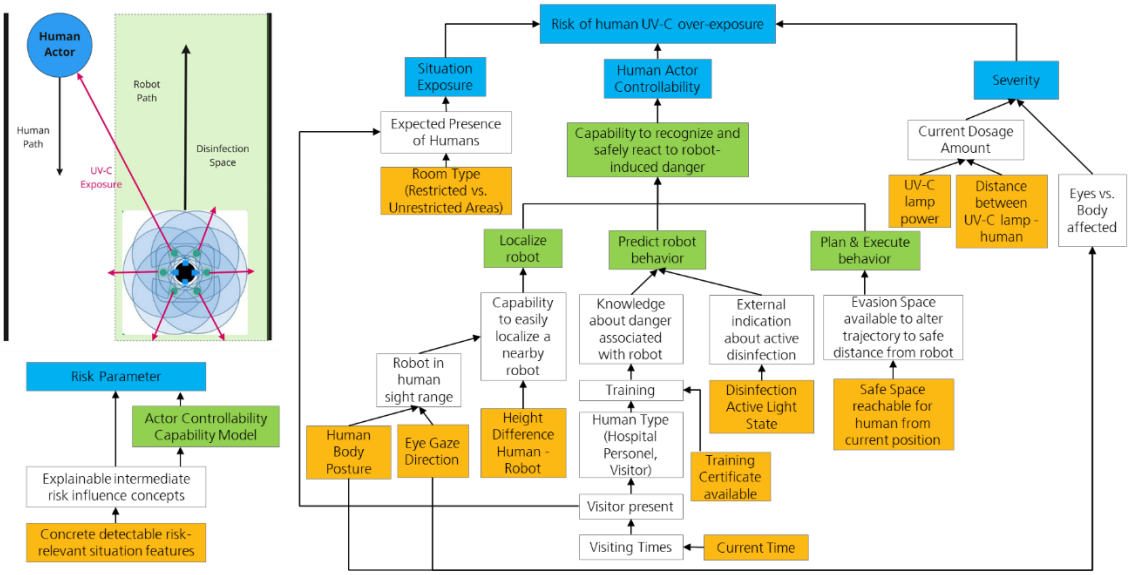


**Figure 16 Example qualitative risk model for human UV-C overexposure risks in disinfection use case**

Another risk variability leading to a performance potential is that all UV-C lamps are shut off in case of person detection. If the task is to disinfect along a wall, where devices are placed, it is not necessary to shut off lamps towards the wall, if it is ensured via detection that no person is between robot and wall. However, if the person is passing

by the robot on the other side, these lamps could be selectively shutoff without interrupting the disinfection mission, i.e. no robot stop would be required, the wall could be continued to be disinfected.

As one example for the quantification of a pair-wise causal hypothesis modelled in Figure 16, let us consider the relationship between "Distance between UV-C lamp-human" and the current intensity of the UV-C exposure reaching a potential human. Based on an experimental setup, LOCOMOTEC measured the worst-case UV-C intensities received at certain distances around the robot (Figure 17). The experimental evaluation suggested that UV-C intensity and therefore also the received dosage is decreasing non-linearly with higher distances. This relationship can be beneficially used in SINADRA, as for instance at 1.5m distance, the dosage is approximately almost 90% lower than with a 0m distance, i.e. a present contact between robot and human.

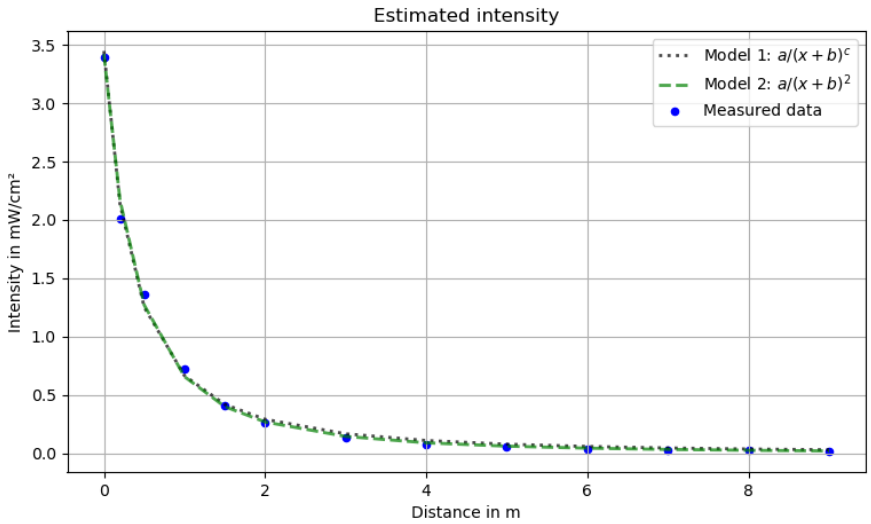


**Figure 17 Experimental determination of the influence of the robot-person distance on UV-C intensity**

In summary, the variability of risk parameters of a particular accident is highly application- and environment-specific. The above examples of risk-relevant situation features may be used to build up a SINADRA risk model according to the method described in the previous subsection. Note, however, that the principal idea behind using SINADRA is *not* to determine the absolute risk of each situation and lead to a shut-off, when the risk exceeds a certain threshold. Instead, the worst-case risk is always assumed to be present and SINADRA looks for situation features indicating lower or even the absence of risk with the goal to improve performance in those situations. For this purpose, assumptions about causal relationships between the situation features and the risk parameters are evaluated at runtime and used for planning the activation of safety functions. Starting at the worst-case risk and detecting low-risk indications has significant benefits compared to starting at zero risk and detecting high-risk indications: The assurance claims to be addressed for both scenarios are very different. If we are able to assure the system behaviour for worst-case risk and use a single feature indicating low risk, the assurance efforts are constrained to making sure this feature is detected reliably and the amount of risk reduction is properly determined. Thus, we can step-wisely extend the performance improvement, if new features are identified. In contrast, if we wanted to use an absolute risk determination, it would be necessary to argue that all risk influences have been considered and properly detected at

all times. In complex operational contexts, there is always an additional relevant risk influence unconsidered or even unknown, which may lead to a risk underestimation.

