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

Project Course in Dependable Systems
22.5 credits

Requirements Specification Guide

Responsible
Claire Namatovu
cnu21001@student.mdu.se

Examiner: Luciana Provenzano

November 28, 2025

DOCUMENT APPROVAL

Name	Role	Version	Date	Signature
Andrea Haglund	Chief Engineer	1.1	2025-11-28	
Yonatan Michael Beyene	Q&C Manager	1.1	2025-11-28	YMB

DOCUMENT CHANGE RECORD

Version	Date	Reason for Change	Pages / Sections Affected
0.1	2025-11-01	Version for review	
1.0	2025-11-10	Version for public release	All
1.1	2025-11-25	A guide for checking for completeness	8

Contents

Glossary

agent

Single UAV that is a member of a UAV swarm.. 4

EARS

Easy Approach to Requirements Syntax. 6, 7

IRDS

Intelligent Replanning Drone Swarm. 4, 7

RRP

Requirements Review Protocol. 7

RS

Requirements Specification. 4, 6

1 Introduction

This document provides an overview of the requirements specification for the communication and replanning protocol developed under the Intelligent Replanning Drone Swarm (IRDS) project. It serves as a foundational reference to ensure clarity and common understanding of the system's intended functionality and constraints.

1.1 Purpose

The purpose of this document is to guide readers in understanding and interpreting the Requirements Specification (RS) for the communication and replanning protocol of the IRDS project. It provides an overview of how the requirements are organised, categorised, and presented, and explains the conventions used throughout the specification.

1.2 Scope

The RS focuses only on the software component of the protocol including aspects like agent management, task assignment, data exchange, and agent monitoring. Hardware design and physical platform details fall outside the scope of the specification.

2 Project Objectives

To understand where the requirements are derived from, it is important to re-highlight the objectives of the project. The requirements outlined in the specification are not arbitrary; they are directly informed by the overall goals and intended outcomes of the project.
The objectives of the project are the following:

- Develop a decentralised replanning protocol that enables a UAV swarm to adapt collectively when individual agents experience degraded health.
- Design and implement safe consensus mechanism allowing 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.
- Design a decentralised protocol for secure and intelligent mission replanning.
- Validate through simulation with fault injection, measuring improvements in mission continuity, resilience, and efficiency.

These objectives were decomposed using the KAOS model [1] to support the formulation of the goals. This model can be found in the documents folder with the filename Kaos-Goals.drawio.png

3 Specification Structure & Usage

The RS can be opened in Excel, where each table is presented as a separate sheet. When reviewing the document, it is recommended to begin with the higher-level requirements before proceeding to the more detailed, lower-level ones.

The requirements are written using the Easy Approach to Requirements Syntax (EARS). To maintain clarity and avoid ambiguity, conjunctions such as "and" or "or" were deliberately avoided. As a result, some requirements may appear similar; this occurs when a single combined statement (e.g., "the procedures and rules") has been separated into two distinct requirements — one describing the procedures and the other describing the rules. The following subsection outlines where to find the relevant data, tables, and requirements.

3.1 Navigating the Requirements

The table below shows where each type of requirement can be found.

Type of requirement	**Table name**
Goals	goals
Drone-swarm related requirements	drone_swarm_requirements
Protocol (software module) requirements	system_requirements
Components of the protocol software module	subsystem_requirements

Table 1: Location of Requirement Types.

3.2 Navigating the Supporting Tables

The table below presents the supporting tables along with a description of their respective roles.

Table name	**Description**
id_glossary	Contains the meaning of each prefix used and helps users locate specific requirements by identifying their associated prefix.
goal_children	Contains the relationship between the different goals and their derived swarm requirements
sys_req_children	Contains the relationship between the different protocol module requirements and their derived sub-module requirements.
swarm_req_children	Contains the relationship between the swarm requirements and their derived system (protocol module) requirements.
test_and_verification	Contains the description of each test /verification method used.
v_join_documents	Contains the mapping for each verification method to its corresponding document.
documents	Contains the filenames of the documents supporting test and verification.

Table 2: Supporting tables and roles.

NOTE: To locate the verification and validation evidence, refer to the document table to find the filename, then locate the matching filename in the documents folder.

4 Guide for Reviewing

The following section provides key information that may assist reviewers when evaluating the requirements related to the IRDS project.

Note: All instructions related to how to use the database can be found in the repository [2] under the path, database/README.md.

In the Checklist section of the Requirements Review Protocol (RRP) document, the first table lists the IDs and description of the quality criteria that should be used for reviewing the requirements. The second table specifies which criteria apply to individual requirements and which apply to a group or set of requirements.

4.1 Guide for Reviewing Individual Requirements

When reviewing an individual requirement, refer to Table 2 under the Checklist section in the RRP to identify the quality criteria applicable to single requirements.

For QC-16, use the EARS syntax as defined for this project. The syntax template is shown below.

