Mälardalen University
M.Sc.Eng. Dependable Aerospace Systems
Västerås, Sweden

Project Course in Dependable Systems
22.5 credits

Final Report

Intelligent Replanning Protocol for a Fail-Operational Drone Swarm

Responsible

Andrea Haglund
ahd20002@student.mdu.se

Claire Namatovu
cnu21001@student.mdu.se

Emily Zainali
ezi21001@student.mdu.se

Esaias Målqvist
emt21001@student.mdu.se

Yonatan Michael Beyene
yme21001@student.mdu.se

Examiner: Luciana Provenzano

December 7, 2025

Title: Intelligent Replanning Protocol for a Fail-Operational Drone Swarm		ID: CE-05 Version: 1.0
Authors: Andrea Haglund[1], Claire Namatovu[2], Emily Zainali[3], Esaias Målqvist[4], Yonatan Michael Beyene[5]	Role: [1]Chief Engineer, [2]Requirements Manager, [3]Verification & Validation Manager, [4]Safety Manager, [5]Quality & Configuration Manager	Page 1 of 16

DOCUMENT APPROVAL

Name	Role	Version	Date	Signature
Andrea Haglund	Chief Engineer	1.0	2025-12-07	[illegible]
Yonatan Michael Beyene	Q&C Manager	1.0	2025-12-07	[illegible]

DOCUMENT CHANGE RECORD

Version	Date	Reason for Change	Pages / Sections Affected
0.1	2025-12-05	Version for review	
1.0	2025-12-07	Version for public release	All

Contents

Glossary

Activity Diagram
A visual representation of a workflow, showing the sequence of actions and decisions within a process. 7

ARC
initial Air Risk Class. 8

ARP4761A
Aerospace Recommended Practice (ARP) providing guidelines for conducting the safety assessment process on civil aircraft systems and equipment, covering techniques such as FHA, FMEA, and FTA to support system safety analysis throughout the development lifecycle. 8

BDD
Block Definition Diagram. 7

CE
Chief Engineer. 7, 10, 12

DSSA
Drone Swarm Safety Assessment. 8

FHA
Functional Hazard Assessment. 3, 8

FMEA
Failure Modes and Effects Analysis. 3

FTA
Fault Tree Analysis. 3, 8

IBD
Internal Block Diagram. 7

iGRC
intrinsic Ground Risk Class. 8

IL
Integrity Level is a value given based on the project's unique characteristics (complexity, criticality, risk, safety, etc.) that represents the importance of the system and software to the user. *(Taken from the standard IEEE 1012-2024 [1]).* 9

IRDS
Intelligent Replanning Drone Swarm. 5–7, 10, 13, 15

PDSSA
Preliminary Drone Swarm Safety Assessment. 8

protocol
Set of rules or procedures that defines how a system communicates. 5

SAIL
Specific Assurance and Integrity Level. 8

SAR
Search and Rescue. 5, 10

SDD
System Design Description. 7, 10, 12

Search Area
Entire region to be searched. 3

sector
Subdivision of a Search Area. 5

SORA	JARUS guidelines on Specific Operations Risk Assessment (SORA). A structured methodology to assess and mitigate risks for specific UAS operations, ensuring safe conduct in complex or beyond-visual-line-of-sight environments. 8
State Diagram	A visual representation of a state machine, showing a system's possible states and the transitions between them triggered by events. 7
TMPR	Tactical Mitigation Performance Requirement. 8
UAS	Unmanned Aircraft System. 4
UAV	Unmanned Aerial Vehicle. 5, 8–10, 13
V&V	Verification & Validation. 9

1 Introduction

Search and Rescue (SAR) missions increasingly rely on autonomous multi agent systems that are capable of covering large areas quickly and safely. Although Unmanned Aerial Vehicle (UAV) swarms offer significant advantages in robustness and scalability, their reliability is still constrained by the health of individual UAVs. This project, Intelligent Replanning Drone Swarm (IRDS), addressed this limitation by developing a system concept for a fail-operational UAV swarm with replanning abilities, enabling the mission to continue efficiently even when individual drones experience partial failures.

