SESAME
SECURE AND SAFE MULTI-ROBOT SYSTEMS

Project Number 101017258

D7.3 Runtime Safety and Security Concept – EDDI-based MAS and Communication

Version 1.0
4 July 2023
Final

Public Distribution

Fraunhofer IESE, University of Hull and FORTH

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

DOCUMENT CONTROL

Version	Status	Date
0.1	Initial outline	10 May 2023
0.6	Internal review version ready	22 June 2023
0.8	Internal review from FORTH integrated	3 July 2023
0.9	Internal review from YORK integrated	3 July 2023
1.0	Review changes integrated, QA, ready for submission	4 July 2023

TABLE OF CONTENTS

TABLE OF FIGURES

EXECUTIVE SUMMARY

Executable Digital Dependability Identities (EDDIs) are meant to be deployed across significantly diverse applications and complex Multi-Robot System (MRS) architectures. In this deliverable, we discuss how RunTime (RT) EDDIs can be designed and deployed as Multi-Agent Systems (MAS), to collaborate towards MRS dependability assurance during operation.

In environments featuring the Robotic Operating System (ROS) communication platform, such MAS can be realised through the deployment of agents on ROS nodes. In non-ROS environments, we have developed a generic runtime component which integrates with several popular web-based interfaces and orchestrates the execution of RT EDDIs.

To do so, we build upon the runtime dependability concept described in SESAME deliverable D7.1 and realised through the RT EDDI code generators described in D7.2. While we include work-in-progress examples across three SESAME use cases, further evaluation for each use case is planned for the remainder of the project in upcoming WP8 deliverables.

LIST OF ABBREVIATIONS

EDDI	Executable Digital Dependability Identity	ODE	Open Dependability Exchange
ODD	Operational Design Domain	ROS	Robot Operating System
ConSerts	Conditional Safety Certificates	XML	Extensible Markup Language
YAML	Yaml Ain't Markup Language	EGL	Epsilon Generation Language
SESAME	Secure and Safe Multi-Robot Systems	MRS	Multi Robot System
JAR	Java Archive	API	Application Programming Interface
XDSL	XML-based Bayesian Network format of the GeNIe toolset	GeNie	Commercial tool for Bayesian network modelling and inference
IDE	Integrated Development Environment	DT	Design Time
PCA	Principle Component Analysis	Hz	Hertz
EBNF	Extended Backus–Naur Form	RtE	Runtime Evidence
MROS	Model Based ROS	RT	Runtime

1. INTRODUCTION

In this deliverable, we discuss the application of Runtime Executable Digital Dependability Identities (RT EDDIs) as Multi-Agent Systems (MAS) for supporting dependability in Multi-Robot Systems (MRS). The MAS abstraction is a convenient medium for expressing, developing, and deploying RT EDDIs for the above purpose. We elaborate further on how this approach can be applied in the later sections of the deliverable.

We reproduce parts of previous and other deliverables throughout, to make this report relatively self-contained. Naturally, more details concerning concepts from other deliverables are best presented in said deliverables. A recap of what preceded this deliverable follows.

The conceptual background of the constituents of an EDDI as envisioned by SESAME are described in deliverable D7.1 (see Figure 1). Concretely, these constituents collaboratively performing dynamic risk management are: 1. Dynamic Safety Capability Assessment with Conditional Safety Certificates (*ConSerts*), 2. Dynamic Risk Assessment with Situation-Aware Dynamic Risk Assessment (SINADRA) using Bayesian Networks, 3. Dynamic Reliability Assessment based on the S*afeDrones* tool, 4. Perception Uncertainty Monitoring with the *SafeML* tool and 5. conditional event monitoring to realize a typed communication interface between the *runtime* EDDI and the nominal functionality of the MRS. Regarding the scope of the tools described in deliverable D7.2, the models consumed by these components are assumed to be present in the shape of a *design-time* EDDI, which is an .xml file conforming to the technical Open Dependability Exchange (ODE) metamodel specification described in D4.2. Thus, the toolset in deliverable D7.2 realizes the pipeline to make the engineered runtime models executable (i.e., generate components that can infer the runtime models based on runtime-available information) and deploy them into common robotic platform architectures.

The Robot Operating System (ROS) has been selected as the exemplary target runtime environment, to which the EDDI shall be deployed in. The reason for this decision is the usage of ROS in several SESAME use cases and the general spread of ROS in the robotics domain. Thus, by having support for deploying runtime EDDIs to ROS applications, the transfer of EDDIs to industrial applications is facilitated. By splitting the runtime EDDI generator pipeline into platform-independent and platform-dependent parts, the extension towards other runtime environments is conceptually and technically simplified.

Specification of RT EDDI models for SESAME use cases is also included, focusing on the design and deployment insights.

In the previous SESAME deliverable D7.2, a set of tools for generating and deploying RT EDDIs onto ROS-based MRS was already provided. In this deliverable, the developed tool provides users with the ability to deploy in non-ROS-based MRS as well. The new and updated tools (and use-case-specific artifacts described in this deliverable can be found in the SESAME public GitHub repository at: https://github.com/sesame-project/runtime_eddis.

Further details on the relationship of the deliverable with others are as follows, also summarised in Figure 1.

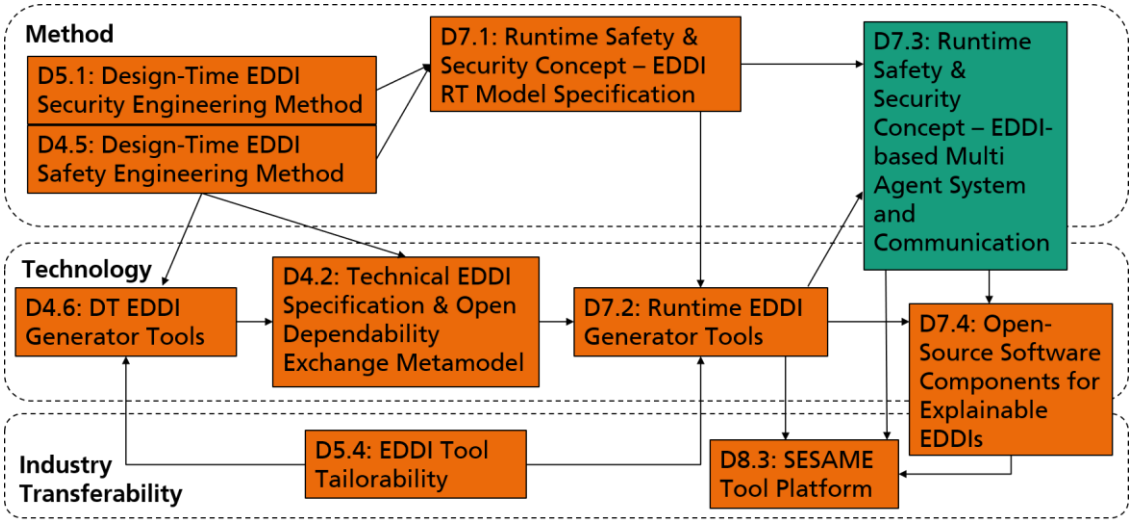


Figure 1 Relationship of D7.3 to other deliverables

The rest of the deliverable is structured as follows. In Section 2, we discuss how the RT EDDIs relate to MAS and what are the benefits. In Section 3, we discuss how MRS missions can be restructured into MAS. In Section 4, we review our RT EDDIs and discuss their deployment as MAS. In Section 5, we provide work-in-progress examples of applying RT EDDIs for SESAME use cases. In Section 6, we discuss the limitations we have observed with our current approach. We conclude in Section 7, summarizing the deliverable's main points and outlining the next steps.

2. RUNTIME EDDIS AS MULTI-AGENT SYSTEMS

Recall that our overall goal is to assure dependability in MRS. RT EDDIs are runtime components responsible for assessing and controlling risk at both the individual robot and MRS level. In this section, we review definitions of MAS, explain how they apply towards RT EDDIs, and consider how this supports our overall goal stated above.

2.1 MULTI-AGENT SYSTEMS DEFINITIONS

In the context of Artificial Intelligence (AI), agents can be defined as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators" [1, p. 34]. Agents can perceive base information from their subordinate sensors, update or infer new knowledge, and apply actions back onto their environment. This cycle of interaction is also captured in the well-established Monitor-Analyse-Plan-Execute-Knowledge paradigm [2].

The agents' environments can be simulated, digital (e.g., network) or physical. In physical environments, this means that agents perceive physical stimuli (e.g., temperature) or derivatives thereof, whereas in digital environments they may perceive e.g., network information, agent statuses, or other digital events. Simulated environments can themselves refer to digital, physical, or hybrid environments, where the agents may interact with both the digital and physical environments the host system establishes.

The agent definition can be further extended to MAS as "multiple agents which collaborate to solve a complex task" [3]; the authors further classify MAS as a subcategory of AI, separate from Parallel AI and Distributed Problem Solving. All the categories have in common a distribution of computation; the difference lies in how each category frames the subject problem and the corresponding solution abstractions.

MAS agents can be purely software, purely hardware, or both aka a cyber-physical system, which includes robots.

For our purposes, MAS benefits, which have already been established in the literature, include parallelisation of computation, redundancy, and specialisation; for instance, see [4] [5] [6].

2.2 VIEWING RUNTIME EDDIS AS MAS: MOTIVATION AND BENEFITS

As explained in the previous section, the agent and MAS definitions are quite broad [7]; they impose few limitations on the purpose, algorithms, or other technical details involved. Therefore, it is important to consider why the MAS paradigm can be useful for exploiting RT EDDIs for dependability in MRS.

To begin with, MAS offer flexibility in encompassing a broad domain of applications. This is a useful property, as MRS domains of applications are remarkably diverse as well; for instance, SESAME alone features use cases from industrial manufacturing, battery assembly, viticulture, healthcare logistics and civil defence. When trying to achieve dependability, one must consider all sources of relevant risk. Since dependability itself encompasses many distinct aspects (including safety, security, reliability, maintainability etc.; see [8]), correctly identifying sources of risk and then designing appropriate controls requires both domain knowledge regarding the application as well as the corresponding dependability-related discipline e.g., safety. Organisationally, this typically involves bringing together experts from the domain with experts in dependability over several iterations of risk identification, analysis, and mitigation specification. While, in theory, the experts do not need to be separate individuals, particularly when assessing and verifying risk-related requirements, maintaining independence of views is known to avoid the dangers of cognitive bias.

However, these processes also limit the capacity of transferring systems and/or components from one domain or application to another. Particularly for domains with strict requirements on dependability, e.g., safety-critical (automotive) and/or mission-critical (space), transfer and reuse should not happen without careful re-engineering, as critical assumptions no longer apply. A key benefit from model-based approaches in this sense is the ability to establish semantic interfaces which can serve to decouple concerns across concepts and implementation. This is realised by separating the 'nominal' functionality from the dependability-related functionality. This looser coupling does not guarantee frictionless transfer, but ideally helps streamline such processes, alleviates effort, and highlights incompatibilities earlier. MAS offer another useful abstraction for modelling and deploying dependability-related components that address MRS dependability.

Specifically, RT EDDIs can be considered as MAS, as they are:

- Independent entities which consume sensor (or derivative) inputs that are relevant to risk and act as input for action recommendations to the host robot. The recommendations they make are not necessarily dependent on the other host's plans. However, they are not autonomous in the sense that, since it is the host system that must account for the recommendations, their actions' execution is ultimately outside of their own control.

- RT EDDIs can be developed and deployed as distributed entities, using communication networks to interact with the host application. In MRS, ROS would be a typical choice for this purpose, but other options can also be considered, especially when communicating with external systems. For example, an agent may communicate via ROS with the host robot, and via a RESTful API with an external online service for collecting cloud information.

- RT EDDIs can be individually engineered using domain expertise in conjunction but loosely coupled with the host robot. This approach aligns with the development process compliant with standards from more traditional dependability-related domains e.g., transport safety.

- RT EDDIs can be structured hierarchically, e.g., see ConSerts and SINADRA Bayesian Networks in sections 4.1 and 4.2, but also heterarchically i.e., providing the host with independent sources of information, in which case conflict resolution across those agents would lie with the host. This property can prove useful for missions where there are conflicting interests across robots. For instance, a robot attempting to search for and rescue people in distress in a disaster-stricken area will have to resolve a conflict when it needs to consider taking over the tasks or rescuing another robot that has malfunctioned. A decision on how to resolve such conflicts can be established top-down (e.g., predefined policy to always prioritise completing own tasks) or bottom-up (e.g., each robot estimates overall risks of each decision and makes a utility-based expectation-maximisation choice) or a hybrid approach is used (e.g., estimated risks are weighed against the policy).

Finally, as is discussed in the following section, MAS-oriented design integrates with the design-time to runtime pipeline introduced in deliverable D7.1 and supported with tools described in deliverable D7.2. This yields an overall workflow that is compatible with established standards-compliant practice in other dependability-critical domains e.g., safety and security for transport, critical infrastructure, and healthcare systems.

3. DESIGNING RT-EDDI-BASED MAS

MRS mission structures have previously been touched upon in SESAME deliverables D2.1 (see sections 3 and 4) and D3.1 (see section 3). Each of the previous deliverables deal with aspects of (roughly) specifying or assessing the nominal functionality for MRS. Both processes also account for dependability assurance input, so it is expected that several iterations of dependability assurance (see SESAME D4.1) would occur over the MRS development before final deployment into operation happens.

As part of this development iteration, it is expected that the mission specification has already described how the mission goals have been decomposed into tasks associated

with specific robots or robot roles. Existing systematic approaches already exist for this purpose, for instance see [9] [10] [11] [12] [13]. Due to the heterogeneity of SESAME use cases, concerning the mission specification approach of each use case owner, and the scope of WP7 not including mission specification, we chose to not impose a specific mission decomposition approach. However, we observe that there could be significant benefits in doing so, particularly for improved automation and traceability to RT EDDI deployment, at least to the extent that WP7 is concerned. Recall that RT EDDIs are also generated during development using DT EDDIs; this concept is discussed in SESAME deliverable D7.1 and tool support developed in D7.2.

RT EDDIs are runtime components deployable directly on robots, or centralised e.g., in a base station. Thanks to their composability, the scope of a given RT EDDI can be restricted to an individual component of a robot, a collection of components, a robot itself, or a group of robots, possibly up to the entire MRS. This is indicated in Figure 2; the RT EDDI corresponding to each abstraction level fundamentally assesses the dependability with respect to that level and uses lower levels to answer this question.

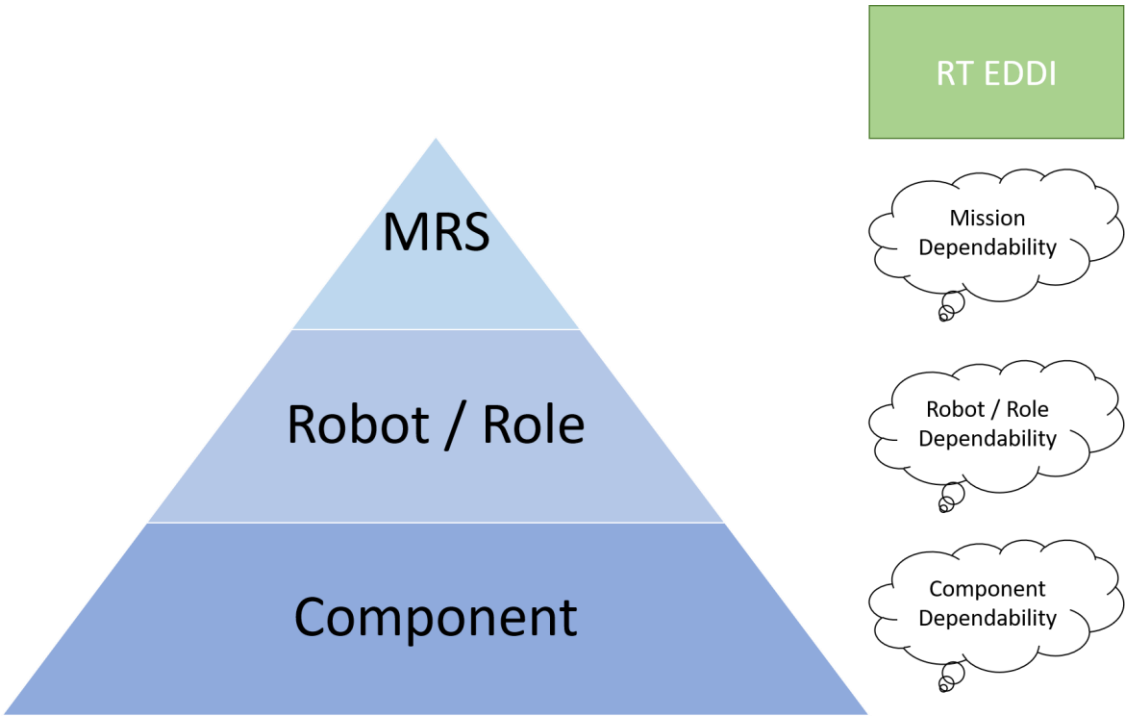


Figure 2 RT EDDI Scope per MRS Abstraction Level

Another dimension to consider is the deployment of RT EDDIs, which does not need to match their scope. For example, if the MRS developer chooses (e.g., due to performance considerations) to deploy RT EDDIs centrally, they could do so e.g., in a robot base station. Two conceptual possibilities for this in the context of a drone MRS are described in Figure 3. In the first case (left side of the figure), a base station (optionally linked to cloud services for extended information) is querying a set of RT EDDIs to plan its upcoming MRS tasks. In the second case (right side of the figure), each MRS robot (drone) is querying its own EDDI to plan itself.