Requirement Type	Description and Example
Active Requirements	These are always active. General form: `The <system name> shall <system response>` Example: `The control system shall indicate the engine oil quantity to the aircraft.`
Event-Driven Requirements	Initiated only when a triggering event is detected at the system boundary. Designated by the keyword When. General form: `When <optional preconditions> <trigger>, the <system name> shall <system response>` Example: `When continuous ignition is commanded by the aircraft, the control system shall switch on continuous ignition.`
Unwanted Behaviour Requirements	Define the required system response to mitigate an unwanted event or prevent entering an unwanted state. Designated by the keywords `If...Then`. General form: `If <optional preconditions> <trigger>, then the <system name> shall <system response>` Example: `If the computed airspeed fault flag is set, then the control system shall use modelled airspeed.`
State-Driven Requirements	Active while the system is in a defined state. Designated by the keyword `While`. General form: `While <in a specific state>, the <system name> shall <system response>` Example: `While the aircraft is in flight, the control system shall...`

Table 3: EARS template [3].

NOTE: Since the protocol defines what should and should not be done, the protocol module and sub module requirements are primarily active. Reviewers should take care not to misinterpret statements such as "The protocol shall define [X] for when [Y] occurs" as event-driven requirements. The protocol defines the expected responses to circumstances, but it is not itself governed or triggered by external events.

When a requirement satisfies all specified criteria, its verification status should be set to "Accepted" in the database. If any criteria are not met, complete Table 3 in the RRP with the corresponding information in the relevant columns.

4.2 Guide for Reviewing a Group of Requirements

When reviewing multiple requirements, confirm bidirectional traceability between higher- and lower-level requirements. Also check for consistency, redundancy, modularity, and clear structure. Functions have been implemented to facilitate these checks, and the following section explains how to use them.

4.2.1 Checking for Traceability

To check for traceability, use the plot tree function — especially when performing downward traceability analysis. This can be done by selecting the corresponding option number from the menu.

You will be prompted to enter the requirement you wish to trace. After providing the necessary details, a binary-style tree will be displayed, showing the IDs of all related (child) requirements.

You can also choose to plot all requirements to identify missing links. For example, if the database contains requirements MSG-01 through MSG-05, the complete diagram should show all of them. If for example MSG-04 is missing from the tree, it indicates a traceability gap.

All goals originate from the KAOS model, which is referenced in the traceability matrix. The KAOS model itself is derived from the project's Project Objectives.

4.2.2 Checking for Consistency

Consistency and redundancy checks can be performed using the Effect column, which indicates how each requirement influences the overall behaviour or result of the swarm. The following are the possible effects;

- Task assignment
- Task re-assignment
- Consensus
- Search mechanism
- Capability assessment
- Synchronisation
- Fault tolerance
- Independent behaviour

Filtering by a specific effect allows reviewers to examine all related requirements and detect potential conflicts or duplications within that effect category.

4.2.3 Checking for Completeness

The goals (G-01 to G-05) express what the system must achieve rather than the operational actions performed by individual agents. Since they don't represent discrete agent behaviours, they cannot be decomposed into behavioural workflows in a meaningful or testable way.

Behavioural modelling begins at the Drone Swarm Requirement level, where functional operations can be explicitly described as agent actions, message exchanges, decision steps, and protocol behaviours. Therefore, traceability from Goals $\rightarrow$ Drone Swarm Requirements is maintained through requirements derivation, while behavioural traceability begins at the level where concrete behaviour exists.

The system requirements were derived directly from the drone-swarm behavioural workflows demonstrated in document titled "Systemreq completeness evidence" with document_id D-02.

Reviewers must verify that the behavioural workflow adequately addresses all key aspects needed to fulfil the final system functions. They must also ensure that every behavioural step has been correctly mapped to a corresponding requirement within the protocol module.

NOTE: Behavioural diagrams were developed only for the transition from swarm-level behaviour to the system level. This decision was intentional, as the primary focus of the project was the definition and analysis of system requirements. Producing full behavioural models for all levels would have required a complete verification and validation process, which was beyond the available time and team capacity. Therefore, diagrams were limited to the level necessary to ensure traceability and correctness of the system requirements.

References

[1] R.-I. sa, *A KAOS Tutorial*, https://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf, Respect-IT sa, Belgium, 2007, tutorial, October 18, 2007. [Online]. Available: https://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf

[2] A. Haglund, C. Namatovu, E. Målqvist, E. Zainali, and Y. M. Beyene, "MDU-C2/Intelligent-Drone-Swarm," github.com, Accessed: Nov. 3, 2025. [Online]. Available: https://github.com/MDU-C2/Intelligent-Drone-Swarm

[3] A. Mavin and P. Wilkinson, "Big ears (the return of "easy approach to requirements engineering")," in *2010 18th IEEE International Requirements Engineering Conference*, 2010, pp. 277–282.