The project focused on designing and implementing the protocol module that allows UAVs to communicate their health status, redistribute tasks, dynamically reassign sectors, and assign failed agents to secondary roles when possible. The key objective was to create a decentralised and intelligent replanning mechanism that enables the swarm to maintain operational capability throughout the mission.

1.1 Objectives & Goals

The main objective of this project was to design, implement, and validate a fail-operational intelligent replanning protocol for a decentralised swarm of UAVs performing SAR missions.
The goals of this project were [2]:

- Develop a decentralised replanning protocol that enables a UAV swarm to adapt collectively when individual agents experience degraded health
- Design and implement a safe consensus mechanism that allows all agents to agree on a new mission plan after fault detection
- Create adaptive task reallocation logic to redistribute tasks from compromised agents to healthy agents, and assign supportive roles to compromised agents
- Validate through simulation with fault injection, measuring improvements in mission continuity, resilience, and efficiency

2 Methodology

2.1 Project Phases

The project followed the three phases defined in the course: Pre-study & planning, execution & validation, and reporting.

In the pre-study & planning phase, the team analysed the project context and baseline architecture, studied relevant standards and related work, defined the dependability perspective, and produced the project and management plans that guided the rest of the work.

During the execution & validation phase, these plans were used to specify, design, and partly implement the IRDS protocol module, develop the requirements and system design description, and validate the protocol through simulation and model checking with fault injection.

In the reporting phase, the team consolidated results and lessons learnt, updated all project artefacts based on reviews and feedback, and produced this final report, providing an integrated view of the technical outcome and the dependability activities performed.

2.2 Tools & Software

2.2.1 Database

The database was developed using SQLite and integrated with Python through the sqlite3 library. It consists of multiple normalised tables with defined primary and foreign keys to maintain relational integrity. The basic CRUD (Create, Read, Update, Delete) functions were implemented for efficient data management, and indexing was applied to optimise query performance. The design ensures data validation and integrity, while error handling mechanisms provide reliability. The database seamlessly interacts with the application backend, enabling real-time data retrieval and updates.

The requirements database served as a central tool for managing, storing, and analysing all system and safety requirements throughout the project. Its purpose was to enforce structural consistency, enable reliable traceability, and support systematic verification of requirement completeness and correctness. To achieve this, the database was implemented in Python, using a schema designed specifically to reflect how requirements are organised and how compliance with traceability guidelines defined in ISO/IEC/IEEE 29148:2018, is ensured.

The schema of the database was constructed around a set of interdependent tables representing requirements, their attributes, their parent–child relationships, and their associated verification methods. By relying on relational constraints, the tool ensured that no requirement could reference a non-existent parent, that trace links were always valid, and that requirements marked as "satisfied" were always accompanied by a defined verification method.

To support the workflow of the Requirements Manager and the V&V Manager, the tool included functionality for automated visualisation of requirement hierarchies. Requirement IDs could be exported as a graph-like structure representing parent–child decomposition, facilitating quick inspection of traceability chains and helping identify missing, inconsistent, or redundant requirements early in the process. Additional features allowed exporting the content of the database in multiple formats—such as CSV for documentation and review—while deliberately omitting binary fields, such as files attached to document entries, to avoid clutter and maintain readability in exported specifications.

2.2.2 Modelling

The conceptual models and designs of the IRDS system were made using a combination of informal and formal notation. In the early stages of the project, draw.io [3] was used for pre-design sketches and to visualise architectural ideas that were not yet mature enough for formal SysML modelling. These informal diagrams were particularly helpful for explaining concepts that could not be intuitively understood from text alone or that would have been cumbersome to express directly in SysML, such as early ideas about neighbourhoods, sector allocation, and rumour flows.

Once the main concepts had stabilised, the protocol and system architecture were modelled using Modelio Open Source 5.4.1 [4] with the SysML Architect 5.4.0 module (SysML 1.2). Block Definition Diagrams (BDDs), Internal Block Diagrams (IBDs), Activity Diagrams, and State Diagrams were created in Modelio to provide a consistent, formally structured description of the IRDS System of Systems, and the protocol module and its interaction with other systems, as documented in the System Design Description [5]. The combination of draw.io for exploratory pre-design and Modelio for formal SysML models allowed the team to move smoothly from rough sketches to traceable, reviewable design artefacts.