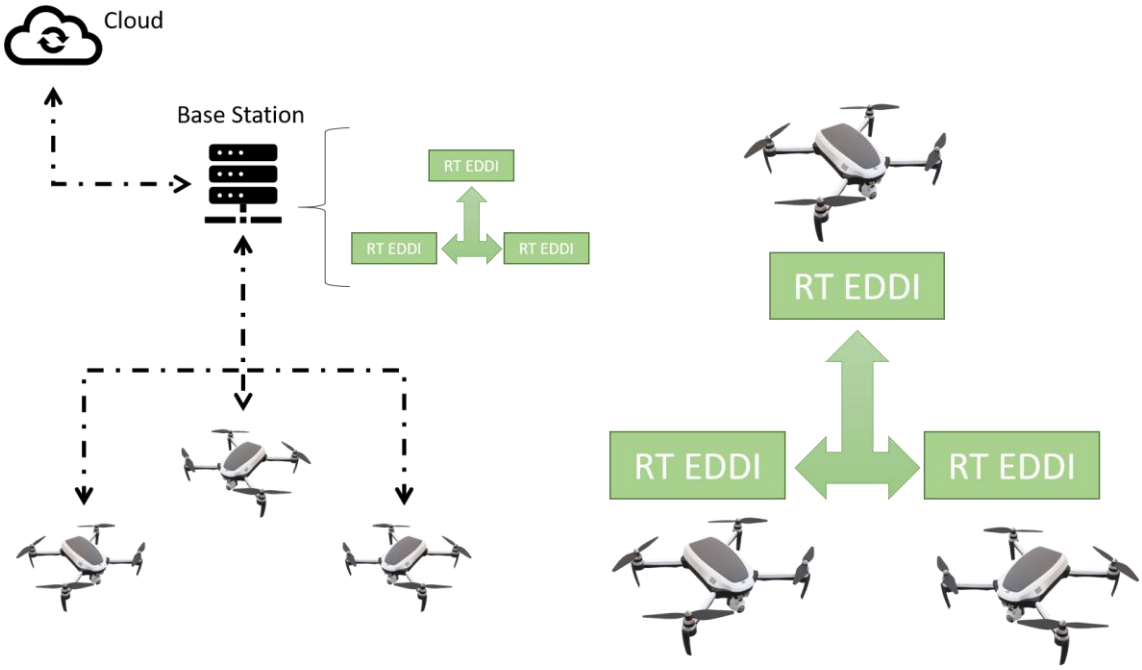


Figure 3 Two conceptual deployment examples (a-left) deployed in base station (governing drones) (b-right) deployed in each drone

At this point, it is important to clarify that, since RT EDDIs address dependability properties rather than nominal function, their behaviour is not intended to directly realise MRS tasks and missions. Rather, their role is to provide additional runtime constraints for robot and MRS-level decision elements to consider for the continuation or adaptation of the mission. RT-EDDI-based MAS share the same view; their role is to enable flexible interfacing and orchestrating different RT EDDI operations.

In this sense, a given agent can be considered as a generic wrapper component which can host an RT EDDI component within it. It integrates with the host robot/MRS, treating it as its environment, and perceives EDDI Events and Actions (see SESAME deliverable D4.1 and D4.2) that are shared within said environment. It uses its embedded RT EDDI to infer its own Actions, and then shares them with other agents or the host robot/MRS. This scheme is depicted in Figure 4. In practice, given the prevalence of ROS, agents can be implemented as ROS nodes which evaluate their RT EDDI periodically, and subscribe/publish EDDI Events/Actions, respectively.

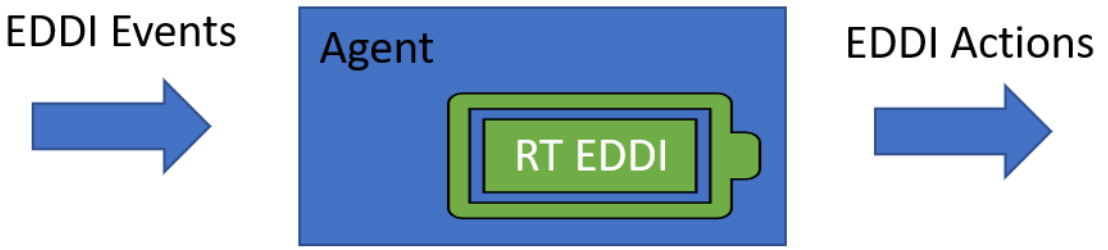


Figure 4 RT EDDI as Agent

An RT EDDI MAS orchestrates multiple agents within a given robot or MRS. This is 6summarized in Figure 5. It is expected that some agents may consume input directly from base or intermediate robot sensors, in which case they are indirectly perceiving the

robot's own environment. Actions propagated by agents can include analytics or robot adaptation recommendations. Note that such a recommendation actually falls outside the scope of RT EDDI models and needs to be customised to the given MRS use case.

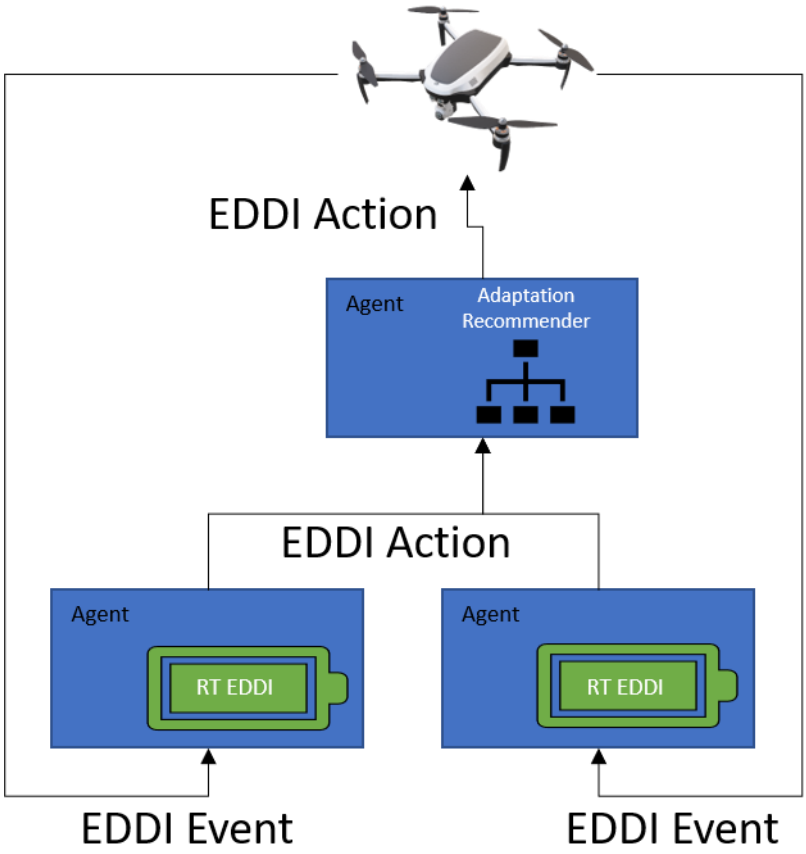


Figure 5 RT EDDI as MAS

4. DEPLOYING RT EDDIS AS MAS

In this section, we review each of the runtime components (already introduced in deliverables D7.1 and D7.2) constituting RT EDDIs and discuss how they can be developed and deployed as MAS.

4.1 RISK CONTROL CAPABILITY

There are two main approaches to improve MRS with ConSerts-based MAS:

a) Each agent evaluates their current dependability via ConSerts and provides guarantees with respect to their mission and capabilities. These guarantees can be sent to a base station that collects guarantees of all agents and then evaluates which agents cannot accomplish their mission by themselves. In other words, the base station can evaluate the risk of mission failure. The base station can then determine further actions through e.g., reconfiguring the mission of other agents to support the agent with insufficient guarantees. For example, a drone must cover a specific area but, during the mission, a rotor fails, so the drone must abort the mission. The drone cannot provide a sufficient guarantee any longer. The base station could then reconfigure a nearby drone, in case sufficient resources are available, to take over the uncovered area of the failed

drone. In this approach, the MAS consists of ConSerts that are distributed between the different drones and the base station.

b) Another approach to support MAS with ConSerts does not rely on a base station. An agent evaluates itself via ConSerts whether it can cooperate with another agent based on the provided guarantees and demands. Platooning, for example, is only possible with communication between the front and the following vehicle because actions such as braking must be performed with minimal latency. To engage in such a cooperation, the front vehicle must fulfil various demands of the following vehicle. Based on these guarantees, the following vehicle can evaluate a distance that maximizes the performance while ensuring safety. For example, the following vehicle could rely on a specific communication technology that must be provided by the front vehicle. Only when the front vehicle provides this technology is the demand fulfilled, and the following vehicle can then engage the cooperation safely.

4.2 SITUATION DYNAMIC RISK ESTIMATION

Situation-aware dynamic risk assessment (SINADRA) aims at optimizing the system performance without deteriorating the system safety. This is achieved through assessing the risk variability of a situation dynamically at runtime to relax from the worst-case assumptions that are taken during design time in the safety engineering process. Thereby, the system can behave in an efficient way tailored to the specific situational risk not depending on the worst case alone. In the SINADRA method, a qualitative model representing causal input features influencing the targeted risk variability is built up and then transformed into a quantified Bayesian network as a runtime representation. For building up such a model, profound knowledge about the system (or the MRS) itself, its purpose and the targeted operational context are required. Further, a previously performed safety analysis to detect possible risk variabilities is needed. The modelled Bayesian network is representing the SINADRA RT EDDI. More details about the SINADRA method and the generated EDDI can be found in the previous SESAME deliverable D7.1.

Regarding the MAS aspects, the link from the SINADRA RT EDDI to other agents is given by the provision of relevant input and the consumption of the SINADRA output. For inputs there are several options that are listed in the following.

- Other RT EDDIs, e.g., SafeML, can provide relevant input information about the environment that is then consumed by the SINADRA RT EDDI.

- Also, there is the option that human operators act as an “agent” in the MAS by defining the operational domain of the MAS in case of different deployment environments, thereby, providing input for other agents like the RT EDDIs. This can help specifically for defining invariant inputs for an operational context that are not easily perceivable by the default sensor suit of a robot.

Besides those agents providing inputs, physical sensor-related components can perceive the environment and are one of the main sources to provide inputs, e.g., by individual sensor data processing or by the fusion of multiple sensor values. Further, each input could be provided with an additional (un)certainty estimate to explicitly handle and propagate uncertainties in the system.

The output provided by the SINADRA RT EDDI is typically consumed by another agent that reasons about adaptation actions to take a safe and informed action so that the robot reacts appropriately to a certain situation and its related risk. For this other RT EDDIs, e.g., the ConSert RT EDDI, can provide additional valuable input which is highly related to the SINADRA output.

With respect to the MRS aspects, the link to the SINADRA RT EDDI is quite similar to the MAS aspects, however, on a system-of-systems scope instead of a single system scope. Inputs can be (collectively) provided by other robots as well:

- Other robots in the surrounding can provide information about the environment that may otherwise not be available to the single "ego" robot for which the SINADRA model is inferred. This collective perception could help to cope with occlusions for instance, providing information about areas not visible by the "ego" robot. This can help to build up a more complete representation of the robot's environment.

- Information from multiple robots perceiving the same environment/situation can be fused together to increase the confidence in the perceived entities, respectively, perceived environmental cues. Again, this is a kind of collective perception, however, in that case there is the caveat of falsely assuming a too high confidence which can lead to false decision-taking, introducing safety issues. Therefore, this option must be carefully considered, and the systems must be analysed for possible dependent faults leading to false perceptions.

The adaptation, meaning the action-decision logic, consuming the SINADRA output could decide upon appropriate MRS action in addition to the single-robot actions, as discussed above in the scope of the MAS. This way, the MRS performance can be optimized safely in a given situational context or the completion of missions that would otherwise be infeasible for the MRS. Regarding the deployment, such an adaptation component could either be running on a single robot, which is then communicating and coordinating the MRS action of other robots, or equivalently on a central decision/planning station.

4.3 RELIABILITY ESTIMATION

Reliability estimation of the components of MRS can be used to dynamically adjust the load and tasks of the MRS participants. In SESAME, a reliability modelling approach called SafeDrones has been already proposed, see SESAME deliverable D7.1. The approach creates a model-based evaluation of dependability for MRS, which enables runtime reliability and risk assessment. By using the realtime reliability estimates, MRS can dynamically adjust mission in accordance with the realtime risk information. In this project, the main focus of the proposed reliability estimation technique is the KIOS use case [14]. The approach has been named after the fact it has been initially developed for the KIOS use case featuring drones but can be applied in non-drone contexts as well. The details of the approach specifics for the KIOS use case can be found in section 5.1.3.

4.4 INTRUSION DETECTION SYSTEM

An Intrusion Detection System (IDS) is a monitoring system that generates alerts when it detects suspicious activities related to communication from and towards a system in question. These alerts are then communicated to a system administrator or incident responder for further investigation and determination of mitigation actions.

Snort, a signature-based IDS was selected as the preferred tool for detection of conducted attacks towards the systems of the SESAME use cases. Snort's extensive usage and recognition along with the familiarity of the authors with it were the main reasons for its adoption. Snort is a widely recognized open-source IDS and has been extensively documented in various studies. It utilizes a collection of rules to define and detect malicious network activity. Whenever a rule is triggered, an alert is generated.

Snort uses a rule language that is highly versatile, enabling the description of the network traffic to be captured and the subsequent remedial actions. A rule comprises two main components: the rule header and the rule options. The header contains essential information such as the rule's action, protocol, source, and destination IP addresses along with their netmasks, as well as the source and destination ports. On the other hand, the rule options section includes alert messages and specific components of the packet that necessitate inspection to determine whether the rule should be triggered. Further details are outlined in the following lists:

Rule Header:

- The *action* is what Snort actually does (alert, log, pass) when it finds a packet that matches the rule criteria.
- The *protocol* references one of the four protocols (TCP, UDP, ICMP, and IP) that are analyzed for suspicious behavior.
- *Source Address* and *Port* give information about the IP address and port of the source.
- The *direction* operator indicates the orientation, or direction, of the traffic that the rule applies to. It defines what is the source and the destination of the traffic.

Rule Option:

- The *msg* option includes the message that is printed along with a packet dump or to an alert.
- The *sid* is a unique identifier for each Snort rule.
- The *priority* assigns a severity level to rules.

Listing 1 includes a Snort rule that is constructed to detect a Denial of Service (DoS) attack. DoS attacks are very common and can be conducted using various tools. This kind of attack is expected to be conducted in the context of the KIOS use case. The *action*, in this case, is *alert* indicating that an alert will be created if the rule is triggered. The *protocol* is defined as TCP. The "`any any -> $HOME_NET any`" part of the rule

defines that the source IP and port of the packets can be any. The same for the destination port. The destination IP can be whatever is included in the $HOME_NET network. Flag S is assigned to SYN (synchronize) packets used for TCP connection establishment. The *msg* is the text that is going to be included in the alert that Snort will create. Finally, the "threshold: type threshold, track by_dst, count 100, seconds 60;" part describes the threshold of the rule. According to this description, if 100 packets with these characteristics are seen from the same destination IP within a 60-second timeframe, an alert will be generated. Note that the time window and number of packets included in this threshold should not be the same for every given system. What is considered normal traffic could be quite different as far as individual systems are concerned.

```
alert tcp any any -> $HOME_NET any (flags: S; msg: "DoS Packet Detect-
ed"; flow: stateless; threshold: type threshold, track by_dst, count
100, seconds 60; sid: 100001;)
```

Listing 1: example Snort rule for DoS packets detection

4.5 UNCERTAINTY MANAGEMENT

The increasing use of ML-based components even in safety and financial-critical applications warrants a demand for dependability assessment of these components. The underlying ML models, although excellent in overall performance, are also bane for the dependability of the system, as they introduce new sources of uncertainties. These can be described under model fit uncertainty (uncertainty due to the model itself), data quality uncertainty (uncertainty resulting due to the quality of the data), and scope compliance uncertainty (uncertainty due to the scope mismatch). Quantifying these uncertainties can be useful in assessing the dependability of the host system [15]. In this work, we focus on scope compliance uncertainty. In previous work, SafeML, a model-agnostic approach to evaluate the Statistical Distance Dissimilarity (SDD) is proposed [16], see also SESAME deliverables D4.1 and D7.1. SafeML estimates the dissimilarity between the training data, and the data observed at runtime. In this section, we discuss an enhancement of SafeML in terms of Scope Compliance Uncertainty Estimate (SCUE).

