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

Project Course in Dependable Systems
22.5 credits

Requirements Management Plan

Responsible
Claire Namatovu
cnu21001@student.mdu.se

Contributors
Andrea Haglund
ahd20002@student.mdu.se

Examiner: Luciana Provenzano

December 1, 2025

DOCUMENT APPROVAL

Name	Role	Version	Date	Signature
Andrea Haglund	Chief Engineer	1.2	2025-11-30	
Yonatan Michael Beyene	Q&C Manager	1.2	2025-11-30	

DOCUMENT CHANGE RECORD

Version	Date	Reason for Change	Pages / Sections Affected
0.1	2025-10-02	Version for review	
1.0	2025-10-03	Version for public release	All
1.1	2025-11-20	Addressing comments	All
1.2	2025-11-26	Aligning plan and execution	15

Contents

Glossary

CE
>   Chief Engineer. 15

EARS
>   Easy Approach to Requirements Syntax. 6

ERD
>   Entity Relationship Diagram. 17

FHA
>   Functional Hazard Assessment. 9

IDRS
>   Intelligent Replanning Drone Swarm. 4, 5

RM
>   Requirements Manager. 4, 6, 8, 15

SM
>   Safety Manager. 6

SysML
>   Systems Modelling Language. 8

UML
>   Unified Modelling Language. 6

VVM
>   Validation & Verification Manager. 6, 7

1 Introduction

This document presents a detailed plan on how the requirements will be elicited, analysed, documented, traced and maintained throughout the Intelligent Replanning Drone Swarm (IDRS) project life cycle. The Requirements Manager (RM) is responsible for overseeing all requirements-related activities and shall ensure that each activity is carried out in accordance with the procedures outlined in the plan. Other managers with other roles will also be expected to participate in requirements-related activities. Their specific responsibilities will be outlined in this document as well.

1.1 Purpose

This plan is intended to guide all project managers involved in requirements-related activities, on how to handle the requirements of this project. Additionally, the document also aims to build confidence of the external stakeholders such as the project owner and course responsible, in the system's development approach.

1.2 Related Documents

Below are the documents that are referenced or utilised in this management plan.

Document ID	Document Title
PP-01	Project Plan
CM-01	Configuration Management Plan
RM-04	Requirements Review Protocol
SM-01	Safety Management Plan
VV-01	Validation & Verification Management Plan
ISO/IEC/IEEE 29148:2018	Systems and software engineering Life cycle processes — Requirements engineering

Table 1: List of related documents

2 Scope

This plan will apply to all requirements associated with the IDRS project throughout the entire development lifecycle.

Additionally, the intended depth of requirement definition, particularly for subsystem requirements, will focus only on the most important functionality needed to realize the intended system functions. This approach ensures that the requirements remain realistic given the project's time constraints, reducing the amount of review and verification effort that might otherwise prevent completion before the deadline.

2.1 Objectives

Based on the ISO 29148 standard, the following objectives have been established to guide the requirements management process:

- To ensure a complete and accurate understanding of stakeholder needs, and to reflect those needs clearly and consistently within the documented requirements.
- To ensure the readability and understandability of the requirements, enabling developers and testers to interpret them accurately.
- To ensure maximum possible traceability of all requirements for example 100% traceability between requirements.
- To ensure compliance with the standard for creating high-quality requirements.
- To develop a clear and structured process for managing changes to requirements.

2.2 Deliverables

Below is a table including the expected deliverables from the requirements management.

Deliverable ID	Deliverable Title
RM-02	Requirements Specification including traceability matrix in both binary tree form and table form
RM-03	Requirements Specification Guide

Table 2: List of expected deliverables

3 Tools & Techniques

This section describes the anticipated tools that will be used to work with the requirements and also includes motivation for the choice of each tool.