For formal behavioural verification, UPPAAL [6] was used as a model checking tool to demonstrate and verify that the system correctly handled a Byzantine fault. The model was built with relevant parts of the system and their possible states, and the scenario was simulated to analyse the system behaviour under incorrect or unpredictable events. Using model checking, it was possible to verify that the system met the specified requirements even in the occurrence of a Byzantine fault.

2.2.3 Simulation

The simulation tool served as the primary platform for testing the fail-operational architecture, validating the replanning logic, injecting faults, and observing swarm behaviour under controlled conditions. The simulation tool used was called PyBullet, which is a physics simulation for robotics, games, visual effects and machine learning [7]. The simulation environment builds on the pybullet_search_rescue_uavs GitHub repository [8], which served as the starting point before being significantly extended and adapted to meet the project's requirements.

2.3 Method by Roles

2.3.1 Chief Engineer

The work of the Chief Engineer (CE) focused on coordinating technical activities and maintaining a coherent system-level view of the Intelligent Replanning Drone Swarm. The role ensured that contributions from the different work areas (requirements, safety, verification and validation, quality, and configuration) were aligned with the overall architectural vision and the project goals.

From an organisational perspective, the planned activities for each role were collected and structured into a unified overview of activities, which facilitated the identification of dependencies between activities and helped ensure that the results of one area were available in time to support others, for example, that requirements and safety analysis could inform system design and documentation. Although this top-down coordination approach provided a clear overview of responsibilities, it also revealed limitations, as activity definitions were initially developed role by role rather than in a fully joint session, which occasionally led to gaps and overlaps between work areas.

For technical design work, a combination of informal and formal modelling tools was used. During pre-design, draw.io [3] was used to sketch architectural concepts and visualise protocol mechanisms that were not yet mature enough for formal SysML modelling or were difficult to communicate intuitively through text alone. The pre-design sketches were helpful in exploring alternatives, enabled rapid iteration of ideas, and supported early discussions. Once the architecture had stabilised, Modelio [4] was used for formal SysML modelling, which included BDDs, IBDs, Activity Diagram, and State Diagram. Using Modelio for formal SysML modelling made it possible to capture the conceptual design of the system in a precise and consistent way and provided a basis for the System Design Description (SDD) [5].

2.3.2 Requirements Manager

The requirements management process began with a detailed review of ISO/IEC/IEEE 29148:2018 [9], which outlines industry best practices for requirements engineering in both systems and software development. This standard served as the primary reference for planning the entire management approach. Each component of the Requirements Management Plan [10] such as change control, traceability strategy, configuration management, verification planning, and quality assurance procedures was derived directly from the key principles and activities highlighted in the standard.

Given the strong emphasis ISO 29148 places on bidirectional traceability, establishing robust traceability mechanisms became a central priority throughout the process. Effective traceability was essential not only to demonstrate compliance, but also to support impact analysis, maintain requirement consistency, and ensure that nothing was lost during decomposition or refinement.

To support the aforementioned traceability mechanism, a lightweight custom requirements management tool was developed. The decision to build a dedicated tool arose from limitations in the available options, most notably Excel—which, while convenient, lacks essential relational features such as enforced linkages between parent and child requirement IDs. In contrast, a relational database provides inherent structural guarantees similar to SQL foreign keys, ensuring that trace links cannot reference non-existent requirements.

By integrating SQL with Python , the tool was expanded to include more advanced analytical capabilities (see section 2.2.1 Database).

Anyone seeking a deeper understanding of the motivations for particular attribute definitions, table structures, or traceability design choices can refer to the Requirements Management Plan [10], where these aspects are explained in full.

2.3.3 Safety Manager

The safety process followed in the project was based on the ARP4761A system-development safety frameworks [11] and the SORA operational risk methodology [12], as defined in the Safety Management Plan [13]. ARP4761A and SORA together guided the identification of hazards, the classification of risks, and the determination of assurance levels for both system-level and operational safety.