In the originally proposed method, SafeML provides a binary outcome. This decision uses a predetermined threshold (specified at design-time), to either accept the model's outcome at runtime, (if the SDD is lower or equal to the threshold) or reject it (if the SDD is higher than the threshold). While useful in principle, this binary outcome is limited when considering the range of dynamic scenarios at runtime. By using a continuous uncertainty value, improved adaptability can be achieved, in terms of reacting to more granular changes in SDD. We term this advanced approach as Scope Compliance Uncertainty Assessment (SCUE).

The SCUE uses the calibration set *C*, in which the SDD of the samples correlate to the accuracy of the samples in the corresponding ML model. Under this assumption, the uncertainty of the samples can be defined as the expected inaccuracy over the samples. The measured SDD and observed inaccuracy of the calibration set can be used to obtain the uncertainty from the measured SDD. The calibration set should fulfil the two important requirements.

- Beyond ODD coverage: The calibration set should also include data points that are beyond the Operational Design Domain (ODD); for our purposes, the ODD can be defined as the domain for which the ML model has been intended to be trained for and operating within. Including such samples is important to ensure that the estimate of the SCUE is meaningful. This can be achieved by injecting the Out-of-Distribution (OOD) samples, either by applying artificial corruptions e.g. [17], or by class exclusion i.e., withholding a class from the training dataset.

- Size sufficiency: The shape, size and range of the calibration set should be sufficient to achieve a trustworthy estimate of the SDD. The number of samples also affect the computation time, and hence, it should be considered to keep the computation within acceptable limits at runtime. This can be obtained by the method described in [18].

The SCUE estimator function, which is applied on the fitted data of SDD vs inaccuracy on the calibration set, is also bounded between the values observed at development time. Any value below the minimum SDD observed during development time is regarded as 0, and SDD values above the highest observed during development time are regarded as 1. For all the values in between the two bounds, a measure of SCUE is provided.

$$f(x) = \begin{cases} 1, \ if\ x > Max(sdd) \\ 0, \ if\ x < Min(sdd) \\ SCUE(x), \ otherwise \end{cases}$$

Equation 1 SCUE Estimator Function

4.6 RT EDDI ORCHESTRATION

We rely upon the communication framework that is relevant for inter and intra-robot communication to deploy and orchestrate RT EDDI MASs. Given the prevalence of ROS, ROS nodes and communication via publishing/subscribing to message-queue-topics is a usual option. This is already supported through the code generation results presented in SESAME deliverable D7.2.

Recall that design-time EDDIs (DT EDDIs) can be provided as input to corresponding code generation tools to yield RT EDDIs; these can be deployed directly in ROS nodes and communicate as outlined above. The messages defined for the corresponding ROS nodes can be mapped to EDDI actions and events, according to which RT EDDI agents are communicating.

An example of how this looks for a ROS node running a ConSert RT EDDI can be seen in Figure 6. In the figure, the ROS node subscribes on the left to ROS topics whose EDDI Event message payloads contain ConSert-relevant Runtime Evidence (RtEs) or Demands. On the right, the ROS node subscribes to ROS topics whose EDDI Action message payloads allow Guarantees to be forwarded to other nodes running ConSerts and/or nodes of the host application. Other RT EDDIs (e.g., using SINADRA Bayesian Networks, SafeML, SafeDrones, etc.) can be orchestrated in the same way.

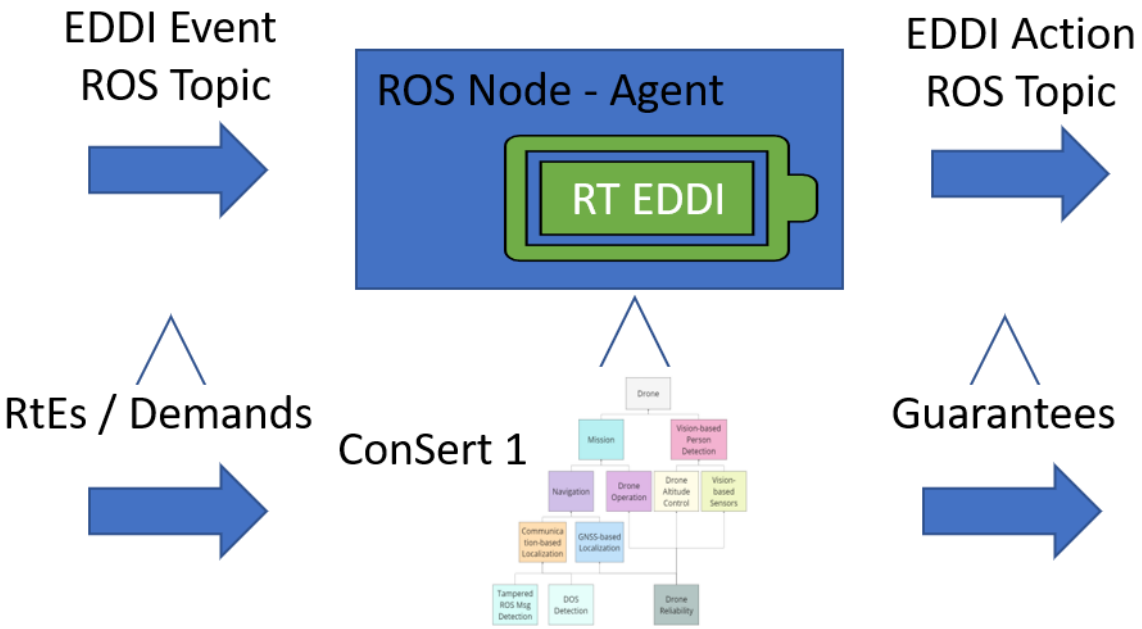


Figure 6 RT EDDI as Agent Deployed on ROS Node

For environments not featuring ROS, we have developed an alternative RT EDDI runner for ConSerts. The runner is developed in Python and uses the ConSerts RT EDDI code generated using the tools described in SESAME deliverable D7.2. For SESAME's KUKA use case, the runner is extended with support for gRPC[1] communication. The user of the runner component can configure it via command-line or configuration file and deploy it as a standalone or Dockerised instance. When the runner is used, each agent of the RT EDDI MAS can be deployed with its own runner instance. An overview of the runner's deployment workflow can be seen in Figure 7.

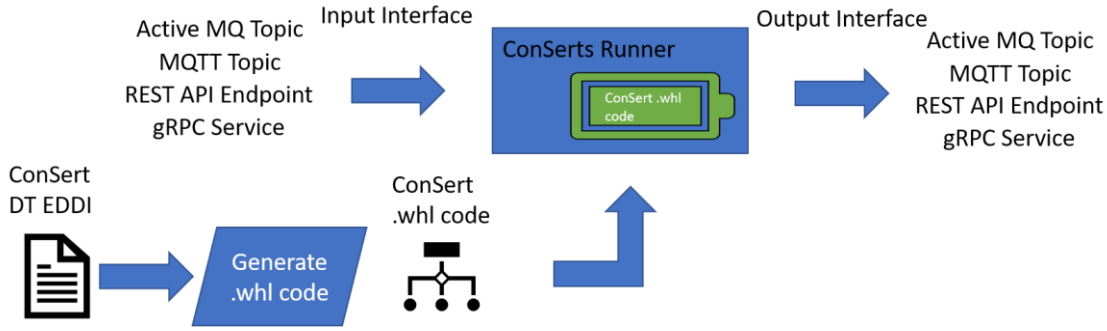


Figure 7 ConSert Runner Deployment Overview

5. USE CASE EXAMPLES

In this section, we present examples of RT EDDIs across the SESAME use cases.

[1] https://grpc.io/docs/what-is-grpc/introduction/

5.1 KIOS USE CASE – CONSERTS, SINADRA, SAFEDRONES, SAFEML, IDS

The KIOS use case has two main modes. In this use case, a fleet of drones can either be used to perform a regular inspection and maintenance. In this operation mode, the drone's camera will mainly be used to identify the abnormalities in the powerplant. In the second operating mode, emergency response and rescue missions, the fleet of drones will be used to observe the powerplant for a potential or imminent collapse, after any catastrophe. The main aim of this mode is to search and rescue any personnel that might be stuck in the powerplant.

An overview of the EDDIs applied for this use case is given in Figure 8. In there, the propagation of the EDDI Events through the EDDIs can be seen, as well as the EDDI Actions used by the adaptation logic. In the following sections, each of the EDDIs for the KIOS use case are described in more details.

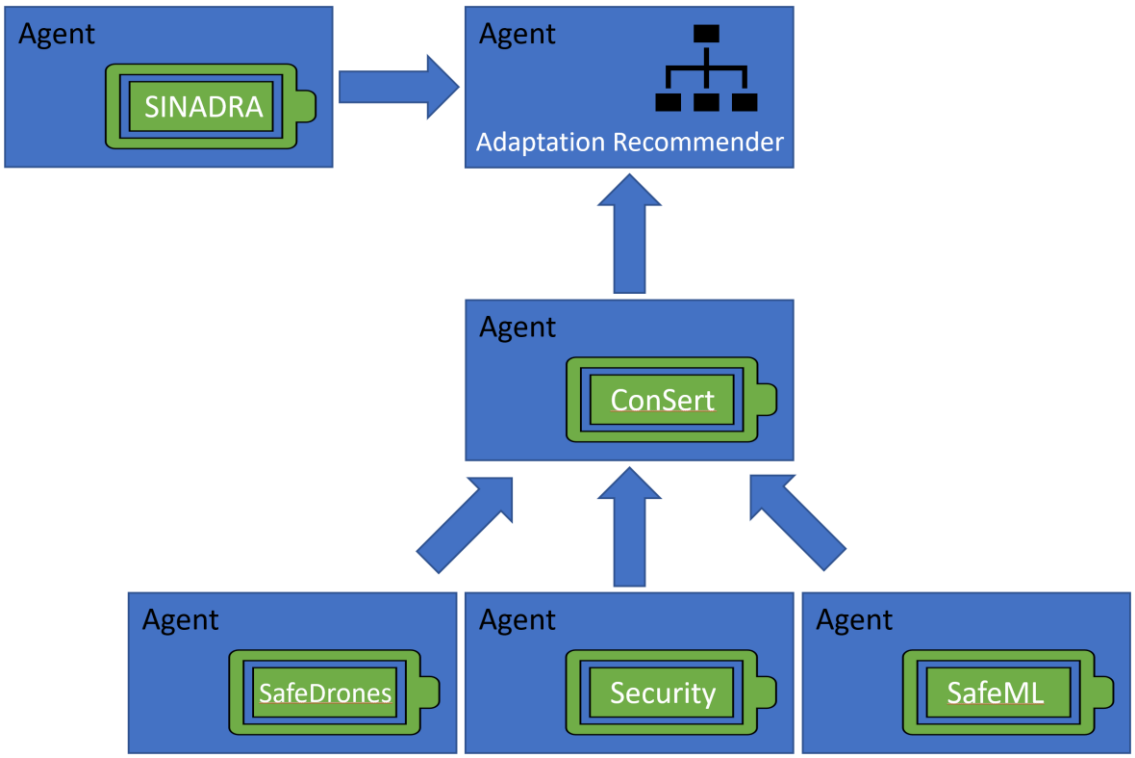


Figure 8 Overview of the connection of the MAS' agents regarding the EDDIs in the KIOS use case

The adaptation recommender is tasked with connecting the dependability assessment provided by the RT EDDI agents (e.g., using ConSert and SINADRA), and make a recommendation to the host robot. Such adaptations can leverage the technology developed in SESAME WP2, including the Perception-Aware MRS trajectory planning and tracking in SESAME deliverable D2.4. The RT EDDI agents can provide relevant dependability metrics for the trajectory algorithms to adjust the MRS inter- and intra-robot distance from other obstacles or drones. For instance, when an individual drone is found to be experiencing partially degraded performance due to sensor degradation, the ConSert-powered RT EDDI could issue an action request to the adaptation component and accompany its request with risk estimation metrics.

5.1.1 ConSerts

The ConSert for the KIOS use case consists of 12 hierarchically ordered ConSerts as shown in Figure 9. The colour of the guarantees of the individual ConSerts correlates to the colour of the ConSerts in the overview.

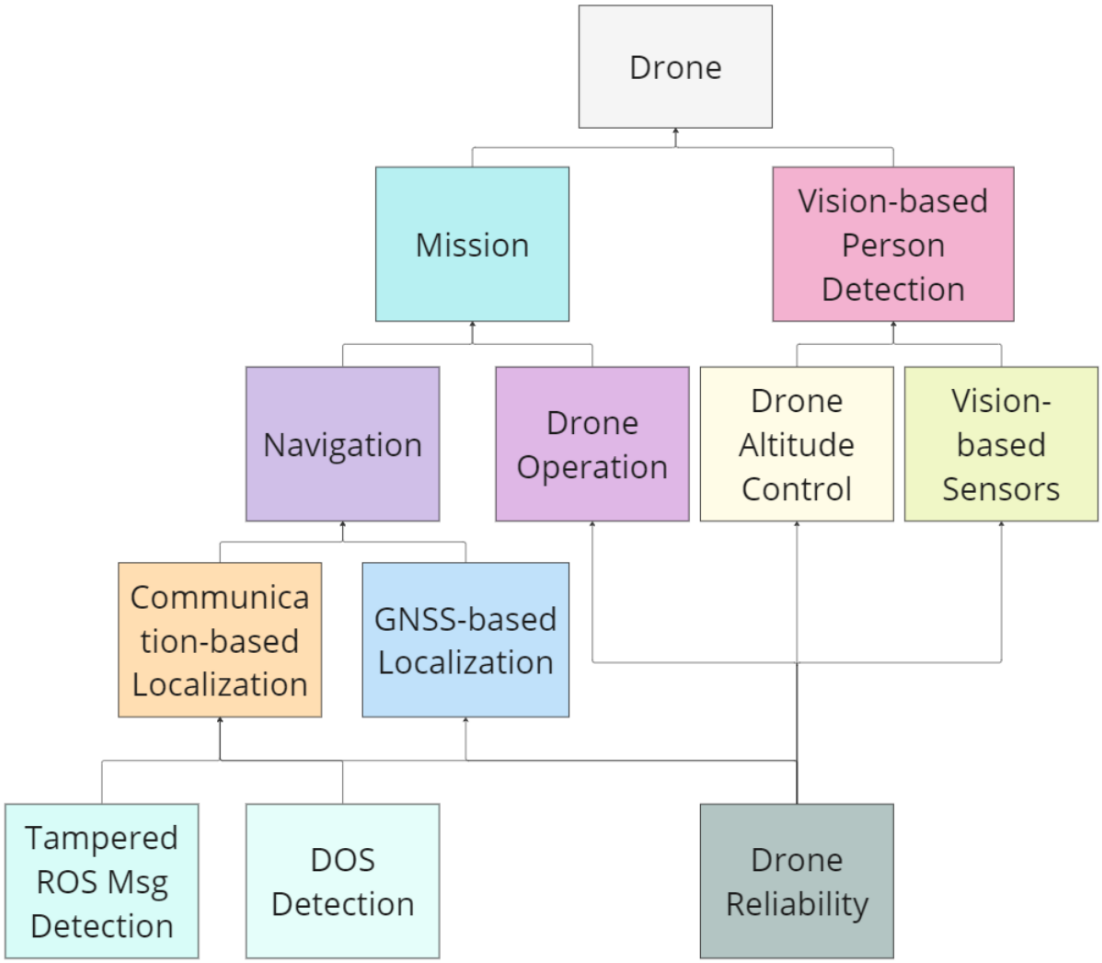


Figure 9 ConSert overview of the hierarchy

The ConSert for the KIOS use case provides on the Drone level a guarantee on how many and how reliably the drone is capable of detecting humans in the target area. Drone can guarantee that it detected a) the correct number of people, b) all people but it counted more people than those actually present in the area, c) it detected most of the people and d) only a minority of people were detected. Based on these guarantees, the base station can then decide to send another drone to increase the number of detected people.

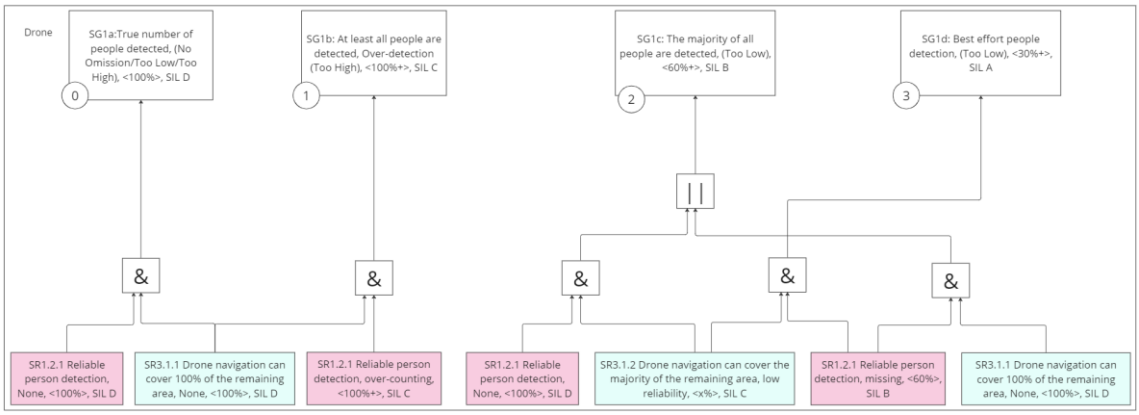


Figure 10 Drone-level ConSert

The drone evaluates these guarantees based on the reliability of its navigation system and its person detection component. Thereby, the best guarantee, i.e., having all people and their exact number correctly counted, can only be provided when the drone is capable of fulfilling its mission by covering the whole area and the person detection system has no failure. With degraded mission guarantees or detection capabilities, the guarantees are degraded accordingly.