# 6. DYNAMIC PERCEPTION UNCERTAINTY MONITORING

Several SESAME use cases utilize Machine Learning (ML) components for object detection and perception tasks [26]. This is expected as the ML components have unmatched performance, especially when it comes to the perception tasks. For example, LOCOMOTEC uses ML components for identifying the person around the MRS. On the one hand, ML components increase the performance of the system, but they also introduce new sources of uncertainties. While it could be possible to address these uncertainties at design-time using "worst-case" assumptions, it renders these components inefficient, as they are over-sensitive to severely restricting performance in even mildly sub-optimal conditions. Hence, a more performant way of addressing this uncertainty is with the help of Dynamic Uncertainty Monitoring. In this section, we describe the technologies for dynamic uncertainty monitoring and their application in SESAME use cases.

## 6.1 BACKGROUND

It is well established that due to the nature of ML components, the output of ML components cannot be certain. The ML components carry inherent uncertainties with them. In [27], the authors classify uncertainty sources in three major sources in an 'onion' 3-layer model as shown in Figure 18. The three major sources of uncertainty are scope compliance, data quality and model fit.

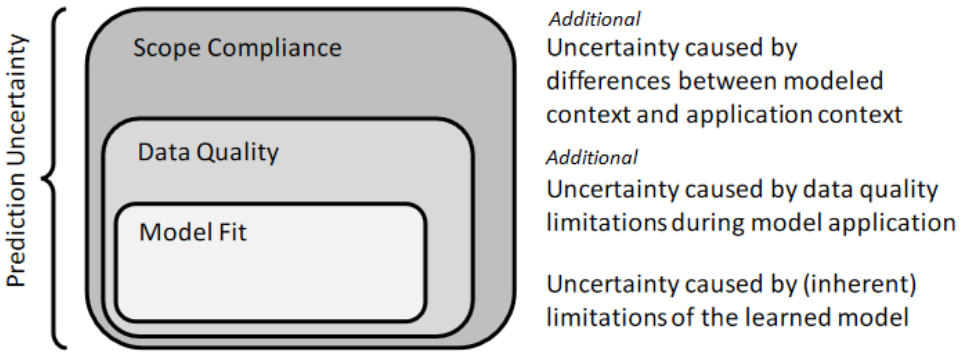


**Figure 18 Onion Layer model for uncertainty in ML components as described in [27]**

- **Model fit:** the uncertainties resulting from the inherent approximation of ML models are covered in model fit uncertainties.

- **Data Quality:** the ML components need input data. In practice, the data collected can have various sources of uncertainties (e.g., human error, sensor error, labelling quality and so on). These are covered under the umbrella of data quality.

- **Scope Compliance:** ML components are trained for a specific context (inherent to their training data). In practical applications, if these components are applied in a significantly different context, their behaviour becomes unreliable. Thus, the

uncertainty generated from the contextual differences is covered under scope compliance.

## 6.2 UNCERTAINTY MONITORING

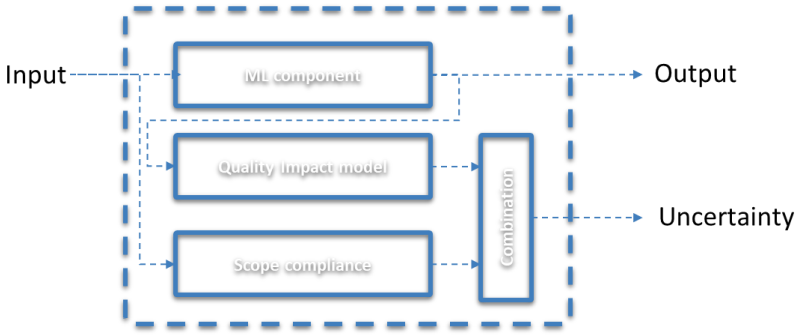


**Figure 19 Uncertainty wrapper overview**

Using the uncertainty classification mentioned in section 6.1, the authors in [28] propose an encapsulation of ML components within the "Uncertainty Wrapper" (UW). This extends the outcome of the ML component with the uncertainty information. Figure 19 shows the main component of the uncertainty wrapper. The uncertainty wrapper wraps the ML component output along with an uncertainty estimate. The uncertainty stemming from the data quality is computed by using the "Quality Impact Model". The Quality impact model is a model generated by varying the input quality in a controlled environment and using the modified input to obtain the performance assessment of the ML component. This is then combined with the uncertainty assessment of scope compliance.

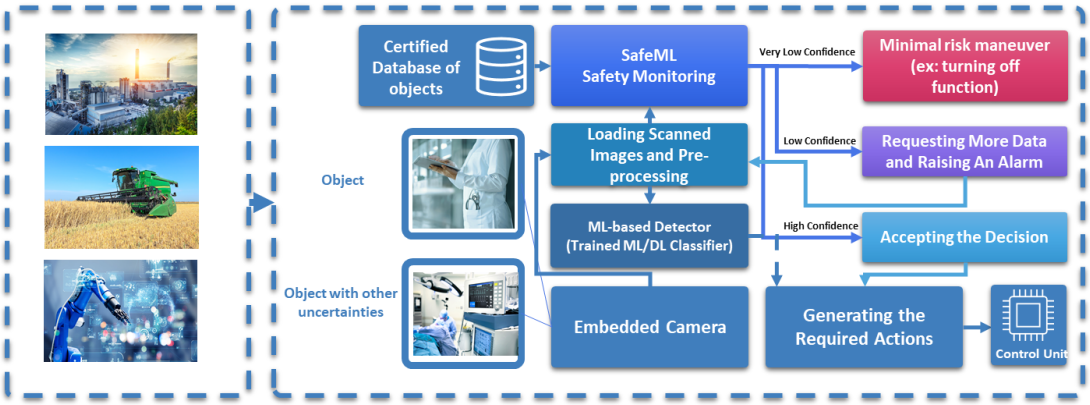


**Figure 20 Application of SafeML**

For scope compliance, the SafeML method proposed in [29] can be used. SafeML is an approach to monitoring the context of the ML component in its operation. The method measures the statistical dissimilarity by computing the Empirical Cumulative Distribution Functions (ECDF) of the concerned data. Figure 20 shows the general application of SafeML involving perception-based ML. The data used at designtime is compared with the run-time data using statistical distance measures to obtain a measure of dissimilarity between the two, thereby obtaining an estimate of scope compliance. This can then be used to assist the decisions reliant on the ML component output.