The safety activities of ARP4761A and SORA were selected based on their relevance to the scope of the project, i.e., activities not applicable to UAV simulation, swarm behaviour, or operational risk assessment were excluded. The selected activities, as outlined in Section 5 of the Safety Management Plan [13], were:

- Functional Hazard Assessment (FHA),
- Preliminary Drone Swarm Safety Assessment (PDSSA),
- SORA-based risk classification, including identifications of:
  - intrinsic Ground Risk Class (iGRC).
  - initial Air Risk Class (ARC).
  - Tactical Mitigation Performance Requirement (TMPR).
  - Specific Assurance and Integrity Level (SAIL).
  - Containment Requirements.
- Drone Swarm Safety Assessment (DSSA) using Fault Tree Analysis (FTA),
- development of a Preliminary Safety Assurance Case, and
- development of a Final Safety Assurance Case.

The results of the aforementioned activities were used to derive safety goals and safety requirements for the project, which were documented and transferred to the project's requirements database to ensure full traceability. The final Safety Assurance Case consolidates all safety evidence and demonstrates how the safety goals have been satisfied.

2.3.4 Verification & Validation Manager

The Verification & Validation (V&V) management process began with a detailed review of the standards that were most suitable, based on understanding the role of a V&V manager. The chosen standards were:

- IEEE Std 1012™-2024 [1]
- IEEE Std 29119-1:™-2022 [14]
- IEEE Std 29119-4:™-2021 [15]
- IEEE Std 15288:™-2023 [16]

All chosen standards played different roles, depending on the phases of the process. The standards were tailored according to the scope of the project and provided the framework for how the activities of V&V would be structured, documented, and implemented during all phases of the project.

After determining the standards and IL, the V&V methods were selected. Dynamic testing via simulations in gym-pybullet drones and model checking via UPPAAL were identified as the most effective ways to verify the system's behaviour and decision logic. Analytical methods, such as requirements review, checklists, and traceability analysis, were used to ensure the quality, clarity, and completeness of the requirements. These choices were guided by the methods defined in the method section of the V&V management plan, where tests, simulations, traceability analysis, and requirements review constitute the main verification methods of the project.

The verification of the system requirements followed the structure according to IEEE 1012-2024 recommendations. Once the requirements had been established, a traceability analysis was performed, where each requirement was analysed.

The test cases were developed with a focus on the requirements that had the highest criticality and the greatest impact on the system's function and risk level. The choice of focus was strategic and was motivated by the fact that the lower-level requirements (system and subsystem requirements) were embedded and directly dependent on the Drone Swarm requirements. By successfully verifying the system's operation at the most complex and critical overall behaviours, it was ensured that the underlying component requirements were also met, and by testing these requirements first, it ensured that any errors in critical parts of the protocol were detected and addressed early, which is also in line with the IL3 requirements and the standards' recommendations.

The validation and verification of the system, which ensured that the replanning protocol met the overall goals of the users and the project, was carried out mainly through simulations of a UAV swarm in various scenarios, including degraded conditions and dynamic missions. Since the project result is an abstract protocol rather than a physical product, the traditional division between verification and validation blended. The validation was based directly on the project goals and system requirements, sourced directly from the project description, which meant that the validation was largely baked into the verification of the system design, where successful verification of the desired behaviour in the simulation environment also constituted evidence that the protocol design met the drone swarm goals.

The entirety of the method ensures that both the system has been built correctly (verification) and that the system as a whole functions as intended (validation), fully in accordance with the structure of the V&V plan and the IL3 requirements. For a deeper understanding, please read the V&V management plan [17].

2.3.5 Quality & Configuration Manager

Relevant quality and configuration management standards were first reviewed to establish a clear understanding of the responsibilities, processes, and required output associated with the role. The standards used were ISO 9001 [18], ISO/IEC/IEEE 15288 [16], ISO/IEC/IEEE 15289, ISO 10007 [19] and ISO/IEC 25002 [20]. Jira was used to track tasks and identify items that required review. When reviewing items, a formal review protocol was used that was developed to ensure a consistent evaluation with respect to completeness, correctness, and traceability. For a deeper understanding of the Quality & Configuration management process, refer to [21] and [22].