The mission ConSert provides guarantees that represent the capabilities of the drone to navigate and fly over the remaining area. The best guarantee is given when the drone can navigate to cover the whole remaining area. Therefore, the drone needs high precision navigation and the capability to reliably operate, i.e., fly over the whole area.

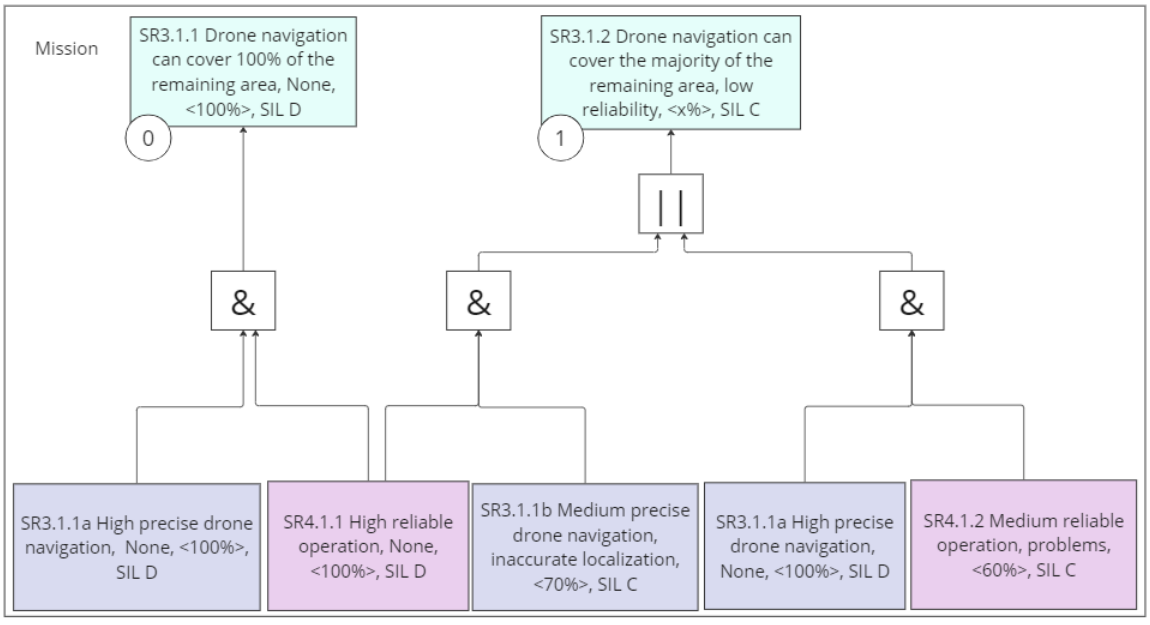


Figure 11 Mission-level ConSert

The navigation component only depends on precise localization and an up-to-date map. Accurate localization can be guaranteed by two different components. Either the (Global Navigation Satellite System) GNSS-based localization or the communication-based localization. The highest level of navigation requires highly accurate localization which is only possible via GNSS localization under good conditions. However, in case

of bad or no GNSS connection, the system can localize itself through communication with other drones.

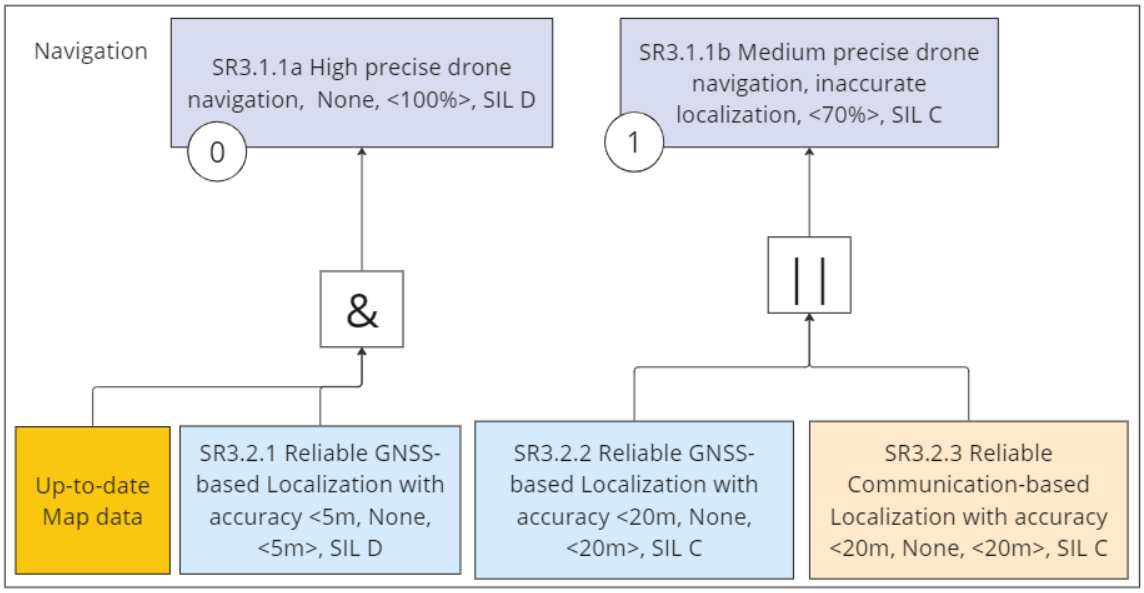


Figure 12 Navigation ConSert

The GNSS-based localization component can provide two guarantees with either less than 5 meters or less than 20 meters accuracy. For the highest accuracy of less than 5 meters, the environmental conditions must be excellent.

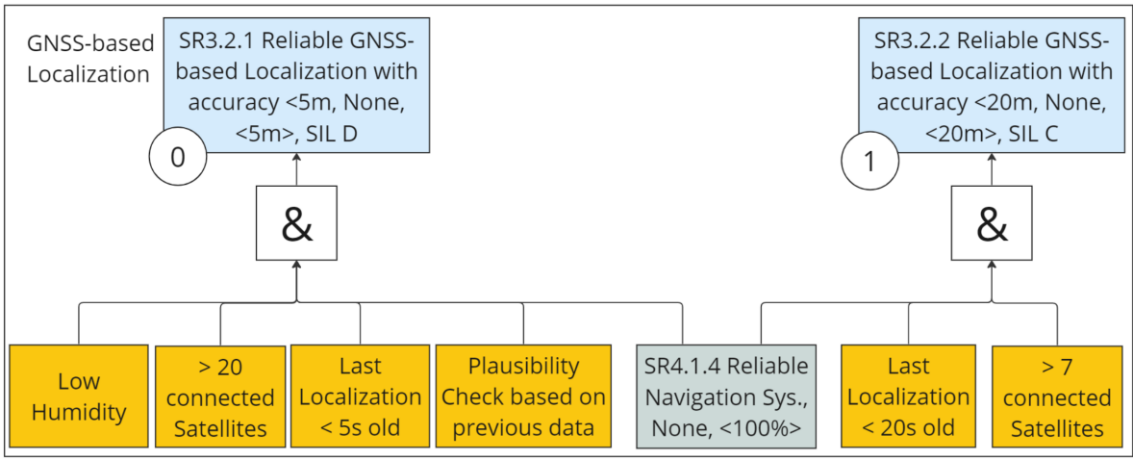


Figure 13 GNSS-based localization ConSert

In case of no GNSS connection or very unreliable GNSS-based localization, the drone can also calculate its position based on communication with other drones. The related ConSert is shown in Figure 15. The drone requires to be connected to at least 3 other drones and the communication hardware must be reliably working. Additionally, the system should not be under attack. Here, components are implemented that detect denial of service (DoS) attacks and tampered messages (see Figure 14) and that provide guarantees as demands for the communication-based localization component.

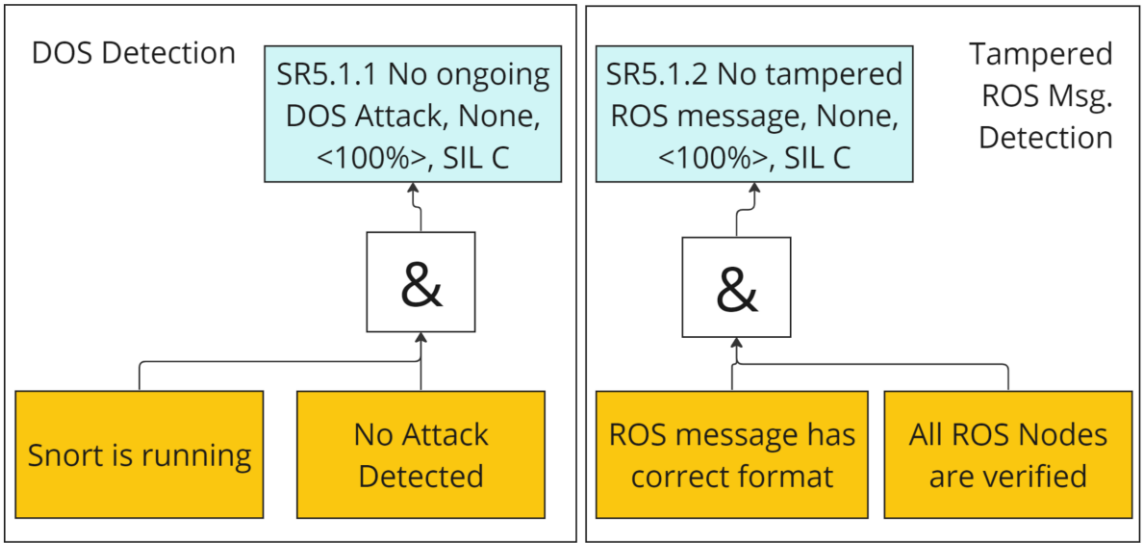


Figure 14 ConSerts of the security components to detect DOS attacks and tampered messages

Another aspect of the mission ConSert is to ensure that the drone can actually fly reliably along the navigation route. Therefore, the internal systems must work reliably. Therefore, a ConSert combines several demands that correspond to system reliability. As shown in Figure 15, reliable operation depends on reliable energy system, propulsion, and obstacle detection. If the battery is degraded significantly so that not enough energy can be provided, a degraded guarantee is given by this ConSert.

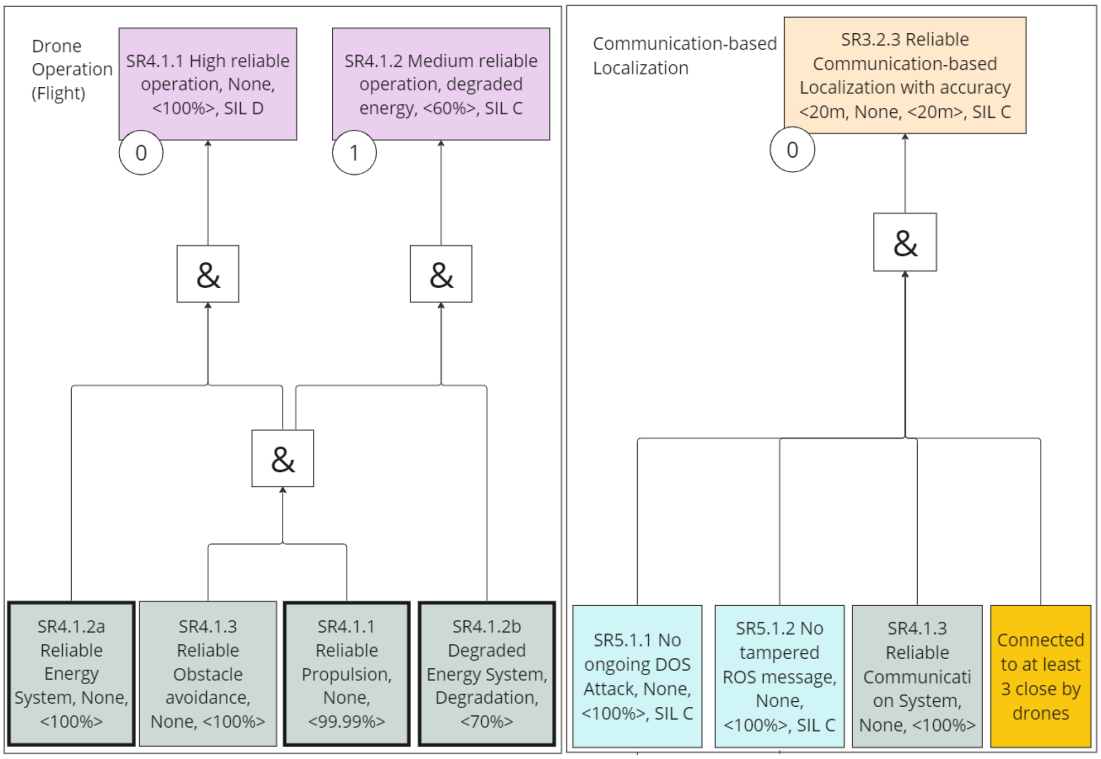


Figure 15 Drone operation and communication-based localization ConSerts

In previous ConSerts, demands are involved that relate to reliable hardware components. These demands are based on the output of SafeDrones and wrapped in different guarantees of the drone reliability ConSert, Figure 16. Here, apart from the

guarantee related to the degraded energy management system, all other guarantees have the same priority and are independent from each other.

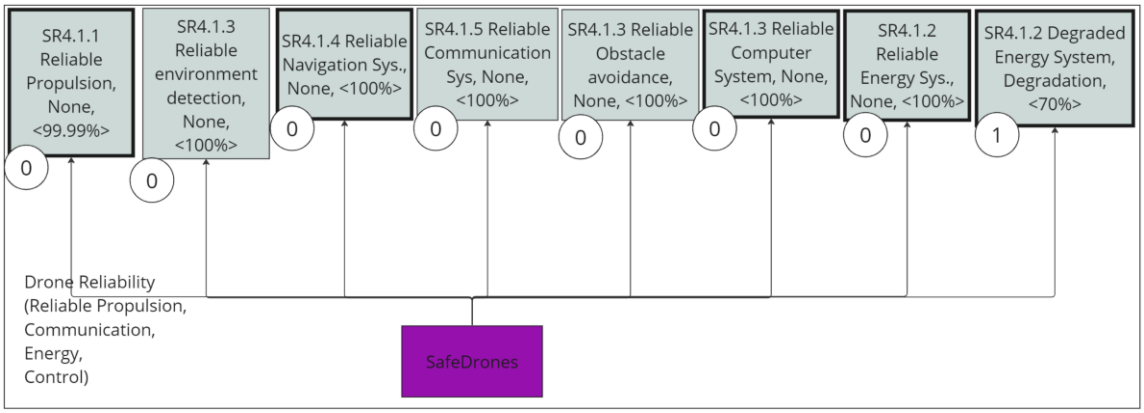


Figure 16 Drone reliability ConSert

On the drone-level ConSert, reliable and precise navigation is necessary, as well as the drone being able to reliably detect endangered persons in the target area. Therefore, the drone-level ConSert receives guarantees for detecting and counting persons. For reliable and accurate detection, a high-quality input image and a proper height are required. Further, the input image is analyzed by SafeML. Only when the statistical distance is small enough can an optimal result be achieved. The concrete ConSert is shown in Figure 17.

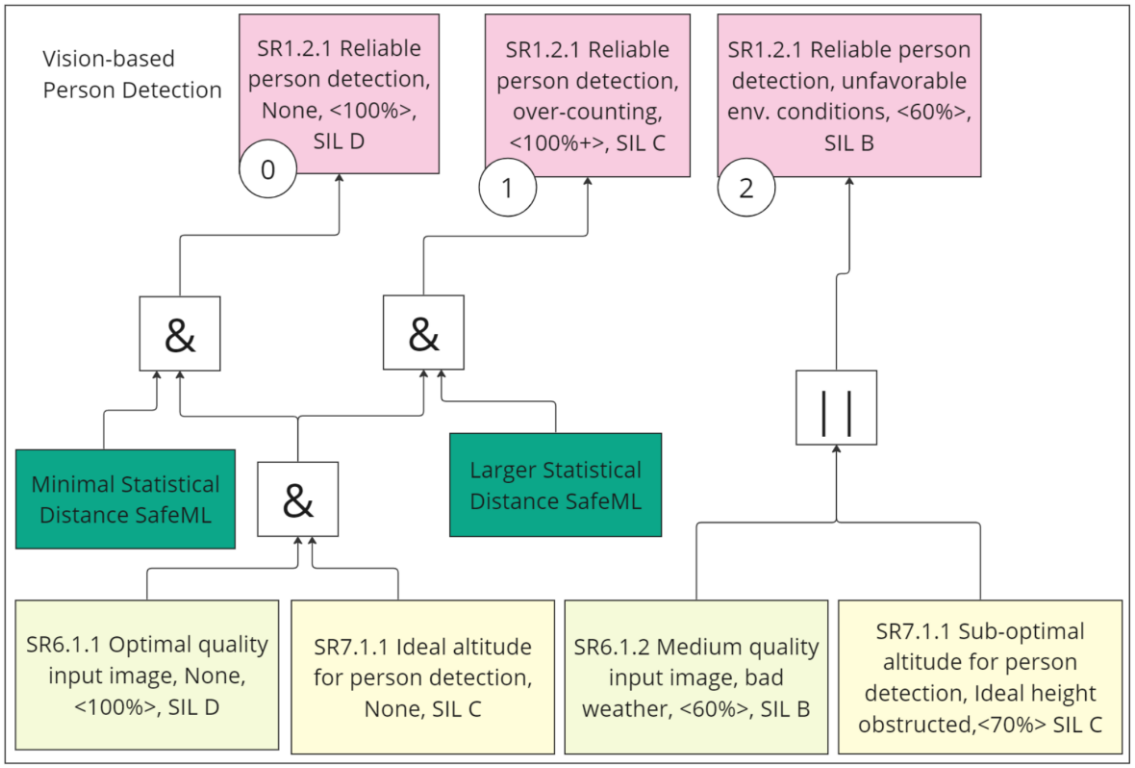


Figure 17 Vision-based person detection ConSert

As shown in Figure 18, the related ConSerts provide guarantees to satisfy the demands by the vision-based person detection ConSert. First, the drone must fly at a specific

altitude to optimize person detection. In case of obstruction, e.g., in a building or under a bridge, the drone cannot fly at an altitude for optimal person detection. It further requires reliable propulsion but also the internal systems of the navigation component must work reliably so that the optimal altitude can be determined and realized.