## 6.3 UNCERTAINTY MONITORING EDDI

By combining SafeML with the uncertainty wrapper shown in Figure 19, we can obtain a holistic uncertainty estimate. Since both of these technologies can be used at run-time, a run-time dynamic uncertainty monitoring can be obtained.

## 6.4 APPLICATION IN SESAME

In SESAME use cases, the perception component can be further strengthened by the use of dynamic uncertainty monitoring. In this section, we discuss further how this concept can be applied in the LOCOMOTEC use case. As a reminder, in that use case, each robot has a Task Execution component which depends on a Person Detection component. The latter component uses ML models fed by onboard cameras in order to detect whether and in which direction people around the robot are detected. Detection of people near enough the robot is the main factor in deciding whether to turn off the robot's UV-C lamps in those directions. Failing to detect people within a short distance of the robot presents a safety risk, which needs to be accounted for during development, and protected against at runtime, with means which can include SafeML.

To apply SafeML for the use case, the datasets used to train the Person Detection ML models to need to be assessed. A subset of the training dataset must be specified, such that it faithfully represents the training dataset. The choice and balancing of this dataset are currently a manual process, involving experimentation and expert judgement.

Once this 'certified' dataset has been formed, some additional evaluation must be performed to establish the number of samples for the runtime phase. The number of samples is an important trade-off, because collecting too few invalidates the tests due to low statistical power, whereas too many samples present a performance bottleneck. Power analysis can be used to identify the minimum number of samples for a chosen effect and confidence level, and the concrete number of samples adjusted according to sampling strategy and performance targets.

At runtime, the SafeML component can be deployed alongside the Person Detection. The component samples from the same camera image set the ML model receives and evaluates the statistical distance between the certified dataset and the sampled. If the test passes, then the ML model answer is trusted and the operation proceeds as expected. If the test fails, then more data can be requested, and if further testing indicates issues, alarms to the robot and MRS application can be raised e.g., to notify human operators to intervene. A visual summary of the above can be seen in Figure 21.

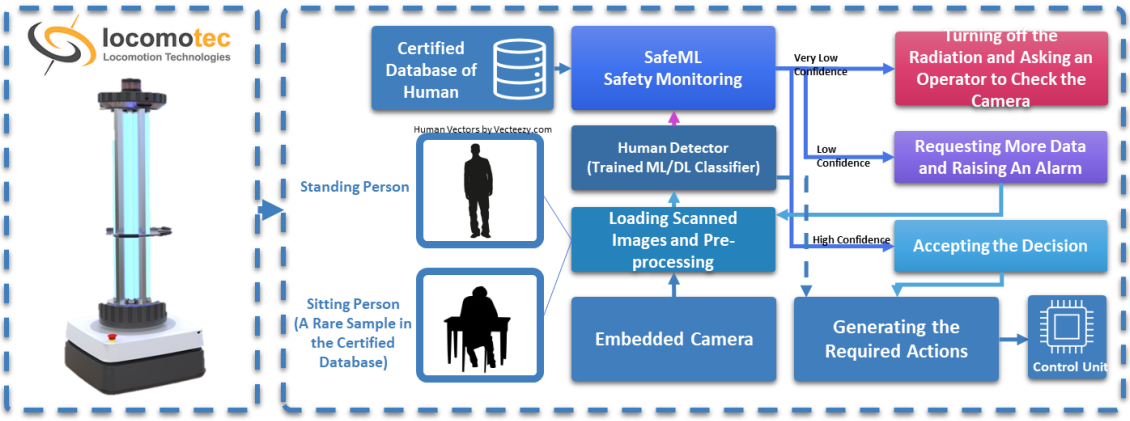


**Figure 21 The use of SafeML on LOCOMOTEC use case of SESAME**

# 7. DYNAMIC EVENT MONITORING

The EDDI is the evolution of the earlier DDI concept. While DDIs were primarily static artefacts created at design time, EDDIs are intended to be executable at runtime, either onboard or alongside their target, in order to perform dynamic dependability management.

This dynamic execution imposes a number of new requirements on the models and algorithms that make up the EDDI. In particular, if they are to react at all to the operational environment, they require some means of receiving information about it (e.g., via data from sensors) and communicating results, conclusions, and recommended responses back to the target robot and/or its operators.

An EDDI is therefore both a repository of knowledge — in form of dependability models and associated parameters — and a means of using that knowledge to interpret input and output received by the EDDI via the target system.

Although there are other aspects involved here (such as the ConSert guarantee & demand exchange mechanism), the primary means of reacting to perceived input is via the **EDDI event monitoring framework.** Following design-time specifications of what events are of interest and what parameters they might need in order to be monitored — such as the component acting as a source for the raw data — generic event monitoring code can be generated. This can then be tailored and adapted for the target platform as necessary so that it can be executed.

The events, once triggered by these monitors, connect to the wider dependability models that make up the knowledge base of the EDDI. In this way, they can activate or inform different processes, e.g., changes in state in a Markov chain or Bayesian network, occurrence of basic events in a fault tree, or evidence to support guarantees offered by a ConSert.

Thus, the events serve as the input for any onboard diagnostic or dynamic dependability evaluation functions. Different conditions can be modelled, while the ability to trigger events on the basis of communication from other agents (EDDI or otherwise) provides support for MRS.