3.1 Modelio

For the elicitation phase, Modelio will be used. It is a modelling tool that supports various Unified Modelling Language (UML) diagrams and requirements engineering features [1]. Modelio will help in capturing and refining stakeholder requirements in a structured and visual format (see section 4.1 Elicitation). Its support for use case diagrams, class diagrams, and requirement diagrams will help document functional and non-functional requirements clearly. Additionally, this will ensure that all requirements are properly captured and aligned with system design from the outset.

3.2 SQLite

As the requirements specification is expected to grow in size and complexity, selecting a tool that supports traceability will be prioritised [2]. While Excel may initially seem like a viable option due to its accessibility, it will likely require extensive manual effort to check for issues such as duplicate IDs and maintain consistent linkages between requirements. Additionally, generating visual representations of requirement relationships in Excel will be challenging and time-consuming.

To address these limitations, SQLite in Python, which is readily available, shall be utilised. SQL's built-in constraints, such as UNIQUE and FOREIGN KEY, will allow the enforcement of data integrity and will also automatically manage connections between requirements. This approach will significantly reduce manual work and improve traceability. Furthermore, it will enable the generation of visualisations of requirement relationships, making the overall process more efficient.

3.3 Python (Visual Studio Code)

Python (scripts written in Visual Studio Code) is a highly versatile programming language that will allow integration of information from various file types and data sources [3]. In this project, the need to visualise data stored in a database is anticipated. To achieve this, SQLite in Python, will be combined with a visualisation tool such as the Matplotlib Python library. This combination will enable efficient storing, querying, and representation of the relationships between requirements in a clear and intuitive format, supporting both traceability and analysis.

3.4 Matplotlib

A specific Python library, i.e. Matplotlib, will be used to generate visual representations of the linkage between requirements. Figure 1 is an example of this visual representation.

3.5 Git and Github

For storage and version control of the database code, Git and Github will be utilised. The specific procedure for using this tool will be described in the configuration management plan [4] and in the project plan [5].

3.6 Responsibilities and Management Process

The RM is responsible for setting up the database and implementing the identified constraints in the code. Additionally, the RM manages the entry and maintenance of all requirements data within the database. In cases where the Safety Manager (SM) inputs safety-related requirements, the RM must review and verify these entries to ensure they comply with defined standards, such as the Easy Approach to Requirements Syntax (EARS) methodology.

Once the requirements are entered into the database, the Validation & Verification Manager (VVM) is responsible for reviewing them and ensuring that each requirement is both validated and verified. This involves examining the origin and rationale behind each requirement to confirm its necessity and alignment with project goals. To simplify this process, the `plot tree` function can be used to visualize the links
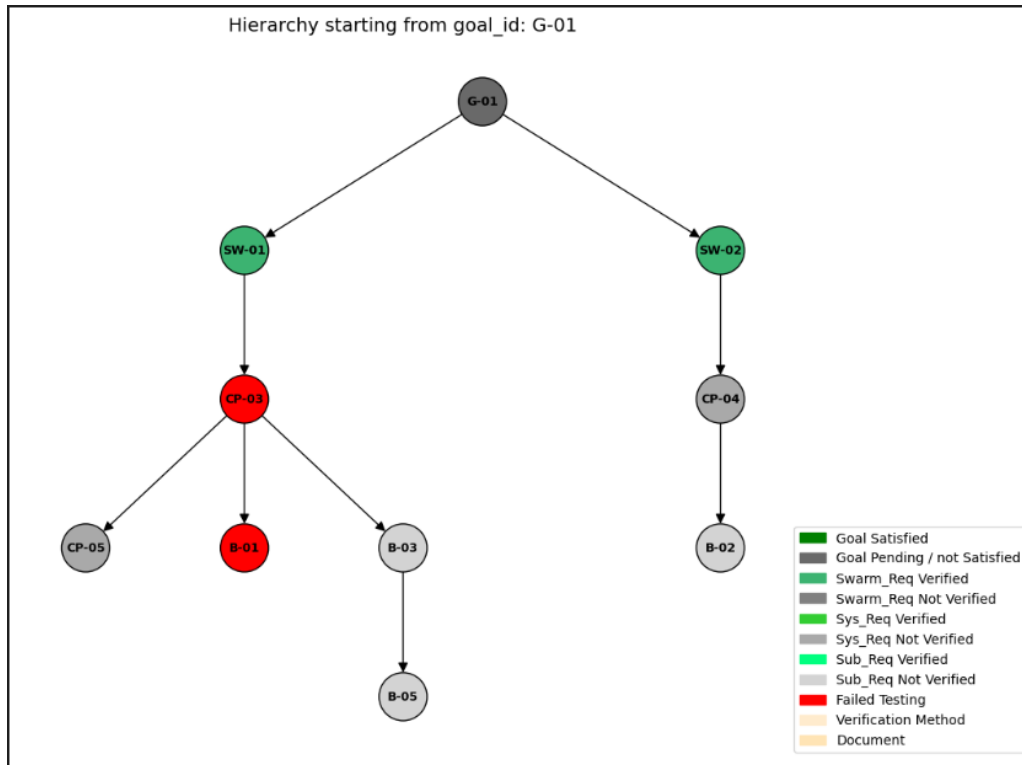
Figure 1: This is a sample visualisation of the linkage between requirements.