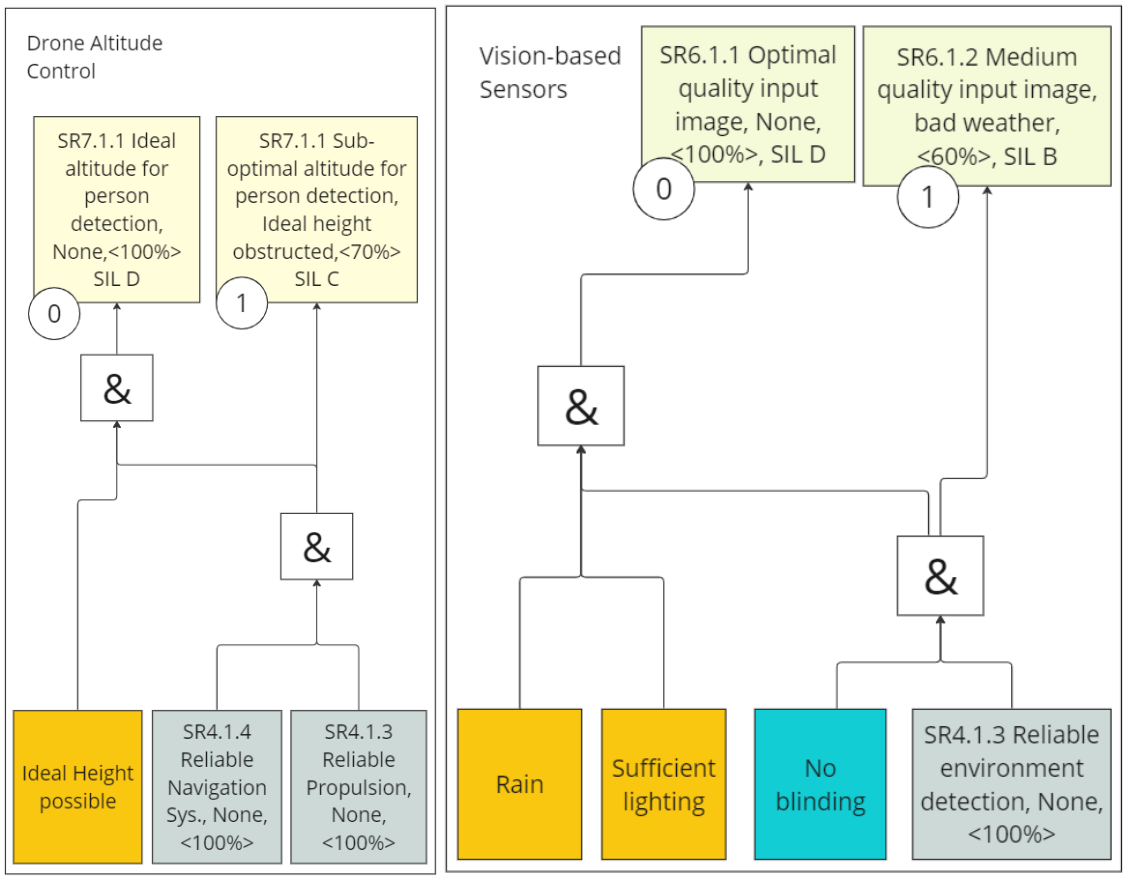


Figure 18 ConSert of the drone altitude and vision-based sensors components

On the other hand, the vision sensors, in this case the camera, must work reliably as well. Therefore, the camera must be capable of perceiving a high-quality image. This is not possible in rain and in bad lighting conditions. Additionally, the camera must not be under attack, e.g., with a laser pointer or other bright lighting sources so that the image is too bright.

5.1.2 SINADRA

For the presented KIOS use case we have modelled two qualitative situation-aware dynamic risk assessment (SINADRA) models representing relevant features and their causal relations towards the risk variability parameters. The features represent the environmental context, as well as the context of the MRS in the specific situation. For inferring the risk variability parameters dynamically during runtime given the environment and the current situation, those qualitative models can be quantified (by expert assessment and/or machine learning) and represented as Bayesian networks. The output of the models is tightly coupled to the ConSert models presented above in Section 5.1.1. This eases the decision taking afterwards to adapt the system accordingly to its capabilities & the situational risk due to the close relationship between the

available capabilities of the system and the risk variability which can be linked to a required capability for the given operational context.

Specifically, this means that the risk variability for different degrees of (a) the person detection and (b) the collision prevention for an individual drone and its assigned area is modelled. In the following, the qualitative SINADRA models are presented, and the inherent concepts and causal relations are briefly elucidated.

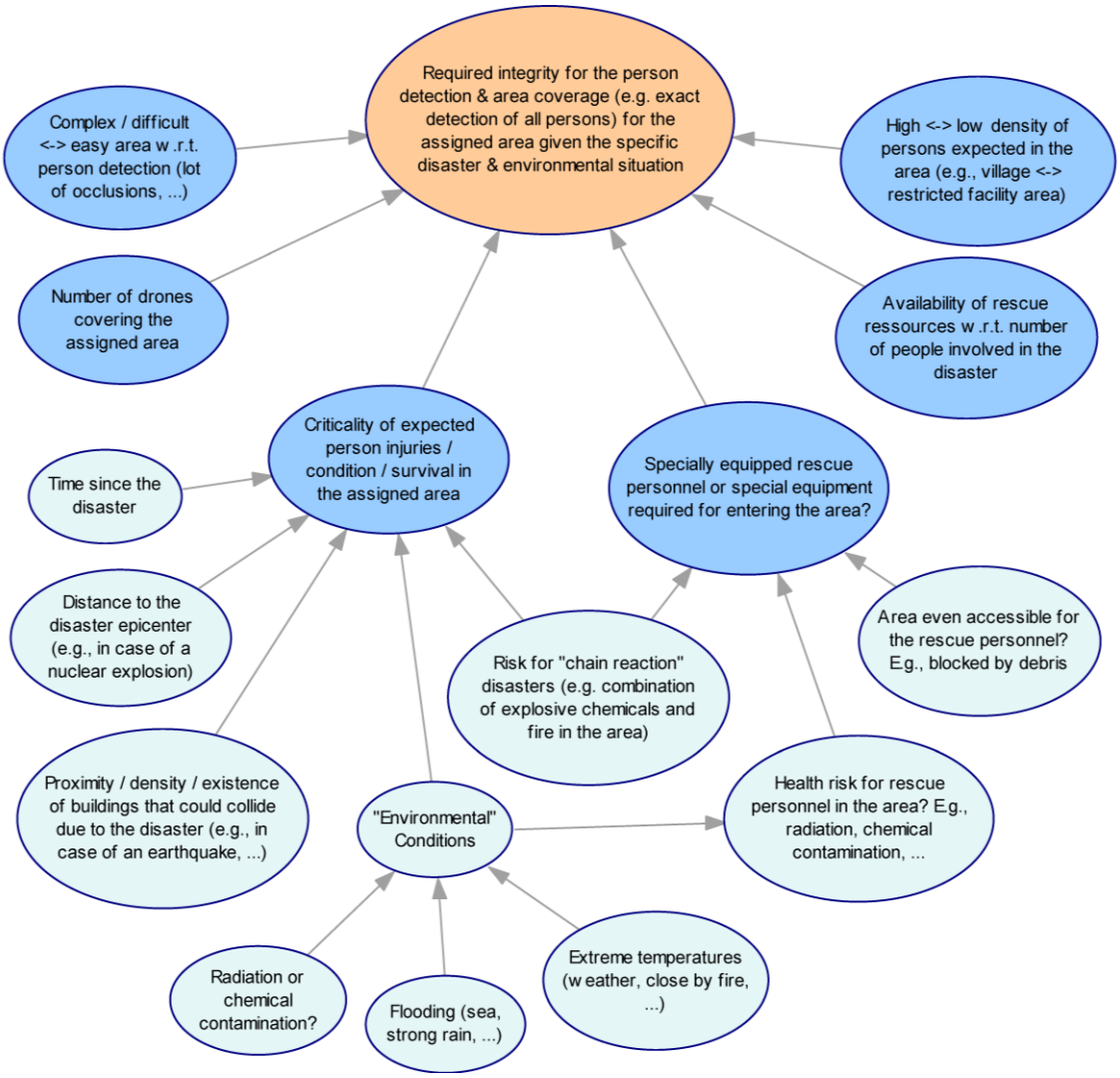


Figure 19 Conceptual SINADRA model (a) for the risk variability w.r.t. the required integrity for the person detection given the situation & environment.

The SINADRA model for the person detection is given in Figure 19. The top node represented in orange defines the risk variability for the person detection, respectively, the required system capability/integrity given the environmental context and the blue nodes represent the main causal influence factors. The main causal concepts that are considered in the model are briefly explained below (starting on the left of the model, going counterclockwise):

- The first concept relates to environmental complexity. For more demanding situations, higher capabilities are required to ensure proper and sufficient person detection. For instance, this variable could be set as invariant for an area

assigned to a drone given the density of buildings, trees, and debris leading to occlusions.

- The number of drones that are redundantly scanning the current area is relevant as well. In case of only one drone scanning an assigned area, a high integrity of the drone's capability must be guaranteed to fulfill the task. In contrast, multiple drones can compensate for faults leading to degraded capability integrity in individual drones still collectively ensuring a high integrity person detection.

- Next, the expected criticality of injuries of the persons to detect in an assigned area is highly relevant. If severe injuries are expected so that the people cannot call for attention or help themselves, a high-integrity person detection is needed to ensure quick help for the injured. A degraded person detection, missing people during the scanning of an area, is only acceptable in case of no severe injuries are expected so that people can provide first aid themselves and are not in immediate danger of life if not rescued by the first opportunity. This criticality depends on several factors like the distance to the epicenter of the disaster or different environmental conditions like the exposure to chemicals or radiation in the assigned area.

- Another aspect is the "cost" of rescue, meaning the need for special equipment for either entering an area safely or for being able to enter it at all. These "costs" can depend on the environmental conditions, e.g., radiation or chemical contamination, which may lead to the requirement of protective gear for ensuring the first-aid responders' safety. Another aspect factoring into the "cost" is the presence of blocked paths to enter the area, for instance, due to debris or floodings. In high "cost" areas it is more difficult to search for and help people, therefore, a high integrity of the person detection is needed to ensure that no endangered person is missed.

- The availability of rescue resources, e.g., equipment or personnel, in relation to the (expected) number of people involved in the disaster is another relevant influence. In case of sparse rescue resources, a higher integrity of the person detection must be ensured to efficiently manage and deploy the resources.

- Finally, the (expected) density of people in an area, e.g., a village location versus a restricted facility, can indicate typical human grouping behavior which helps in finding and rescuing several people at once even if only one person in a group was detected by the drone. Thereby, the potential risk of missing a person decreases.

Figure 20 Conceptual SINADRA model (b) for the risk variability w.r.t. the required integrity for the collision avoidance given the situation & environment.

Analogously to the SINADRA model for the person detection above, Figure 20 shows the SINADRA model for collision avoidance. Again, the top node represented in orange defines the risk variability, respectively, the required system capability/integrity given the environmental context and the blue nodes represent the main causal influence factors. The main causal concepts that are considered in the model are briefly explained below (starting on the left of the model, going counterclockwise):

- Starting with the complexity environment, it makes a difference for the required collision avoidance whether the assigned area is, for instance, an open field with no obstacles or a city landscape with lots of buildings requiring precise motion, navigation, and obstacle detection.

- There is always the risk of injuring people when colliding with an obstacle and consequently crashing into the ground or when colliding directly with a person. This risk can be quantified by, e.g., the expected drop/collision speed and the

weight and form factor of the drone itself. Further, a person must be in the vicinity of the drone, whereby the vicinity depends on the height, the wind strength, and the drone's velocity.

- Next, the collision criticality with other objects is assessed. For this the "strength" of the possible collision is considered, e.g., given by the mass ratio and velocity delta (approximating the conservation of momentum). Further, the existence of possible collision partners, i.e., static objects or other drones, in the drone's vicinity is checked, e.g., by examining the global path planning.

- Another relevant aspect is the ground collision criticality, specifically dropping to the ground after an aerial collision. Here, the focus lies on the expected damage to the drone, e.g., given by the hovering height, and the "follow-up" damage or worsening of the overall disaster situation, possibly endangering humans, e.g., given by the risk of starting a fire due to chemicals or dried out grass on the ground.

- Finally, the importance of the specific drone is assessed. This includes the value for the overall mission success and the drone's monetary value. Former can be defined, e.g., by the number of already lost drones during a mission or the importance of the currently assigned task, respectively, the assigned task queue. The latter can be defined, e.g., by the price to replace that drone including aspects like specialized or adapted drones which cannot easily or quickly be replaced.

Another important aspect is how the input features are provided to the SINADRA model and checking which ones are dynamically assessed during runtime and which ones are invariant for the situation. Due to the broad variety of features, there is no correct way applicable to all the features but rather this depends on the nature of the feature itself. Some of the features can be provided at runtime via sensor values, e.g., the radiation contamination at the current location, or via communication with the centralized operation station, e.g., the number of remaining healthy drones or the vicinity of other drones, or via comparison of the current location with a 3D map, e.g., for assessing difficulty w.r.t. occlusions at the current location. Others are invariant for the mission and can be defined by trained operators at the start of the rescue mission, e.g., the expected density of people in the respective areas.

Uncertainties are another key factor for the provision of input features. The SINADRA model will be probabilistically inferred at runtime, enabling propagation of uncertainties. To increase the certainty of an observation, other drones in the overall MRS that are close by can provide their own observations, e.g., measurements of radiation contamination, to increase confidence. The fusion of (un)certainty values for inputs is no easy objective and potential dependent faults must be considered but the flexibility for extending the dynamic risk assessment in the future is given.

After assessing the risk variability dynamically, the inferred output can be used in an adaptation component to take an informed and safe decision. Other EDDI components of the MAS, as well as the overall MRS and the collection of the individual robot's EDDI outputs, can be considered in this decision process. For instance, in case of a degraded drone system, it can be decided that the drone can still proceed with its

assigned task due to a "less-than-worst case" situation, or that a second drone in the MRS must support accomplishing the assigned task. Depending on the situation, it could also be decided that a degraded, thus, partial, task completion is sufficient and that more rescue resources are allocated for the areas instead to compensate.

The next steps for the application of SINADRA in the KIOS use case are to finalize these qualitative models and quantify them, represented by the EDDI (Bayesian network), which can be used for inference at runtime. Also, the SINADRA models provided will be further tailored to the use case specifics in this project to enable beneficial application and efficient deployment to the drone MRS in the end.

5.1.3 SafeDrones

SafeDrones is a reliability modelling approach designed to enable runtime reliability and risk assessment of Drones [19]. It is based on EDDI concepts. SafeDrone uses fault trees as the overall model with Complex Basic Events (CBEs) to support reliability evaluation dynamically. For a generic drone, a generalised fault tree with 9 main failure categories and 28 sub-categories of failure is proposed. This can be seen in Figure 21. This is further detailed in Figure 22.