## 7.1 EDDI EVENT FRAMEWORK CONTEXT

The context for how event monitoring fits into the wider EDDI framework is illustrated in Figure 22.

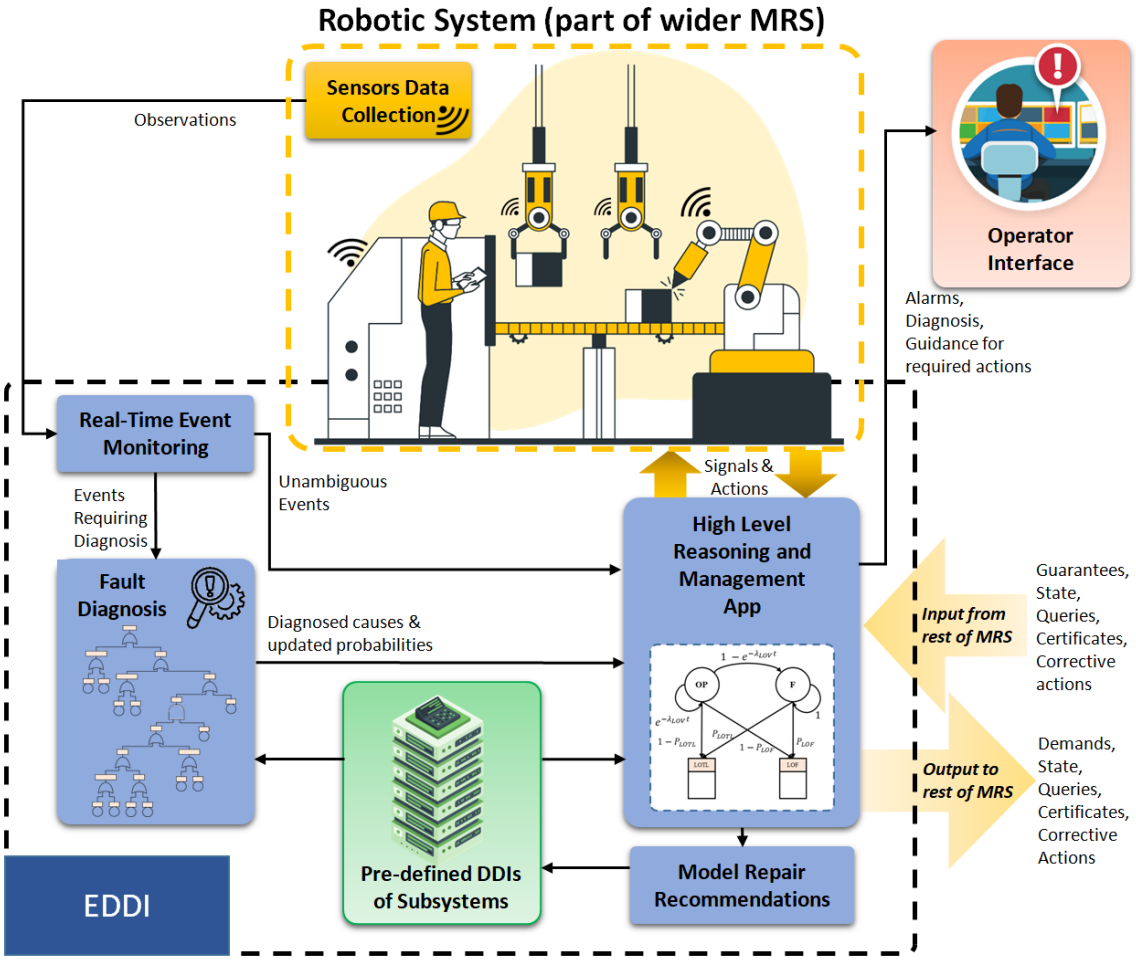


**Figure 22 The basic Executable Digital Dependability Identity architecture**

However, it is important to remember that input is only one-half of the equation; the EDDI also needs to be able to talk as well as listen. To that end, it requires a two-way interface to its target system (e.g. a robot); events provide the input, while **actions** provide the output.

The bi-directional interface is shown in the diagram in two places: the observations (top left) coming from the system's sensors, and the signals and actions communicated directly to the system controller (centre-right). Together, these form the inputs received from the system and outputs the EDDI sends to it.

Through the system inputs, the EDDI receives information about the state of the system and its parameters, e.g. the readings from any onboard sensors that have dependability implications. Sensors also provide the EDDI with information about its operating environment and any prevailing conditions that may impact safety or security (e.g. bad weather). Such information will necessarily be platform-dependent, and most likely in a format dictated by the platform, though some pre-processing of the information may

exist rather than reading e.g. raw sensor data, depending on the nature of the platform's controller.

In return, the EDDI sends to the system information about the dynamic level of risk as well as possible corrective measures. Note that the EDDI is not expected to perform actions directly, but rather issue recommendations to the platform and/or operator; it should not have the authority to take control of the robot itself (and is unlikely to have the platform-specific control capability in any case). However, it could for instance recommend the system enter a safe state, abort its current mission, or that it activates some backup or redundant component.

## 7.2 EVENT MONITORING

The event monitor components of the EDDI perform low-level detection of events. They are responsible for the evaluation of real-time sensory data and determining whether or not particular events of note have occurred (e.g. a fault), and if so, reporting this to the rest of the EDDI.

The exact form of an event monitor is heavily platform-specific, depending as it does on the nature of the data being monitored and the platform itself. However, there are frequent commonalities that can be generalised and so the event monitors can, to some degree, be considered instantiations of specific patterns. For example, sensor data is likely to be buffered into a time series store (e.g. via a circular buffer with shifting time windows), to better identify trends and long-term conditions and help filter out transient phenomena or spurious readings. Expressions to confirm the occurrence of events can be complex operations that involve querying both current and historical data points, and a system of three-valued logic — incorporating an 'unknown' value in addition to 'true' and 'false' — may help in processing situations that involve a degree of uncertainty or otherwise incomplete information.