3 Results

3.1 Summary

This project resulted in the conceptual design and partial implementation of the Intelligent Replanning Drone Swarm (IRDS), a decentralised protocol that enables a UAV swarm to remain fail-operational during Search and Rescue missions. The work produced a structured requirements specification, safety goals and safety requirements, a system-level architecture captured in SysML models and the System Design Description, and a set of verification and validation artefacts aligned with Integrity Level 3. Formal model checking in UPPAAL and simulations in a PyBullet-based swarm environment with injected degradations and failures showed that the protocol can detect degraded UAVs, reallocate tasks, and maintain coordinated mission progress in the scenarios tested, while also highlighting limitations in the current implementation and tooling that are discussed in later sections.

3.2 Results by Roles

3.2.1 Requirements Manager

The application of the requirements' management methodology resulted in a well-structured specification that allowed for easy navigation and retrieval of information, and it also improved the analysis of traceability by providing an alternative view of the relationships between requirements. The complete specification is available in [23]. Alternatively, to view the specification in the database, please follow steps 1–4 under section **2) Start inserting data** in the **README.md** file in [24].

When changes to the requirements were necessary, the hierarchical structure facilitated the rapid identification of affected requirements and associated documents, ensuring efficient updates.

However, implementing the tool introduced challenges that simpler solutions, such as Excel, would not typically present. These included the complexity of the underlying code and the need to correct multiple files when errors were detected. Additionally, only two team members had sufficient Python programming expertise to address these issues, which limited the team's responsiveness.

3.2.2 Safety Manager

The safety activities conducted throughout the project led to the identification of a set of safety goals and safety requirements. The final safety requirements provide a structured foundation for demonstrating compliance with applicable EU aviation regulations. The resulting goals and requirements form the basis for risk mitigation in all identified hazards and failure conditions, and their effectiveness and contribution to overall system safety are ultimately demonstrated within the Safety Assurance Case [25], which presents the structured argument and supporting evidence that the system achieves an acceptable safe level of operation.

3.2.3 Chief Engineer

The work of the CE resulted in a conceptual architecture at the system-level for the IRDS and in the integration of contributions from all roles into a consistent set of deliverables. The general concept of operations and the structure of the protocol module were captured and refined in the SDD [5], which combined requirements, safety considerations, and verification and validation activities into a single system-level view. Using the SDD as the central system-level document and by coordinating the contributions from all roles, it was ensured that the decentralised replanning protocol, the simulation environment and the management plans were aligned with the project objectives and with each other, rather than evolving as isolated items.

The modelling effort produced a complete set of conceptual and formal representations that describe the protocol module and the IRDS as a system of systems. The formal models provided traceability from high-level requirements to protocol-level mechanisms such as rumour-based consensus, sector allocation, and agent role transitions. The resulting models formed the technical foundation for the definition of simulation scenarios and verification cases, enabling a structured link between system design and validation activities.

However, the coordination and modelling work also revealed certain limitations. Some architectural aspects, such as detailed pathing logic and low-level communication behaviour, were only defined conceptually due to the scope and time constraints of the project. Additionally, several modelling decisions had to be simplified to maintain consistency and readability across the SysML diagrams. Despite the constraints, the coordination and modelling work provided a unified and technically consistent definition of the system that guided the development of all project artefacts.

3.2.4 Verification & Validation Manager

The results of the verification and validation activities are based on the V&V plan (VV-01) established at the beginning of the project and were carried out in accordance with IEEE Std 1012-2024, IEEE Std 29119-1:2022, IEEE Std 29119-4:2021 and IEEE Std 15288:2023. All activities were carried out in accordance with Integrity Level 3 (IL3), which requires traceability, structured documentation and formal verification steps.

Replanning logic, detection of degraded UAVs, task reallocation and swarm coordination, were verified mainly through simulated scenarios in gym-pybullet drones. The results showed that the protocol follows the decision-making process and behaviors specified. No deviations were identified within the parts of the system covered by the project.

The test cases developed according to IEEE 29119-4 focused on the requirements with the highest criticality according to the IL3 classification. Simulations performed showed that the protocol could correctly identify degraded UAVs, initiate replanning according to defined decision criteria, and reallocate task allocation between UAVs in a manner that complies with the requirements. The results confirm that the protocol exhibits correct functionality in the scenarios tested.

The traceability matrix showed complete connection between requirements, verification method and results. All V&V related deliverables (VV-02 to VV-13) have been produced in accordance with the IL3 requirements.