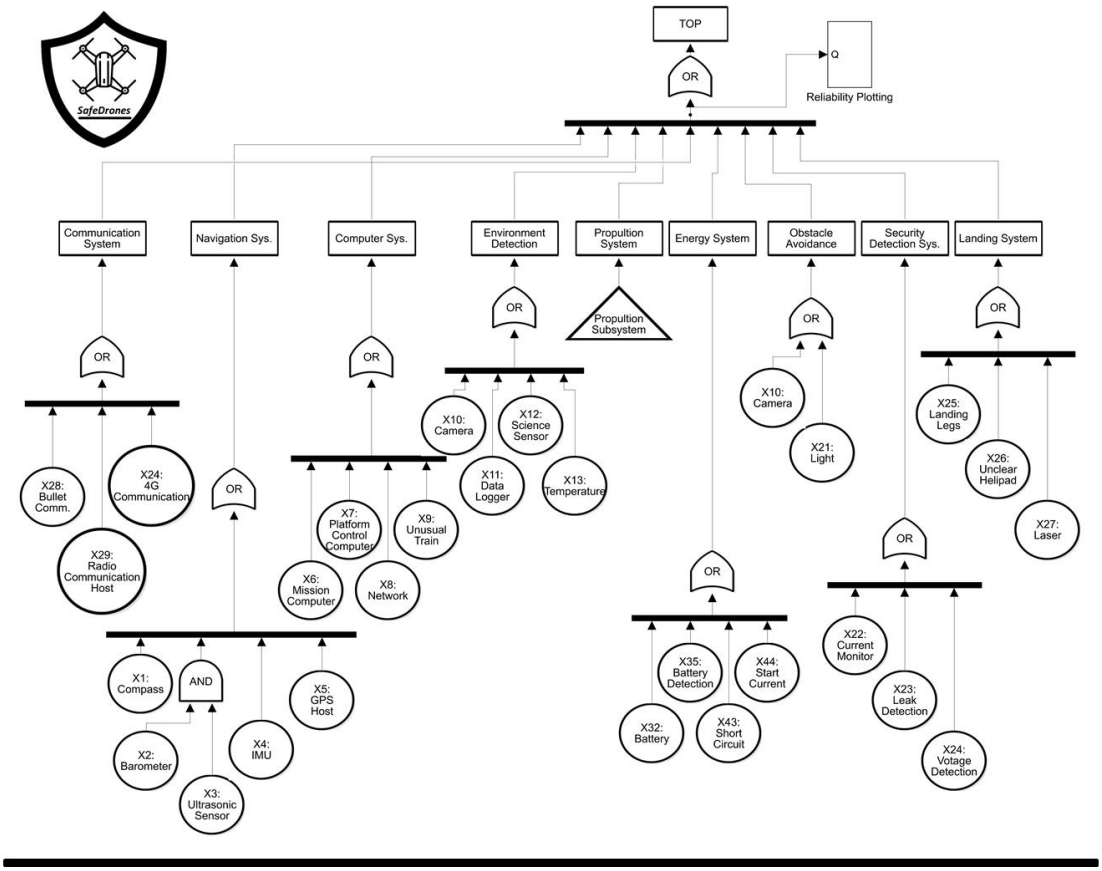


Figure 21 Proposed fault tree of a generic Drone

The consideration of CBEs expands the FTA to consider not only the Semi-Markov-Processes (SMPs) but also other functions (e.g., Arrhenius Equation[2]). It also requires a

[2] https://en.wikipedia.org/wiki/Arrhenius_equation

symptom event for each CBE. By using the pre-defined models, the model handles the reliability evaluation of different configurations. The reliability assessment function is executable and can be independently executed at individual drones to find their remaining reliability at runtime.

In this work, three main failures are considered as the CBEs. They are the Battery failure, which uses the battery model with four degradation levels to obtain the probability of the failure of the battery. Processor failure, which can be updated using the actual temperature, and its probability of failure can be obtained based on it. Finally, the Propulsion failure uses the configurations of drones possible (either on quadcopter or on hexacopter), and evaluates the probability of failures, based on the probabilities of failures of the rotors. It is also possible to consider the probability of communication failure due to runtime security attacks, see sections 4.4 and 4.5. Thus, achieving a Safety-Security con-engineered EDDI solution for the drone use case of SESAME.

From these probabilities of failure, reliability can be obtained. These can be used to extend the FTA with the "Symptoms layer", which can further be the symptoms for basic events. These symptoms should be observable and should be identified by the domain expert during the design phase. This can be useful in anticipation of imminent failures and proactive prevention of accidents by using appropriate responses to reduce the risk of accident due to the symptoms observed.

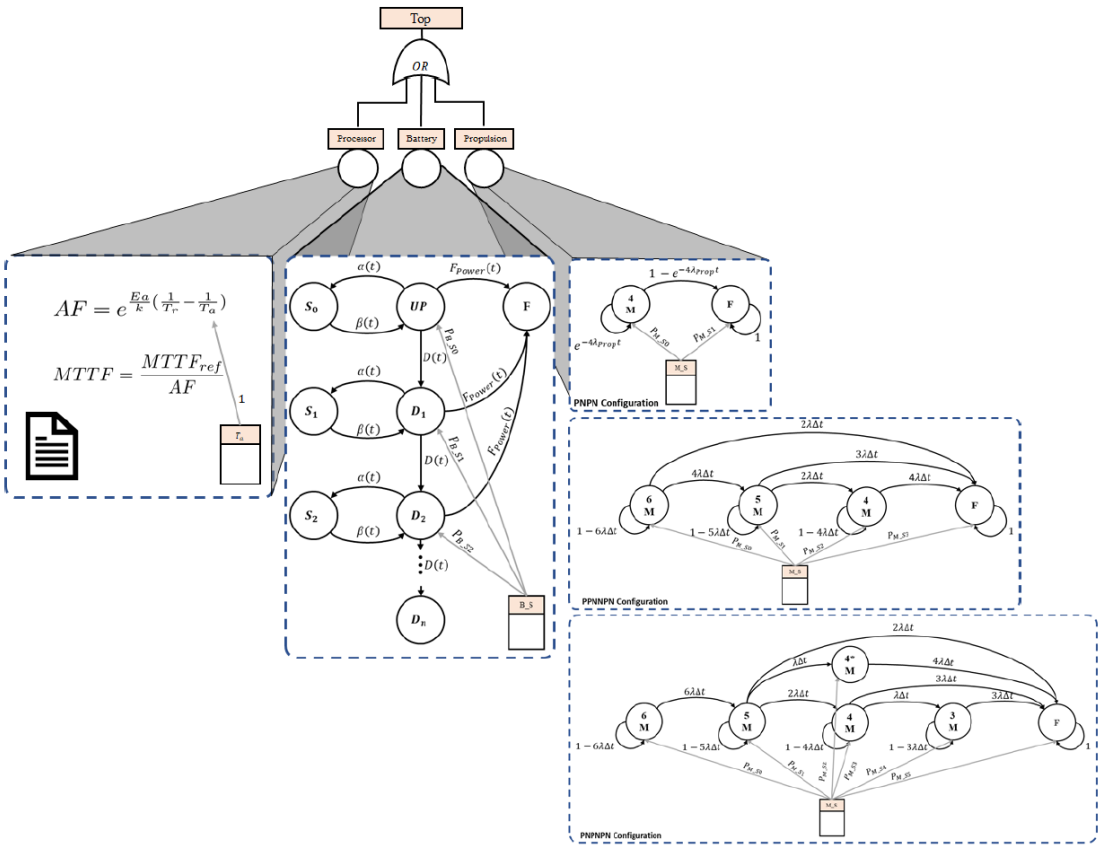
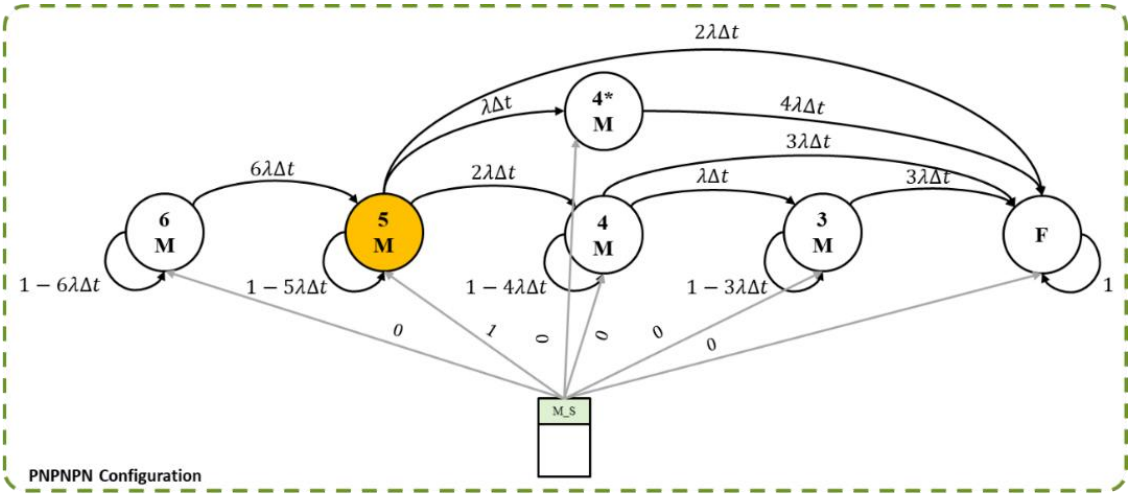


Figure 22 Simplified FTA of the drones with three different types of propulsion configurations

5.1.3.1 Extensions to SafeDrones from D7.1

In the previous version of SafeDrones featured in D7.1, the uncertainty of symptoms was not considered (Figure 23) while in the updated version (Figure 24), the uncertainty of the symptoms has been considered by making a probabilistic link between symptoms and states in complex basic events.

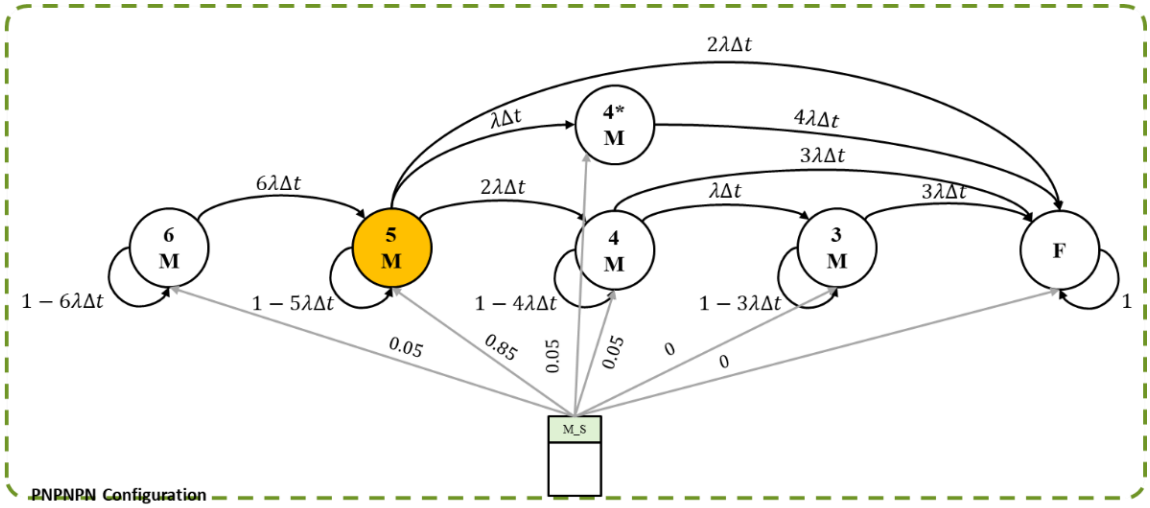



```
P_Fail, MTTF = Motor_Failure_Risk_Calc([0,1,1,1,1,1], 'PNPNPN', 0.001, 100)

print(P_Fail)

print(MTTF)
```

Figure 23 Markov model of a hexacopter with PNPNPN configuration and motor status (M_S) as a symptom – binary link between the symptom and the system's states.




```
P_Fail, MTTF = Motor_Failure_Risk_Calc([0,1,1,1,1,1], 'PNPNPN', 0.001, 100)

print(P_Fail)

print(MTTF)
```

Figure 24 Markov model of a hexacopter with PNPNPN configuration and motor status (M_S) as a symptom – considering the uncertainty of symptoms.

The second update for SafeDrones was developing more complex basic events. In Figure 25 is a numerical result created by a 12-state Markov model as a complex basic event of a GPS system.

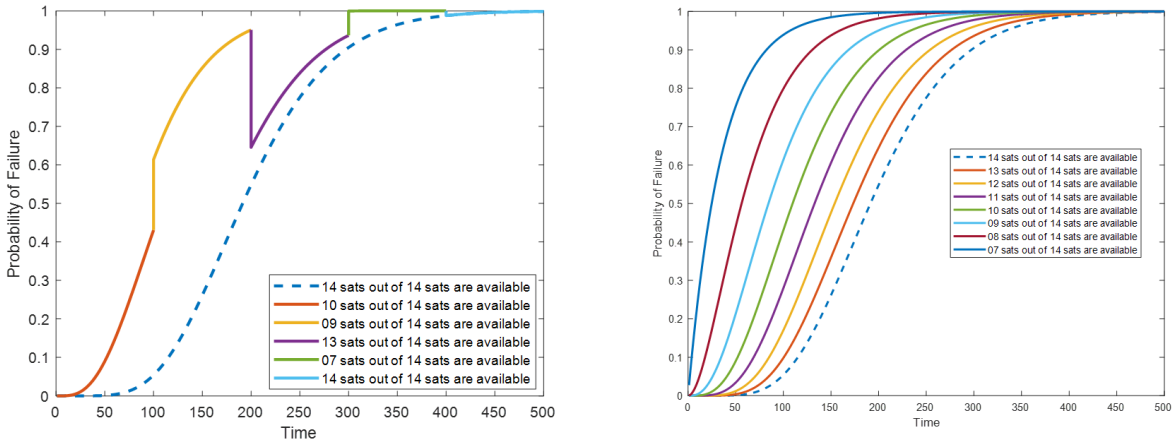


Figure 25 Updating probability of failure in GPS based on number of available satellites.

The third update to SafeDrones introduced two vital functions for multi-UAV systems. A) 'Collision Probability Assessment', analyses the positions and trajectories of multiple UAVs to calculate the likelihood of collisions. By monitoring UAV movements in real-time, it enhances safety through proactive alerts and evasive actions. B) 'Communication Failure Impact Assessment', evaluates the integrity of communication links between UAVs. It gauges the consequences of communication failures, such as loss of coordination, and initiates corrective measures like switching to alternative

communication channels. Together, these functions extend SafeDrones capabilities towards supporting multiple UAVs.

5.1.3.2 SafeDrones and ConSerts

In this subsection, the integration between SafeDrones and ConSerts is described. Recall that previously in Figure 16, the ConSert that wraps the output of SafeDrones into ConSert-Guarantees is shown. Technically, this ConSert can be realized in two different ways:

a) An independent ROS node publishes the results of SafeDrones. This ROS message contains values for the reliability of the subsystem, like the propulsion system, energy management system, and so on. This module can either publish one ROS message with fields for each subsystem's reliability or it publishes one ROS message for each of the subsystems. Custom messages must be added to the catkin workspace so that the ConSert monitor can access them. Instead of custom messages, standard messages but also *ConSertOutput* messages can be used. The ROS messages can then be linked to demands in the ConSert config file. Therefore, the values from SafeDrones are specified in the *SimulatorOutputs* section. Afterward, in the *EDDIInput* section, the previously specified values can be linked via the *required* field. In case further processing is required, a function or a field can be specified (see documentation in D7.2). This connection must be done for each ConSert featuring demands which are fulfilled by SafeDrones. For the KIOS use-case, all ConSerts that are connected to the *Drone Reliability* ConSert, as shown in Figure 16, must specify this connection to the SafeDrone module as described above. In this approach, the *Drone Reliability* ConSert would be effectively subsumed by the SafeDrone module.

b) Another approach is to construct a ConSert that wraps all the reliability values from SafeDrone into guarantees. This ConSert would treat each reliability value as runtime evidence that is connected to a guarantee. SafeDrone values are connected to the runtime evidence as described above in approach a) via the consert configuration YAML. The advantage of this approach is that there is only one place where the ConSerts have a direct interface to the SafeDrone module. Such a configuration file is depicted in Figure 26.

```
---
Model:
    id: SafeDrone
    type: ConSert
    frequency: 10
    threshold: 10
    parameters:
      - guarantees:
        - id: ReliablePropulsion
        - id: ReliableEnergy
        - id: DegradedEnergy
        - id: ReliableNavigation
        - id: ReliableCommunication
        - id: ReliableObstacleAvoidance
        - id: ReliableEnvironmentPerception
        - id: ReliableComputerSystems
SimulatorOutputs:
  - id: reliablePropulsion
    type: std_msgs.Bool
    topic: /rte/reliablePropulsion
  - id: reliableEnergy
    type: std_msgs.Bool
    topic: /rte/reliableEnergy
  - id: degradedEnergy ···
  - id: reliableNavigation ···
  - id: reliableCommunication ···
  - id: reliableObstacleAvoidance ···
  - id: reliableEnvironmentPerception ···
  - id: reliableComputerSystems ···
EDDIInputs:
  - id: RtE_reliablePropulsion
    type: bool
    requires: [reliablePropulsion]
    default: false
    field:
      name: data
  - id: RtE_reliableEnergy ···
  - id: RtE_degradedEnergy ···
  - id: RtE_reliableNavigation ···
  - id: RtE_reliableCommunication ···
  - id: RtE_reliableObstacleAvoidance ···
  - id: RtE_reliableEnvironmentPerception ···
  - id: RtE_reliableComputerSystems ···
EDDIOutputs:
  - id: ReliablePropulsion
    topic: /safedrones/reliablePropulsion
    type: msg.ConSertOutput
  - id: ReliableEnergy
    topic: /safedrones/reliableEnergy
    type: msg.ConSertOutput
  - id: DegradedEnergy ···
  - id: ReliableNavigation ···
  - id: ReliableCommunication ···
  - id: ReliableObstacleAvoidance ···
  - id: ReliableEnvironmentPerception ···
  - id: ReliableComputerSystems ···
```

Figure 26 Example of a SafeDrone-ConSert configuration

5.1.4 SafeML

The ML component in the KIOS use case is the person detector from the drones. Since this is planned to be used in extreme conditions under emergency, the assessment of its dependability is crucial. It is planned to use the SafeML in this regard to assess the scope compliance uncertainty. SafeML can be used to monitor at runtime the data observed and compare its statistical distance dissimilarity (SDD) with design-time data. A demonstration of SafeML is built on the Microsoft Common Object in Context (COCO) dataset [19]. The dataset consists of a set of images of common objects along with their annotations. We filtered the dataset and obtained the images consisting of "person" and ignored the other classes. A YoloV3 pre-trained model is used as a detector [20]. For the Out-Of-Distribution (OOD) data, corruptions are applied on a subset of the COCO dataset filtered for person. As shown in the Figure 27.