It is also important to note that EDDIs do not purely monitor for hardware faults. The system component being monitored may be an AI component, for instance, using SafeML or Uncertainty Wrappers to provide information about its current status and dependability of its performance. As described in Section 3, AI components such as DNNs introduce a more probabilistic type of uncertainty as they always have a chance to misclassify an input. Thus, a camera connected to a person detection algorithm, for example, has a chance of not recognising a person in the vicinity of the robot, leading to a hazardous scenario. While we cannot detect this situation with 100% confidence, we may know that the ML component has low confidence (e.g. due to environmental conditions like darkness or bad weather) and adjust behaviour accordingly.

Alternatively, an event monitor may be monitoring for security threats of some description, such as a network intrusion of some kind or a security authentication failure. Both security and AI problems as well as more traditional hardware-related faults are subordinate to the higher-level reasoning of the EDDI so that it can respond to any dependability-related event, whether that be a hardware fault, security threat, or AI performance degradation.

There are three primary components of the EDDI dynamic event framework: Events, Event Monitors, and Actions. Further detail can be found in "D4.2/D5.2 ODE and EDDI Specification" but a brief overview will be provided here.

As mentioned in section 7, the EDDI requires some means of communicating with its host system (e.g. a drone or robot). For example, dynamic risk assessment may require information about the operating environment, while feedback from onboard sensors may be needed for fault diagnosis and failure prediction. Equally, the EDDI communicates whatever conclusions or results it reaches back to the host. In both cases, there must be a way to specify what data is being monitored, how the EDDI reacts to it, and what it does with that information.

The result is the Event-Action cycle (as shown in Figure 23), in which the EDDI obtains input from Event Monitors in the form of Events, processes the Events according to their internal models and algorithms, and then issues Action recommendations accordingly.

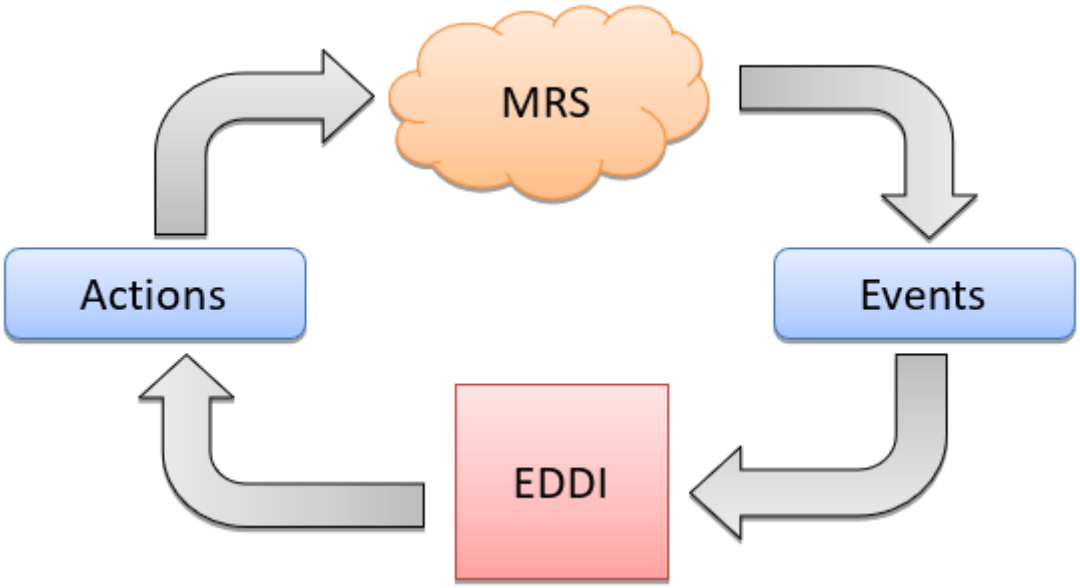


**Figure 23 The Event-Action cycle**

Hence an event-based approach has been adopted to model the way in which an EDDI is able to process information from the host system (since Actions can also be thought of as a form of event, merely a type triggered by the EDDI and issued from it, rather than received by it).

## 7.3 EVENTS

While there are many types of events that may occur during the nominal operation of a given system, for the purposes of the EDDI framework we are only interested in those that bear relevance to dependability. Furthermore, the causes of the events must also be observable — i.e., can be monitored — or there would be no way to know when to trigger them.

Different categories of Event can be distinguished in this context, including (but not necessarily limited to):

- "Condition" or "observed" events, which are conditions that hold over one or more variables/values and which occur when those conditions are observed to be true;

- "Intelligent" or "Machine Learning" events, which are triggered as a result of some kind of decision-making process (such as an ML classification or security intrusion alert) and which may involve some degree of uncertainty;

- "External" events, are essentially messages received from other EDDIs operating externally in the wider context (or, potentially, the same EDDI). This may involve information about failures or problems encountered by other robots in the MRS, for instance.

These three categories form subclasses of the ODE's primary Event class. However, to be relevant at runtime, we also need to know how to monitor them, which is the purpose of the EventMonitor. Having both allows us to keep the concept of the information being monitored (the Event) separate from how it is being monitored (the EventMonitor), meaning the platform-specific details are largely encapsulated in the monitor so that the Event itself can remain fairly generic.

Of the three types of events, external events are the simplest type. These are not monitored as such but occur when a particular message is received by the EDDI's host platform (which may, in turn, have been triggered by an Action from another EDDI on the network). External events carry only the message data, along with some metadata about its source, type and destination.

The "intelligent" events are the ones most closely tied to the specific implementation since they are triggered directly by whatever intelligent, likely ML-based component is involved. For example, a person-detection system may trigger a "person detected" event on the basis of information from multiple sensors (cameras, LIDAR etc) on the basis of black-box algorithms that the EDDI has no knowledge of.