The protocol is therefore assessed to meet all defined requirements within the scope of the project, and the results support the conclusion that the protocol is ready to be implemented and tested in a future UAV system environment.

3.2.5 Quality & Configuration Manager

The specified deliverables of a quality & configuration manager was created, which included review protocols for all the deliverables created in this project, and configuration change requests. A total of 14 deliverables were reviewed using the review protocol, and in these reviews, some issues were identified that were primarily related to template misuse and unclear assumptions. All issues were solved before being approved for baseline. The review protocol and configuration change request process resulted in clear, traceable, and well structured documentation that ensured that the decentralised replanning protocol module and simulation environment were developed in a controlled manner, with improved reliability and reduced ambiguity across all deliverables.

4 Discussion

Several lessons were identified during the project, related to both technical work and project organisation.

One of the key lessons learnt is the importance of realistic planning within the available time frame, as the planning provides direction for the entire development process. Dependable systems often operate in contexts where the stakes are extremely high, for example when failure could lead to loss of life or significant financial impact. For this reason, planning decisions must be well-grounded in established standards, sound engineering judgement, or similar best practices.

The method used to define project activities also revealed limitations. Activities were collected individually from each role and later consolidated by the CE, which led to gaps, overlaps, and inconsistent understanding of responsibilities. A joint planning session would have produced a more coherent and aligned activity structure. Similar issues arose with deliverables and artefacts, as their definitions and expected outputs were not agreed upon early, resulting in ambiguity later in the project. Establishing these jointly at the beginning would have improved coordination and progress tracking.

Another key lesson was the value of establishing the SDD [5] from the outset and maintaining it as a living document. Because the SDD was developed later in the process, modelling activities initially lacked a clear architectural structure, and it became evident that drafting the SDD earlier and treating it more like a living document would have clarified which SysML diagrams were required and how the IRDS architecture should be presented.

Working with requirements reinforced the benefits of maintaining abstraction. Abstraction not only fosters creativity and innovation among implementers, but also supports design flexibility and simplifies validation by focusing on design principles rather than implementation details. The work with traceability reinforced its importance, as it proved essential to maintain consistency and avoid unnecessary revisions by ensuring that each requirement remained linked to its origin and rationale. By clearly demonstrating the origin and purpose of each requirement, consistency could be maintained and alignment with project objectives ensured.

Finally, the project underscored the challenges associated with introducing new tools in the context of dependable systems. The development of a new tool should be strongly motivated by the need to address the gaps existing tools cannot fill; otherwise, their adoption will lack justification. Any new tool must be justified, certified where applicable, and demonstrably necessary, as tool adoption introduces integration, compatibility, and training burdens that can affect both timelines and resource allocation. The risks associated with new tools should not be taken lightly in regard to dependable systems unless the benefits clearly outweigh the trade-offs.

In conclusion, the lessons learnt illustrate that the structure of the work, the clarity of documentation, and the methodological choices made throughout development have a direct impact on the efficiency and quality of the final system.

5 Future Work

Future work on the IRDS can be divided into two key areas: Technical extensions of the IRDS architecture and improvements to the development workflow and supporting tools.

5.1 Protocol and System Development

All components of the IRDS remain conceptual and require full implementation. The Rumour Mill, Sector Allocation Market, and Task & Role Allocation mechanisms must be translated into deployable onboard software, including concrete message formats, memory-efficient data structures, and embedded integration on UAV microcontrollers. Further development is also needed for the SwarmInterface, including map-based mission tools, visualisation of swarm status, and operator controls suitable for real SAR deployments.

Additional technical work includes integrating navigation and path-planning logic with the protocol, expanding the behaviours of degraded agents, and introducing basic cryptographic protection for rumour messages. Scaling studies should be performed to understand performance in larger swarms, and formal verification should be applied to confirm consensus properties, fault-tolerance guarantees, and safety behaviour under adverse conditions. Long-term work may also involve testing the protocol in physical UAVs to evaluate real-world communication, flight dynamics, and reliability.

For more details on future work, see Future Work in [5].