(a) (b) (c) (d)

Figure 27 A sample image from the COCO data (a) original (b) with defocus blur (c) contrast (d) with zoom blur

The SDD is then measured between the validation set (without corruption) and the training set, and the validation set (with corruption, considered here as OOD dataset) and the training set. The hypothesis is that the SDD between OOD data and the training set is significantly higher than the SDD between the OOD data and the training set. This can be seen from the bar plot in Figure 28.

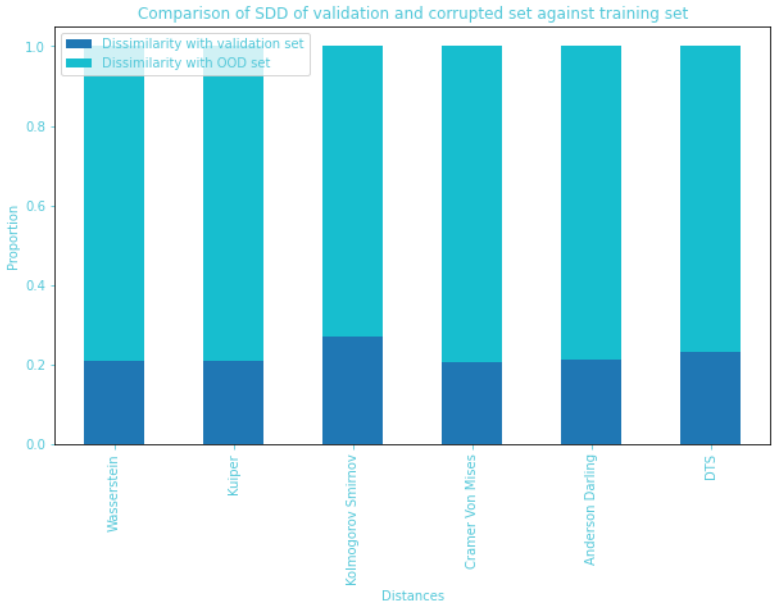


Figure 28 Comparison of SDD of validation and OOD set with the training set

Currently, a dataset prepared by KIOS for vehicle detection using drone point-of-view (pov) is available [21]. However, for person detection, the dataset is still under construction and will be made available, once ready. Meanwhile, KIOS has advised us to use the publicly available Heridal [22] and VisDrone [23] datasets. The Heridal database contains over 68,750 image patches from the drone pov of wilderness, with 29,050 samples containing persons and 39,700 samples without. The VisDrone dataset is collected by the AISKYEYE team at Lab of Machine Learning and Data Mining, Tianjin University, China. It consists of 288 video clips formed by 261,908 frames and 10,209 static images. It is also annotated by bounding boxes of targets such as pedestrians, cars, bicycles, etc. The detector model used by the KIOS is TinyYolov4. In the next step, it is planned to demonstrate the SafeML on drone pov data, using the TinyYolov4 as the detector.

5.2 KUKA USE CASE - CONSERTS

In this use case, KUKA produces robotic assemblies for their customers, which involves composing KUKA robots, conveyor belts, and other components, in an enclosure (protecting human operators by physical separation). This enclosure operates as a part of a production step in an overall manufacturing plant. KUKA aims to (a) evaluate their assembly with respect to safety requirements and (b) assure safety requirements are monitored and preserved during operation. This can be challenging, as the process of transitioning from simulation testing of the planned assemblies into physical testing is not fully automated, and manual effort is required. Therefore, consistent verification of dependability-related requirements across both simulation and real environments is relevant.

In the scenario discussed here, a KUKA KR22 robot is attempting to plug a driveshaft mechanical component into a driveline unit. The driveline can then push the driveshaft into an engine (the end-product of KUKA's customer) for the latter to be tested. The set of RT EDDIs depicted below intend to monitor dependability of the plug-in motion with respect to assuring that the motion only proceeds when all involved components are in

the correct positions and associated signals have been exchanged as expected. The RT EDDIs in subject are all ConSerts, each corresponding to associated guarantees issued by each participating device and robot.

The top-level ConSert to be discussed corresponds to robot KR22, seen in Figure 29. It provides two guarantees for "high" and "medium" reliability of the plug-in motion, and one default where, effectively, the motion cannot be guaranteed to proceed as planned. These guarantees are depicted at the top of the figure in green and red colours.

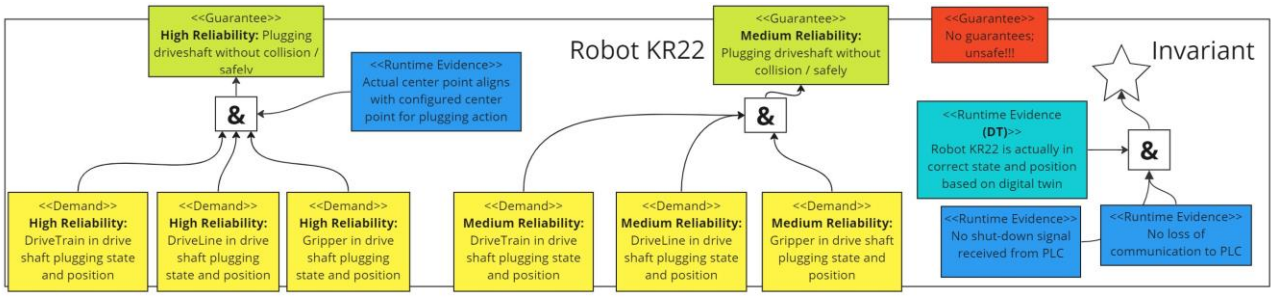


Figure 29 KUKA KR22 Robot ConSert

For "high reliability" to be guaranteed, the ConSert requires runtime evidence (RtE, blue and turquoise colours), indicating that the robot has correctly aligned with the driveline's position, and that the other involved components are also guaranteeing correct states and positions (demands, yellow colour). If the robot cannot itself confirm that it is aligned with the driveline for plug-in, or the other components are reporting lower reliability with respect to state and/or positions, then the robot also reports reduced reliability for the motion. Finally, in both previous guarantees, it is assumed that an invariant should always be true. This is depicted with the 'star' symbol on the right side of the ConSert. The invariant evaluates whether the digital-twin-based simulation is reporting that the KR22 robot is estimated to be in the correct position, that communication to the PLC is still valid, and that no shut-down signal has been issued. If the invariant becomes invalid or neither of the high or medium reliability guarantees can be provided, then the ConSert issues a 'default' guarantee (in red colour). In this case, this guarantee means that the motion cannot be assured. A medium or default guarantee should be communicated to the simulation user interface or plant control, so that appropriate responses are taken.

In Figure 30, the ConSert of the Drive Train component is depicted; it differs from KR22's ConSert in terms of the RtEs relevant for this component, but the logic is otherwise similar. The RtEs specific to each guarantee check whether the latest set of sensor and actor signals issued by the PLC are correct for the component to proceed with the plug-in motion.

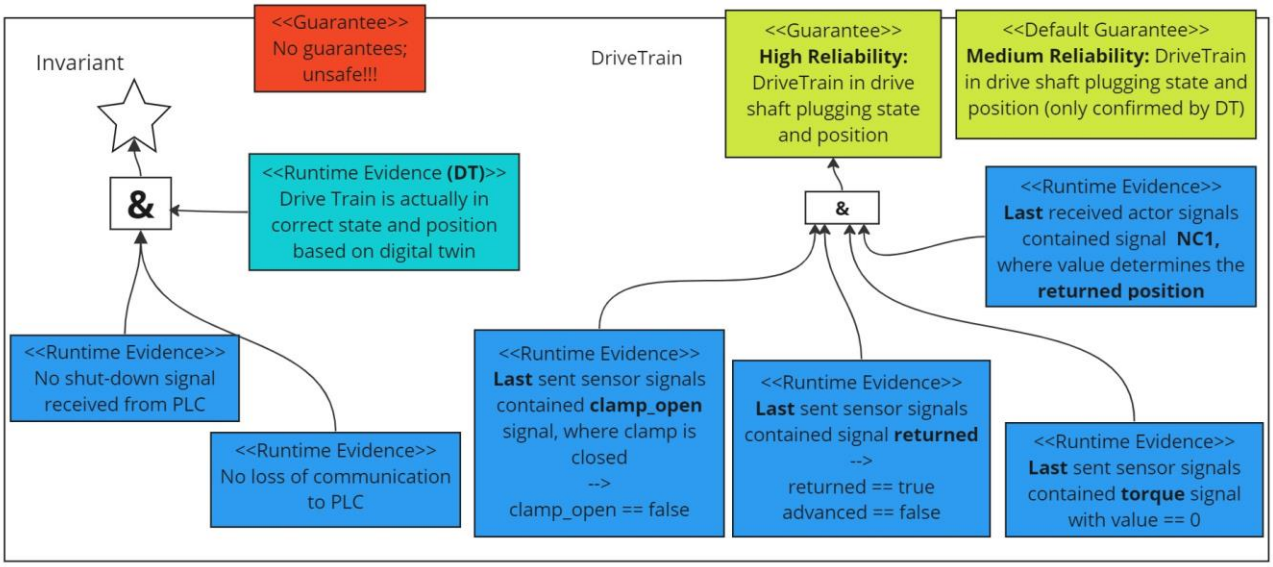


Figure 30 KUKA Drive Train ConSert

For reference, the ConSerts for the two other components (Drive Line and KR22's gripper component) are also listed in Figure 31 and Figure 32, and are evaluating appropriate sensor and actor signals from the PLC. Note that all ConSerts are sharing the same invariant condition checks.

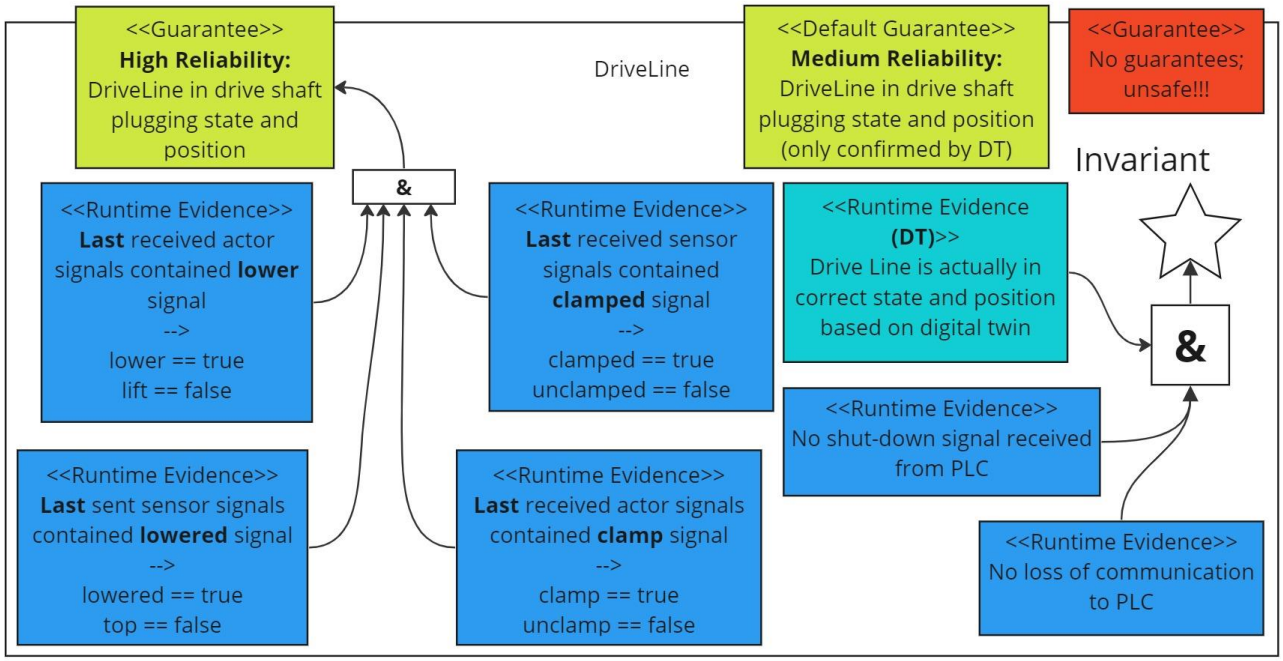


Figure 31 KUKA Drive Line ConSert

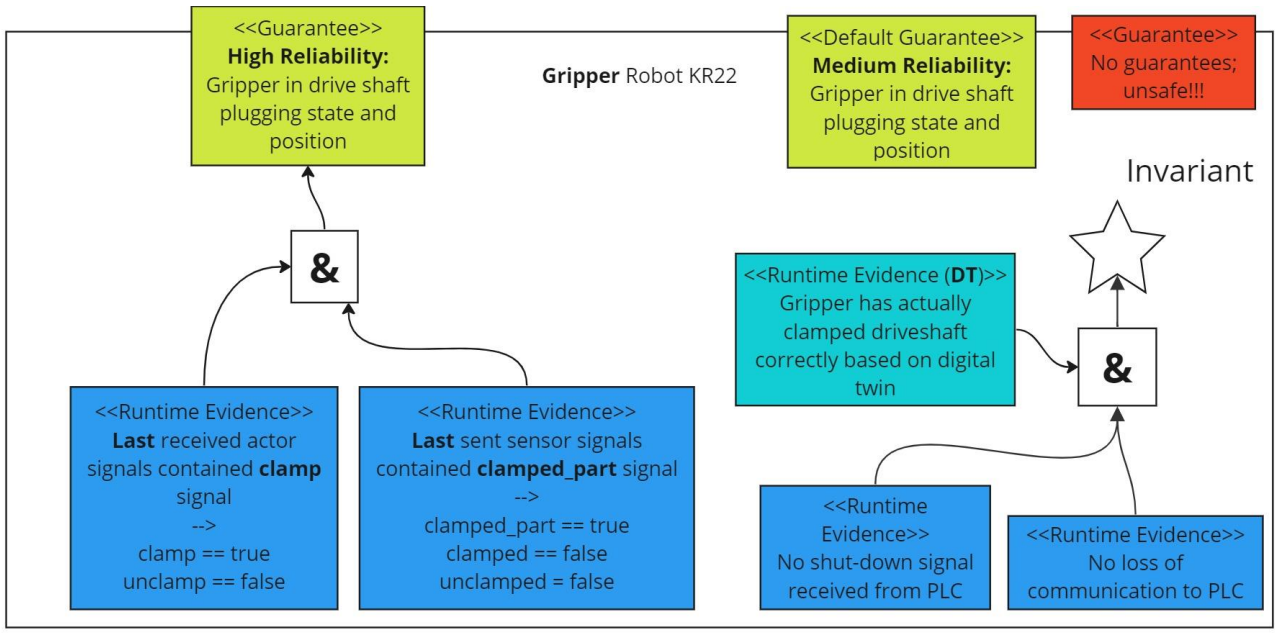


Figure 32 KUKA Robot KR22 Gripper

To deploy these ConSerts as MAS, the runner component, discussed in section 4.6, is appropriate, as the technologies involved in this use case do not use ROS to communicate. The ConSert runner can connect to the rest of the technologies using the topology seen in Figure 33. The EDDI (ConSert) Runner is depicted at the bottom of the figure, communicating via gRPC with the 'Shared Memory I/O'. While the figure is showing only one 'EDDI Runner' element for brevity, each ConSert would be executed using its own runner instance. The shared memory is used as an intermediary between the digital-twin-based simulator, testing framework, and KUKA's own simulation/robots and PLC (see SESAME deliverable D6.6 for additional details).