between requirements.

Note: The generated image only displays the IDs of the requirements. To view the full requirement text, one **must** refer back to the database. However, the IDs will be designed with meaningful prefixes that often indicate the nature of the requirement. If the meaning is not immediately clear, the prefix and its meaning can be easily located in the Table 0: id_glossary.

Requirements shall also be reviewed in accordance with the established requirements review protocol. In cases where any deviations are identified, the VVM is responsible for documenting them appropriately in the designated table within the provided review document.

Once a requirement has been reviewed, the VVM updates the review status column to reflect the prevailing review status (Accepted or Rejected) and enters their initials in the reviewer column to indicate who performed the review.

If a requirement is rejected during review, the author must either correct the identified issue or provide a documented justification for dismissing the reviewer's comments. This response must be recorded in the designated table within the review document, following the procedure outlined there.

4 Requirements Engineering Process

This chapter outlines the approach used to define and manage system requirements. It covers the methods employed for requirements elicitation, the adopted specification standards and the analysis process.

4.1 Elicitation

The elicitation of requirements for this project will be conducted using a combination of collaborative techniques namely the following:

- Brain storming: The project members shall engage in collaborative discussions meant to identify and agree upon the scope and expectations of the system. The RM shall be responsible for documenting the outcomes of these sessions ensuring that all relevant points are captured and translated into formal requirements.
- Scenario-based discussions: To make the elicitation process more organised, discussions shall be structured around specific scenarios. For instance, if a scenario pertains to safety aspects, the RM will involve only the relevant stakeholders—such as the Chief Engineer and Safety Manager during the analysis of that scenario.
- Modelling: Modelling languages like Systems Modelling Language (SysML) (3.1) will be used to help visualise stakeholder requirements thereby facilitating clarification and early refinement[1].
- Interviews: Following the modelling of the system, interviews will be conducted with key stakeholders to validate the preliminary requirements and gather additional insights. This method ensures that the elicited requirements align with stakeholder expectations and needs.

NOTE: The RM **MUST** partake in **ALL** the elicitation activities.

4.2 Specification Standards

When the requirements have been elicited, the next step is the formal documentation and specification of these requirements.

The requirements' management shall adhere to the principles and guidelines outlined in ISO 29148:2018 – Systems and Software Engineering — Life Cycle Processes — Requirements Engineering [6]. By aligning with the standard, the project ensures that requirements are clearly defined, traceable, consistently documented and maintained. According to the standard it is recommended to use a structured template and unified language when formally writing requirements. Specifically, the EARS (Easy Approach to Requirements Syntax) syntax will be employed. EARS provides a set of structured "boilerplate" templates that guide the phrasing of requirements based on their type (e.g., event-driven, state-driven, optional behaviour) [7].