The condition events, however, are the "traditional" events that are monitored and triggered as a result of those conditions becoming true. For this purpose, the ODE defines a particular grammar — essentially a small domain-specific language — for specifying such conditions. This language encapsulates common generic event-monitoring patterns, e.g. allowing timed expressions (= a condition that must hold for a given period) or filtering out anomalous data by performing operations over a range of data (e.g. via a buffer).

For reference, the grammar is repeated below, but further information can be found in D4.2/D5.2. The grammar is defined in EBNF[2] in Listing 1.

[2] https://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_form

```
condition =          timed_exp, {"AND" | "OR" | "XOR",
                     timed_exp };

timed_exp =          bool_exp
                     | "TIME", "(", bool_exp, ",",
                     constant, time_unit, ")";

time_unit =          "ms" | "s" | "m" | "h" | "d";

bool_exp =           bool_comp, {"AND" | "OR" | "XOR",
                     bool_comp}
                     | "(", bool_exp, ")";

bool_comp =          not_exp, [("==" | "!="), not_exp];

not_exp =            ["NOT"], bool_val;

bool_val =           comparison
                     | variable
                     | "EVENT", "(", variable, ")"
                     | bool_constant;

bool_constant =      "TRUE" | "FALSE" | "UNKNOWN";

comparison =         expression, rel_op, expression;

rel_op =             "==" | "!=" | "<" | ">" | "<=" | ">=";

expression =         term, {["+" | "-"], term};

term =               factor, (["*" | "/"], factor};

factor =             (num_val, ["^", num_val])
                     | unary_function
                     | timed_function;

unary_function =     "ABS" | "SQRT", "(", expression, ")";

timed_function =     func_name,
                     "(", variable, ",", constant,
                     time_unit ")";

func_name =          "MAX" | "MIN" | "SUM" | "AVERAGE"
                     | "DIF" | "INTEG";

num_val =            variable | constant |
                     "(", expression, ")";
```

**Listing 1 - EDDI Event EBNF Syntax**

This grammar allows for a variety of expressions, including:

- Boolean expressions, e.g. `X == TRUE AND Y == FALSE`
- Conditions that must hold over time, e.g. `TIME(temp > 100, 5s)`
- Negated conditions, e.g. `X == NOT UNKNOWN`

- Comparisons, e.g. `battery_level <= 100`
- Arithmetic operations, e.g. `battery_level / 1000 <= 50`
- Powers, e.g. `x ^ 2 < 100`
- ABS and SQRT, e.g. `ABS(var) > 5`
- Functions that operate over a range of historical values, e.g. MAX, MIN, SUM, AVERAGE, DIF, INTEG etc. The operand indicates the time period over which values should be used. The precise number of values then depends on the time and the relevant sampling rate.
- Combining all of the above, e.g. `AVERAGE(temp, 5s) > 100 AND TIME(warning == TRUE, 5s)`

Variables are taken to be any combination of numbers and letters, as long as it starts with a letter (and excluding the keywords defined in the grammar), while numeric constants can include numbers in base 10 scientific formats (e.g. 1e-5) and specified with either '.' or ',' as the decimal separator.

To connect variables to their sources, EventMonitors are needed. An EventMonitor indicates both the variable name, the source component of the value in question (e.g. a sensor) and a sampling rate (which can be used in conjunction with the time constants to determine the number of historical values that must be stored in a buffer). It may also specify the data type provided by the source, e.g. real values, integers, or logical values (TRUE/FALSE/UNKNOWN).

Additional complexity is that at runtime we also need to account for the fact that our knowledge is limited. Therefore, in addition to simple true/false conditions, we also need to accept the possibility of unknown results, meaning we will be employing a three-valued logic — TRUE, FALSE, UNKNOWN — instead. UNKNOWN may apply, for example, if a sensor fails to provide a reading, or a function does not yet have enough historical data to provide a result.

To understand how the three-value logic should behave, please consult the truth Table 1 (where T = True, F = False, and U = Unknown):

**Table 1: Truth table of three value logic**

| X, Y | X AND Y | X OR Y | X XOR Y | NOT X |
|---|---|---|---|---|
| **F, F** | F | F | F | T |
| **F, T** | F | T | T | T |
| **F, U** | F | U | U | T |
| **T, F** | F | T | T | F |

| X, Y | X AND Y | X OR Y | X XOR Y | NOT X |
|---|---|---|---|---|
| **T, T** | T | T | F | F |
| **T, U** | U | T | U | F |
| **U, F** | F | U | U | U |
| **U, T** | U | T | U | U |
| **U, U** | U | U | U | U |

# 8. DYNAMIC SECURITY MANAGEMENT

An Intrusion Detection System (IDS) is a security monitoring system that can detect suspicious activities and generates alerts when detection occurs. A system administrator or an incident responder makes use of these alerts to further investigate and define mitigation actions.

The wide adoption of Snort[3] led to its selection as the IDS used in the SESAME security assessment. Such a tool allows for the detection of attacks on the system in question and the creation of corresponding alerts. Snort performs real-time traffic analysis and packet logging. A set of rules defines the malicious network activity and generates alerts for the users.

Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information. On the other hand, the rule options section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken. More information about Snort can be found in section 3.1.1 of D5.3.

Using Snort or alternative IDS systems can be part of a security concept, and also support broader dependability aspects. The configuration of the IDS (via rules or otherwise) should address the corresponding requirements. The work described in SESAME deliverable D4.3 goes further into detail with respect to how this can be realized during development. With respect to this deliverable, focusing on the runtime, an IDS is satisfying an active system security requirement (e.g. "system must detect intrusions"), based on the assessment of the impact identified security threats have on the system's assets (i.e. aspects of the system that need protection). Part of these assets can include dependability-critical elements of the system (e.g. robot CPUs, robot functionality), but also information that is critical with respect to dependability e.g. perceptual information received outside the robot's own sensors.