5.2 Tooling and Development Process Improvements

One challenge identified during the project was the complexity of synchronising updates to the requirements database. Retrieving changes required users to pull the updated feature branch, reconfigure the local database instance, and overwrite existing content, while uploading changes required exporting the database to JSON and committing the file to the repository. For new users, this procedure introduced unnecessary complexity and increased the risk of mistakes.

A potential improvement is to adopt a cloud-based platform that supports collaborative database management. Such a solution would reduce the number of manual steps required for synchronisation, although the storage limitations of free service tiers would need to be considered when selecting an appropriate platform.

The implementation of the database itself can also be strengthened. The current version supports basic operations such as insertion, deletion, and constraint enforcement (e.g., ensuring that a ”satisfied” status has an associated method), but future work could extend it with functions that automatically assess requirement quality against ISO 29148 guidelines by detecting inappropriate terms, such as vague adjectives (”better”) or indefinite quantifiers (”some”), before they are added to the database. Additional enhancements may include syntax checking or authoring-assistance features to support a consistent and compliant requirement formulation within a selected requirement specification syntax.

Improvements and enhancements to the simulation software would further strengthen future developments, allowing closer alignment between requirements, architectural models, and validation activities. These potential improvements are:

- Implement the consensus architecture Rumor Mill and display it on the Graphical User Interface (GUI).
- Implement a system that shows battery drainage during flight and ensures that, depending on the distance from home, the drones return before the battery is completely depleted.

There are some issues in the simulation that, due to time constraints, could not be addressed. These issues are:

- If a drone reaches its assigned section first, locates the subject, and triggers the voting system before the closest drones have reached their own sections, those drones will still prioritize reaching their assigned area instead of responding to the drone that initiated the voting.
- If there are four drones and one crashes, it cannot participate in the vote when the subject is found, since three functioning drones are required. (Currently, the drones simply fly around the subject and nothing happens if there are not three available drones.) This could be resolved by starting a timer once a drone initiates the voting system; if the timer exceeds a certain threshold, all drones should return home.

- If a fault is injected into a drone, it will resume flying if selected as one of the three drones required for voting, as long as it has not crashed. This means that even a drone that performed an emergency landing may start flying again. A straightforward solution is to exclude any drone that has executed an emergency landing from the selection of the closest drones required for voting.
- Some drones drift slightly after landing. This appears to be related to the physics engine of the PyBullet environment.
- If the fault “LOW battery” is injected, the drone returns home and can then be recharged using the charge drone button, after which it will return to the search area. However, if the same fault is injected again, either to the same drone while it is returning or to another drone, the charge button becomes available once the drone is home, but pressing it produces no effect.

6 Conclusion

This project developed the conceptual design of the Intelligent Replanning Drone Swarm (IRDS) and examined its suitability to support fail-operational behaviour in UAV-based Search and Rescue missions. Through the integration of decentralised consensus, sector allocation, and task and role selection, the proposed architecture demonstrates how a swarm can maintain mission continuity despite individual agent degradation or failure. The modelling, requirements work, safety considerations, and verification activities collectively show that the IRDS architecture satisfies the project's objectives and provides a coherent foundation for future development.

The execution of the project also provided valuable experience in coordinating work in several dependability-focused roles and managing the interactions between requirements, safety analysis, system design, verification, configuration management, and quality management. The process highlighted the importance of early alignment with activities and deliverables, consistent documentation practices, and clear communication between roles. These organisational insights were essential for maintaining coherence in the project's artefacts and to ensure that methodological work effectively supported the development of the system architecture.

Although components of the IRDS remain conceptual and require implementation, validation, and refinement on embedded hardware, the results indicate that the project has achieved its intended goals. Overall, the work establishes a clear basis for future research and continued development of dependable, decentralised UAV swarm systems.

References

[1] Institute of Electrical and Electronics Engineers (IEEE), *Standard for System, Software, and Hardware Verification and Validation*, IEEE Std 1012™-2024, Nov. 2024. [Online]. Available: https://standards.ieee.org/ieee/1012/7324/

[2] A. Haglund, *Project Plan*, Intelligent Replanning Drone Swarm, Dec. 6 2025, Version 1.2.