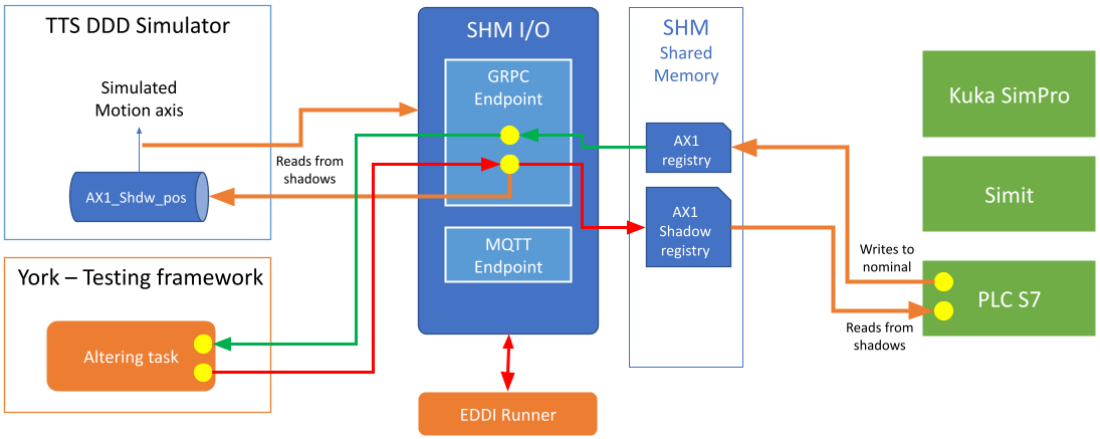


Figure 33 KUKA Use Case Technology Overview

5.3 PAL USE CASE – CONSERTS, SAFEML

We should note that the use case is a late addition to the SESAME project, and therefore further development is expected. It is possible that some elements are further adapted over the remainder of the project.

In this use case, two PAL Base robots are expected to pick up products from designated positions and deliver them to human end-users, with the assistance of a PAL Gripper robot, see Figure 34. The delivery robots need to navigate around static obstacles, e.g., pillars, arrive at their loading positions, be loaded with their products by the gripper, and then return to their starting position (where presumably the recipients are waiting). RT EDDIs are tasked with supporting MRS adaptation in a scenario involving locomotion failure of one of the base robots. In such cases, the other delivery robot should, once it completes its own delivery, takeover the failed one's delivery task. The gripper robot should accordingly relocate to transfer the product. Another potential scenario involves the entrance of a human in the path of one of the delivery robots, in which case the robots should perceive the obstruction and avoid collisions. Further details on the use case are provided in a separate upcoming deliverable planned to document the use case itself and its evaluation.

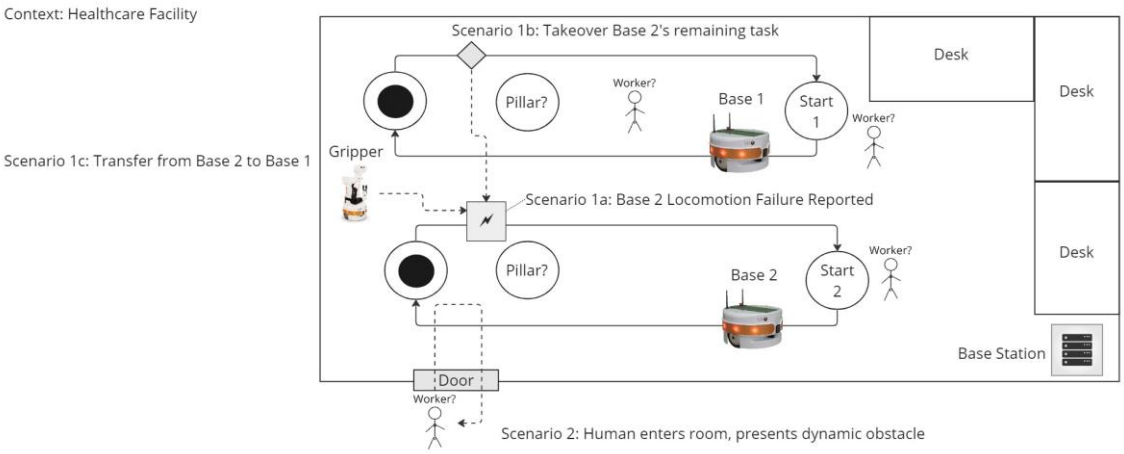


Figure 34 PAL Use Case Overview

PAL robots are planned to use the camera-based person detector for identifying the humans around the shop floor. A well-known Yolov7 model is planned to be used as a person detector. Similar to the KIOS use case, SafeML can be used to monitor the data observed at runtime, and make sure it is within the scope it was designed for. SafeML application on PAL robotics is planned as the next steps.

In the following an example ConSert is shown for the PAL use case. On the robot level, it is guaranteed whether the robot can accomplish its mission, as shown in Figure 35.

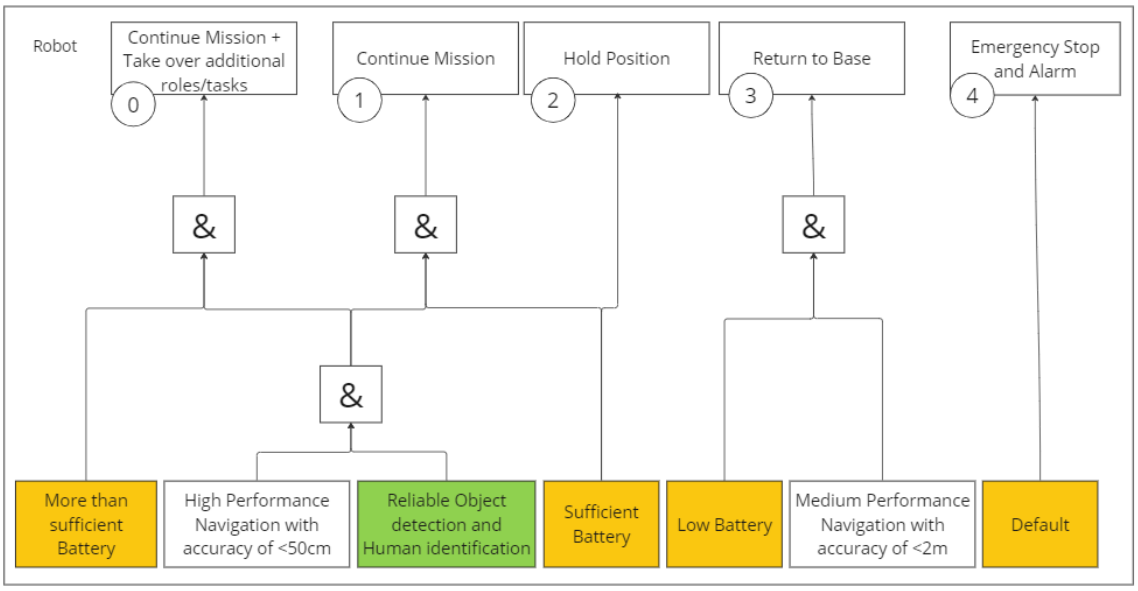


Figure 35 High-level robot ConSert

The robot can guarantee that it can fulfil the mission with and without additional resources, whether it must pause the mission or whether it must abort and return to base. In the worst case, when no other guarantee can be given, the robot must stop and emit an alarm. The robot depends on the energy level of its battery, its navigation capabilities and on the capability to detect humans in danger. The mission can only be achieved if the robot can navigate precisely, it has sufficient energy and it can detect humans. The given guarantee is degraded according to degradation of the above capabilities. Similar to the KIOS use case, a base station could collect the guarantees of all the robots and initiate actions to ensure that the mission can be accomplished. Alternatively, the robot itself can also contact other nearby robots or human operators in case it needs support.

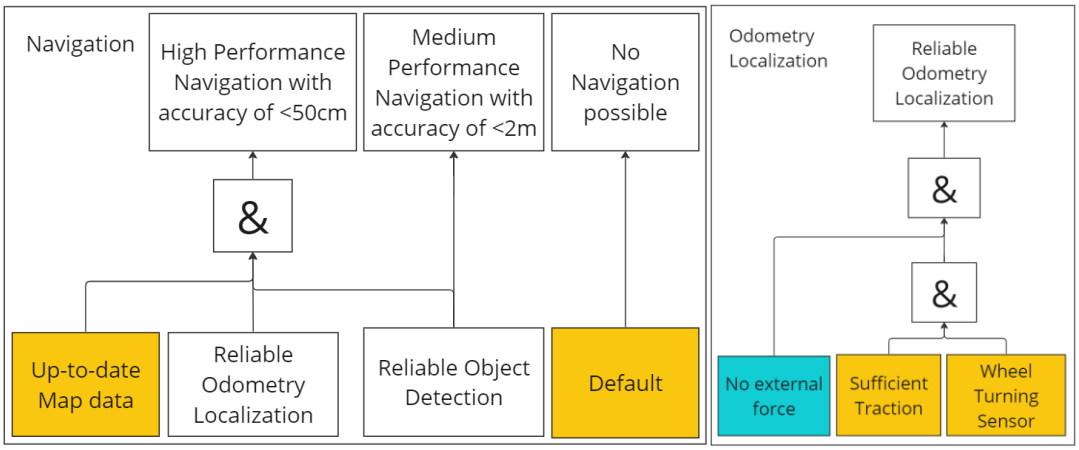


Figure 36 ConSert for the navigation and the odometry localization

In Figure 36 the navigation ConSert is shown. The robot must reliably navigate from start to the destination as specified in the mission. It must determine a feasible path and follow it accordingly. Therefore, it must avoid collision with objects and humans. Therefore, it requires an up-to-date map of the environment and reliable odometry-based localization and object detection. The odometry-based localization module

provides a guarantee for precise and reliable localization in case no external force is applied, it has sufficient traction and the wheel turning sensor is connected.

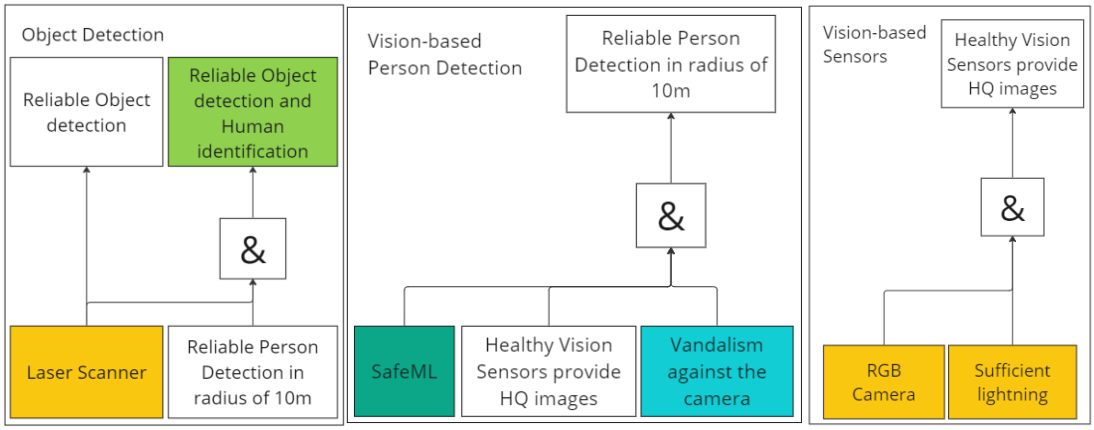


Figure 37 ConSerts for object detection, vision-based person detection and vision-based sensors

In Figure 37 the remaining ConSerts are depicted. As mentioned before, the navigation module depends also on the guarantees of the object detection module. The robot can reliably ensure to detect objects in its trajectory based on a laser scanner. For reliable person detection, additionally the vision-based person detection must guarantee its reliability. This can only be ensured by a small statistical distance according to SafeML, healthy vision-based sensors that provide a high-quality image and if no vandalism happened against the camera. The vandalism refers to any physical damage that can impact its capabilities such as a changed angle, any kind of damage to the lens, etc. A high-quality image can be guaranteed by the vision-based sensor ConSert in case of sufficient lighting conditions and a connected RGB camera.

6. LIMITATIONS

In this section, we discuss limitations we have perceived regarding our current approach and discuss how we plan to address them moving forwards.

- ConSerts are limited due to
    - binary logic semantics; replacing with e.g., Bayesian Networks or more complex logic models more expressive power can be made available, at the cost of additional computation time.
    - lack of explicit time and space semantics, which are important for kinematic risk assessment. These limitations restrict the solution space that ConSert can support directly and limit ConSert applicability; thus, ConSert effectiveness is dependent on the existence of sufficiently developed dependability-related concepts and RtE sources in the host application (robot).
- MAS specification, deployment and orchestration are currently performed ad-hoc on a use-case basis. This can be more explicitly supported in the future; for instance, through inclusion of supporting concepts in the ODE metamodel, and with

corresponding tooling to integrate into the DT to RT EDDI generation pipeline described in SESAME deliverable D7.2.

- We have not proven dependability properties during development; these need to be verified via testing or through other means. Approaches using formal methods and verification could still be leveraged to increase confidence in the developed models.

- The runner component is limited to ConSerts for now but can be extended to support other existing RT EDDI types in the future, as the code generation process is mostly similar.

7. SUMMARY

In this deliverable we present our work in developing Multi-Agent Systems (MAS) powered by Runtime EDDI (RT EDDI) models, in support of dependability for Multi-Robot Systems. We argue that the MAS is a useful abstraction for supporting our overall goal.

We recap the RT EDDI components generated in work described in previous deliverables, and explain how they can be deployed in conjunction as MAS within MRS.

We further discuss and provide work-in-progress examples of how three of the SESAME use cases can leverage these components to help assure MRS dependability. The use case evaluation of the technologies discussed in this deliverable follows in deliverables from WP8 of the SESAME project.

8. REFERENCES

[1] P. Norvig and S. Russell, Artificial Intelligence: A Modern Approach, New Jersey, NY, USA: Pearson Education Inc., 2010.

[2] R. Ben Halima, M. Hachicha, A. Jemal and A. Hadj Kacem, "MAPE-K patterns for self-adaptation in cyber-physical systems," *The Journal of Supercomputing,* vol. 79, no. 5, pp. 4917-4943, 2023.

[3] A. Dorri, S. S. Kanhere and R. Jurdak, "Multi-Agent Systems: A Survey," *IEEE Access,* vol. 6, pp. 28573-28593, 2018.

[4] K. Hanga and Y. Kovalchuk, "Machine learning and multi-agent systems in oil and gas industry applications: A survey," *Computer Science Review,* vol. 34, pp. 100191-100206, 2019.

[5] K. Ahmed, Z. Ismail, I. Shehata, S. Djirar, N. Talbot, S. Ahmadzadeh, S. Shekoohi, E. Cornett, C. Fox and A. Kaye, "Telemedicine, e-health, and multi-agent systems for chronic pain management," *Clinics and Practice,* vol. 13, no. 2, pp. 470-482, 2023.

[6] A. Tour, A. Polyvyanyy and A. Kalenkova, "Agent system mining: vision, benefits, and challenges," *IEEE Access,* vol. 9, pp. 99480-99494, 2021.

[7] Y. Kubera, P. Mathieu and S. Picault, "Everything can be agent!," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 2010.

[8] A. Avizienis, J. 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.

[9] E. Gil, G. Rodrigues, P. Pelliccione and R. Calinescu, "Mission specification and decomposition for multi-robot systems," *Robotics and Autonomous Systems,* vol. 163, 2023.

[10] F. Thompson and D. Guihen, "Review of mission planning for autonomous marine vehicle fleets," *Journal of Field Robotics,* vol. 36, no. 2, pp. 333-354, 2019.

[11] C. Lesire, R. Bailon-Ruiz, M. Barbier and C. Grand, "A Hierarchical Deliberative Architecture Framework based on Goal Decomposition," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 9865-9870, 2022.

[12] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos and M. Carreras, "Rosplan: Planning in the robot operating systems," *Proceedings of the International Conferencce on Automated Planning and Scheduling,* vol. 25, pp. 333-341, 2015.

[13] B. Ferreira, T. Petrovic and S. Bogdan, "Distributed Mission Planning of Complex Tasks for Heterogeneous Multi-Robot Systems," *IEEE 18th International Conference on Automation Science and Engineering,* pp. 1224-1231, 2022.

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

[15] 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.

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

[17] A. Laugros, "Synthetic, Adversarial and Natural Corruptions: Image Classifier Robustness Transfers from one Distribution Shift to an Other," University Grenoble Alpes, Grenoble, France, 2022.

[18] F. Al-Harith, I. Sorokos, A. Schmidt, M. N. Akram, K. Aslansefat and D. Schneider, "Keep Your Distance: Determining Sampling and Distance Thresholds in Machine Learning Monitoring," in *Model-Based Safety and Assessment: 8th International Symposium, IMBSA 2022, Munich, Germany, September 5--7, 2022, Proceedings*, Munich, 2022.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision--ECCV 2014: 13th European Conference, September 6-12, 2014, Proceedings, Part V 13*, Zurich, Switzerland, 2014.

[20] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767,* 2018.

[21] R. Makrigiorgis, P. Kolios, S. Timotheou, T. Theocharides and C. G. Panayiotou, "Extracting the fundamental diagram from aerial footage," in *IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020.

[22] D. Božić-Štulić, Ž. Marušić and S. Gotovac, "Deep learning approach in aerial imagery for supporting land search and rescue missions," *International Journal of Computer Vision,* vol. 127, no. 9, pp. 1256-1278, 2019.

[23] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu and H. Ling, "Detection and Tracking Meet Drones Challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2021.