An intrusion, if left unchecked, can potentially jeopardize such assets, and lead to unsafe, or disabled MRS applications. Therefore, as part of the EDDI, the IDS is

[3] https://www.snort.org/

responsible for detecting intrusions and propagating alarms to other EDDI runtime components, e.g. to perform dynamic capability (section 3) and/or reliability assessment (section 4), to update their evaluation and potentially trigger individual robot and/or MRS adaptation. An example of the above workflow from development to runtime can be seen in Figure 24. On the left side of the figure, safety and security assessment can be combined to provide input for specifying IDS rules. 'Dynamic Dependability Engineering' represent development activities for setting up e.g. Dynamic Safety Capability Assessment and Dynamic Reliability Assessment. They result in ConSerts and SafeDrones models which are aware of which kinds of alerts and other information IDS can propagate at runtime. Once deployed, the right side of the figure depicts how, upon detecting an intrusion based on a specific rule, an IDS can propagate the alert to either or both corresponding runtime EDDI components, depending on the relevance to safety and/or reliability. Based on the context, each of the EDDI components can recommend adaptations to the MRS or individual robot.

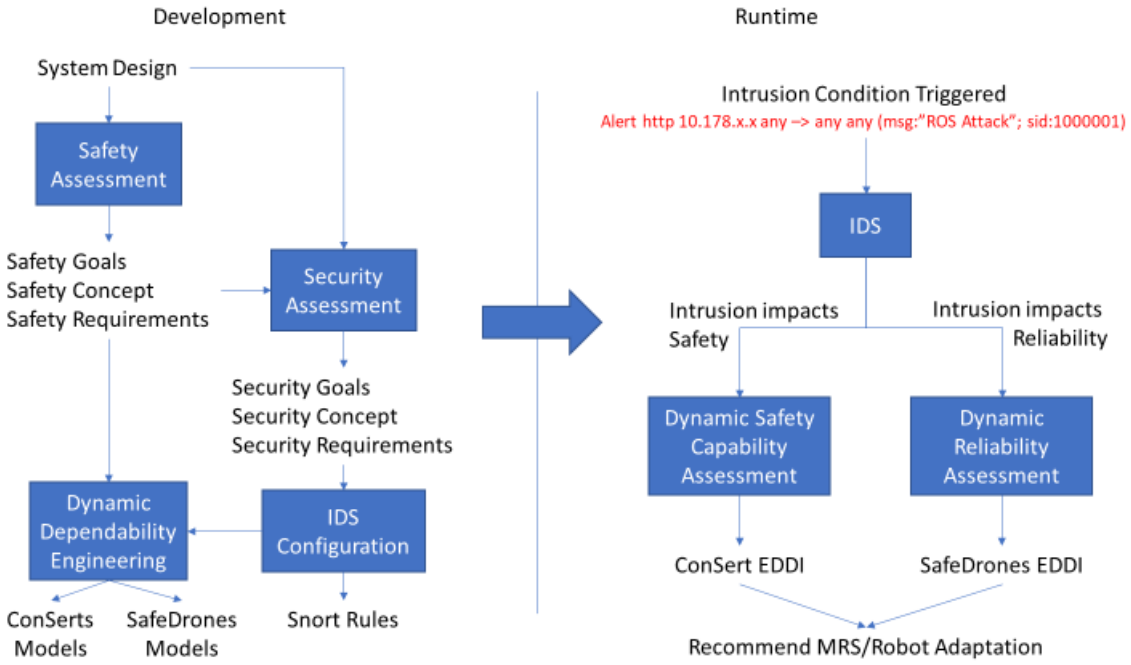


**Figure 24 Example workflow of IDS as part of runtime EDDIs**

# 9. CONCLUSIONS

Multi-Robot Systems (MRS) present both a challenge in terms of complexity, intelligence, and autonomy, but also an opportunity to overcome this challenge by advancing existing ideas and technologies into the runtime domain. This transition is also reflected in the corresponding shift from the existing concept of Digital Dependability Identities (DDIs) into Executable DDIs (EDDIs).

In the present deliverable, we elaborated further on how our vision of runtime EDDIs looks like (sections 1 and 2), and discussed what we believe to be fundamental building blocks for composing effective support for MRS applications (sections 3 to 8). Our ongoing work follows the concepts outlined here and is directly operationalized in D7.2, where tooling for generating runtime EDDI components is discussed in more detail. Our

next step in this regard are to address and deploy our runtime components in MRS, which is handled through the activities of task T7.3, and applied in practice on the SESAME use cases.

Our concept also includes a preliminary approach for establishing a basis for communication across EDDIs and their host MRS, through a system of runtime event monitoring and action recommendation. This is described further in section 7.

Another point that requires further development as part of the SESAME project is increasing the scope of our activities from the level of individual robots, to the level of the MRS application. Mission-level runtime EDDIs can be positioned to have higher-level awareness of the operational context and thus the ability to recommend larger-scale adaptations to achieve missions, or at least offer additional layers of protection for their dependability.

With respect to security at runtime, we briefly discussed our concept for integrating active security measures (specifically an Intrusion Detection System – IDS) with other runtime EDDI components (section 8). To arrive at this integration, an integrated dependability engineering approach is required to set up the EDDI components that will be used at runtime. The runtime EDDI components need to be aware of the types of alerts the IDS can trigger that are relevant to the dependability properties of concern e.g. safety and reliability. Given this setup, runtime EDDI components can also incorporate IDS alerts, and recommend MRS and/or robot behaviour adaptation accordingly. At this point, this concept still requires further technical development and is part of our next steps in the SESAME project.

# 10. REFERENCES


[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41-50, 2003.

[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] D. J. Snowden and M. E. Boone, "A leader's framework for decision making," *Harvard business review*, vol. 85, no. 11, p. 68, 2007.

[4] D. Schneider and M. Trapp, "Conditional Safety Certification of Open Adaptive Systems," *ACM Trans. Auton. Adapt. Syst. (ACM Transactions on Autonomous and Adaptive Systems)*, vol. 8, no. 2, pp. 1-20, 2013.

[5] D. Schneider, "Conditional Safety Certification for Open Adaptive Systems," Technical University Kaiserslautern, 2014.