[3] *draw.io*. 29.2.2, draw.io Ltd and draw.io AG. [Online]. Available: https://github.com/jgraph/drawio

[4] *Modelio Open Source*. 5.4.1. [Online]. Available: https://www.modelio.org/index.htm

[5] A. Haglund, *System Design Description*, Intelligent Replanning Drone Swarm, Dec. 7 2025, Version 2.0.

[6] *UPPAAL*. 5.0. [Online]. Available: https://uppaal.org/

[7] Y. B. Erwin Coumans, "Pybullet quick start guide," 2017, [Online; accessed 2025-12-02]. [Online]. Available: https://github.com/bulletphysics/bullet3/blob/master/docs/pybullet_quickstart_guide/PyBulletQuickstartGuide.md.html

[8] L. Giacomossi, "pybullet search rescue uavs," [Online; accessed 2025-12-02]. [Online]. Available: https://github.com/luizgiacomossi/pybullet_search_rescue_uavs

[9] International Organization for Standardization (ISO), *Systems and software engineering — Life cycle processes — Requirements engineering*, ISO/IEC/IEEE 29148:2018, Nov. 2018. [Online]. Available: https://www.iso.org/standard/72089.html

[10] C. Namatovu, *Requirements Management Plan*, Intelligent Replanning Drone Swarm, Nov. 30 2025, Version 1.2.

[11] SAE International, *Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment*, ARP4761A, Dec. 2023. [Online]. Available: https://www.sae.org/standards/arp4761a-guidelines-conducting-safety-assessment-process-civil-aircraft-systems-equipment

[12] Joint Authorities for Rulemaking on Unmanned Systems (JARUS), *JARUS guidelines on Specific Operations Risk Assessment (SORA)*, JAR-DEL-SRM-SORA-MB-2.5, May 2024, Accessed: Sep. 30, 2025. [Online]. Available: https://jarus-rpas.org/wp-content/uploads/2024/06/SORA-v2.5-Main-Body-Release-JAR_doc_25.pdf

[13] E. Målqvist, *Safety Management Plan*, Intelligent Replanning Drone Swarm, Nov. 14 2025, Version 1.1.

[14] International Organization for Standardization (ISO), *Software and systems engineering — Software testing - Part 1: General concepts*, ISO/IEC/IEEE 29119-1:2022, Jan. 2022. [Online]. Available: https://www.iso.org/standard/81291.html

[15] ——, *Software and systems engineering — Software testing - Part 4: Test techniques*, ISO/IEC/IEEE 29119-4:2021, Oct. 2021. [Online]. Available: https://www.iso.org/standard/79430.html

[16] ——, *Systems and software engineering — System life cycle processes*, ISO/IEC/IEEE 15288:2023, May 2023. [Online]. Available: https://www.iso.org/standard/81702.html

[17] E. Zainali, *Validation & Verification Managment Plan*, Intelligent Replanning Drone Swarm, Dec. 5 2025, Version 1.1.

[18] International Organization for Standardization (ISO), *Quality management systems — Requirements*, ISO 9001:2015, Sep. 2015. [Online]. Available: https://www.iso.org/standard/62085.html

[19] ——, *Quality management — Guidelines for configuration management*, Mar. 2017. [Online]. Available: https://www.iso.org/standard/70400.html

[20] ——, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality model overview and usage*, ISO/IEC 25002:2024, Mar. 2024. [Online]. Available: https://www.iso.org/standard/78175.html

[21] Y. M. Beyene, *Quality Management Plan*, Intelligent Replanning Drone Swarm, Nov. 13 2025, Version 1.2.

[22] ——, *Configuration Management Plan*, Intelligent Replanning Drone Swarm, Nov. 13 2025, Version 1.1.

[23] GitHub.com, Dec. 2025, intelligent-Drone-Swarm/documents/requirements. [Online]. Available: https://github.com/MDU-C2/Intelligent-Drone-Swarm/tree/main/documents/requirements

[24] A. Haglund, "Beginner-friendly guide to the database," GitHub.com, Dec. 2025, intelligent-Drone-Swarm/database/README.md. [Online]. Available: https://github.com/MDU-C2/Intelligent-Drone-Swarm/tree/main/database

[25] E. Målqvist, *Safety Assurance Case*, Intelligent Replanning Drone Swarm, Dec. 5 2025, Version 1.1.