[6] K. Sohag, K. Aslansefat, I. Sorokos, Y. Papadopoulos and Y. Gheraibia, "A conceptual framework to incorporate complex basic events in HiP-HOPS," in *International Symposium on Model-Based Safety and Assessment*, Thessaloniki, 2019.

[7] M. Ottavi, S. Pontarelli, D. Gizopoulos, C. Bolchini, . M. K. Michael, L. Anghel, M. Tahoori, A. Paschalis, P. Reviriego, O. Bringmann, V. Izosimov, H. Manhaeve, C. Strydis and S. Hamdioui, "Dependable multicore architectures at nanoscale: The view from europe," *IEEE Design & Test*, vol. 32, pp. 17-28, 2014.

[8] K. Aslansefat, F. Marques, R. Mendonça and J. Barata, "A markov process-based approach for reliability evaluation of the propulsion system in multi-rotor drones," in *Doctoral Conference on Computing, Electrical and Industrial Systems*, 2019.

[9] J. Eggert, "Risk estimation for driving support and behavior planning in intelligent vehicles," *at - Automatisierungstechnik*, vol. 66, no. 2, p. 119–131, 2018.

[10] M. Machin, J. Guiochet, H. Waeselynck, J.-P. Blanquart, M. Roy and L. Masson, "SMOF: A Safety Monitoring Framework for Autonomous Systems," *IEEE Trans. Syst. Man Cybern, Syst. (IEEE Transactions on Systems, Man, and Cybernetics: Systems)*, vol. 48, no. 5, p. 702–715, 2018.

[11] C. Pek, S. Manzinger, M. Koschi and M. Althoff, "Using online verification to prevent autonomous vehicles from causing accidents," *Nat Mach Intell (Nature Machine Intelligence)*, vol. 2, no. 9, pp. 518-528, 2020.

[12] S. Shalev-Shwartz, S. Shammah and A. Shashua, "On a Formal Model of Safe and Scalable Self-driving Cars," Intel/Mobileye, http://arxiv.org/pdf/1708.06374v5, 2017.

[13] M. Trapp, D. Schneider and G. Weiss, "Towards Safety-Awareness and Dynamic Safety Management," in *14th European Dependable Computing Conference (EDCC)*, 2018.

[14] C. Hartsell, S. Ramakrishna, A. Dubey, D. Stojcsics, N. Mahadevan and G. Karsai, "ReSonAte: A Runtime Risk Assessment Framework for Autonomous Systems," in *16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2021.

[15] S. Khastgir, H. Sivencrona, G. Dhadyalla, P. Billing, S. Birrell and P. Jennings, "Introducing ASIL inspired dynamic tactical safety decision framework for automated vehicles," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2017.

[16] R. Johansson and J. Nilsson, "The need for an environment perception block to address all ASIL levels simultaneously," in *IEEE Intelligent Vehicles Symposium*, Gotenburg, Sweden, 2016.

[17] P. Feth, "Dynamic Behavior Risk Assessment for Autonomous Systems," Germany, 2020.

[18] M. Schreier, V. Willert and J. Adamy, "An Integrated Approach to Maneuver-Based Trajectory Prediction and Criticality Assessment in Arbitrary Road Environments," *IEEE Trans. Intell. Transport. Syst. (IEEE Transactions on Intelligent Transportation Systems)*, vol. 17, no. 10, pp. 2751-2766, 2016.

[19] S. Lefevre, C. Laugier and J. Ibanez-Guzman, "Intention-Aware Risk Estimation for General Traffic Situations and Application To Intersection Safety," [Research Report] RR-8379, INRIA, 2013.

[20] J. Dahl, G. R. d. Campos, C. Olsson and J. Fredriksson, "Collision Avoidance: A Literature Review on Threat-Assessment Techniques," *IEEE Trans. Intell. Veh. (IEEE Transactions on Intelligent Vehicles)*, vol. 4, no. 1, pp. 101-113, 2019.

[21] S. Lefévre, D. Vasquez and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal 1*, pp. 1-14, 2014.

[22] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Kramer, C. Gutenkunst and E. Böde, "Criticality Metrics for Automated Driving: A Review and Suitability Analysis of the State of the Art,"

https://arxiv.org/abs/2108.02403, 2021.

[23] J. Reich and M. Trapp, "SINADRA: Towards a Framework for Assurable Situation-Aware Dynamic Risk Assessment of Autonomous Vehicles," in *16th European Dependable Computing Conference (EDCC)*, Munich, Germany, 2020.

[24] J. Reich, M. Wellstein, I. Sorokos, F. Oboril and K.-U. Scholl, "Towards a Software Component to Perform Situation-Aware Dynamic Risk Assessment for Autonomous Vehicles," in *Dependable computing - EDCC 2021 Workshops. DREAMS, DSOGRI, SERENE 2021*, 2021.

[25] M. Wellstein, "Development of a Bayesian Network for Situation-Aware Lane Change Prediction based on the highD Dataset," Technical University Kaiserslautern, 2021.

[26] SESAME, "D1.2 Evaluation Plan," EC Distributions, 2021.

[27] M. Kläs and A. M. Vollmer, "Uncertainty in Machine Learning Applications: A Practice-Driven Classification of Uncertainty," in *SafeComp Workshop First International Workshop on Artificial Intelligence Safety Engineering WAISE*, Västerås, 2018.

[28] M. Kläs and L. Sembach, "Uncertainty Wrappers for Data-driven Models: Increase the Transparency of AI/ML-based Models through Enrichment with Dependable Situation-aware Uncertainty Estimates," in *SSAFECOMP workshop International Workshop on Artificial Intelligence Safety Engineering (WAISE)*, Turku, 2019.

[29] K. Aslansefat, I. Sorokos, D. Whiting, R. Tavakoli Kolagari and Y. Papadopoulos, "SafeML: Safety Monitoring of Machine Learning Classifiers through Statistical Difference Measure," in *International Symposium on Model-Based Safety and Assessment*, Lisbon, 2020.