4.3 Analysis

The process of analysing requirements will be guided by the traceability matrix(see section 6.1 Traceability management) which provides a structured means of navigating through the requirements. This matrix enables users to efficiently locate individual requirements or groups of related requirements based on their origin, dependencies, or their association to a specific component of the system. Once the relevant requirements have been identified, the analysis should proceed in accordance with the review and quality assurance protocol described in section 6.3 Quality assurance).

5 Requirements Organisation & Structure

The requirements and their associated documentation will be organized within a structured SQL database, as outlined in section 3 Tools & Techniques. Each requirement will be represented as a record characterised by a unique identifier, descriptive attributes, and other categories that facilitate traceability and analysis. To accommodate cases where a requirement may be related to multiple goals, subsystems, items etc, junction tables will be used to handle the many-to-many or many-to-one relationships. Unlike in Excel, where multiple values can sometimes be placed in the same cell, SQL relational database design principles recommend that each cell store a single value, hence the need for the junction tables. The database schema consists of 13 interrelated tables, each of which is described in detail in the sections below:

5.1 Tables

For a graphic representation of the tables below, see figure 2 in the appendix.

5.1.1 Table 0: id_glossary

This table is meant to have information on the different prefix used and their meaning. It has 2 columns namely:

prefix: The letter combination of an ID chosen by the author.
meaning: An explanation of what the prefixes means, for example "SG" for "Safety goals".

5.1.2 Table 1: goals

This table will contain the goals obtained from the stakeholder, task description and from the Functional Hazard Assessment (FHA). The FHA table description can be found in the Safety Management Plan [8]. The anticipated columns of the table are the following:

goal_id: Each goal must have a unique ID that is meaningfully connected to the goal itself. For example, safety goals from the FHA could have IDs starting with "FH_G," though this is not strictly required especially if the prefix and meaning are present in the id_glossary table. This approach helps to simplify searching and querying in later phases.
goal description: A clear description of the goal in natural language.
stakeholder: Identifies the primary stakeholder responsible for the goal, enabling traceability by allowing reviewers to reference and verify the goal description directly with the stakeholder.
origin : As some goals will be derived from the FHA or standards, it is important to have an "Origin" column to trace to the corresponding hazard, thereby demonstrating how the goal addresses its source.
priority: This attribute should only have three possible values-"Key", "Mandatory", "Optional". A mandatory goal is one that must be achieved in order for the project to be considered successful whereas a key goal is one that significantly contributes to the project success but may not be strictly required. Optional priority means that the project may still be considered successful even though the goal is not achieved.
rationale: Under this column, the author will, where necessary, explain for example the reason for chosen priority, to ensure clarity and transparency in the decision process.
satisfaction_status: This column will have three possible values-"Pending","Not satisfied" and "Satisfied".
method_id: Reference to the test and verification table, indicating what method has been used to justify the satisfaction of the goal. It may remain empty only when the satisfaction status is marked as "Pending" or "Not satisfied".

5.1.3 Table 2: drone_swarm_requirements

It was determined that the high-level goals must first be broken down to the drone swarm operational level before they can be translated into system-level (replanning protocol) requirements. This hierarchical breakdown will not only facilitate the derivation of more precise, implementable, and verifiable system

requirements, but it also follows the project's V-model (see Project Plan [5]). In this table, there will be the following columns:

swarm_req_id: Contains the ID of the requirement.
requirement : This is where the requirement will be written.
priority: See Priority under section 5.1.2 Table 1: goals
effect: This column describes what the requirement affects as a result. For example, if a requirement says, "The system shall be red", the effect would be appearance. This column was added for quality assurance purposes. See section 6.3 Quality assurance for further description. Project managers must come to a consensus on the terminology to be used in the "effect" column. Once a keyword such as appearance is agreed upon, it must be applied consistently across all documentation. Alternative terms or synonyms (e.g., visuals) should be avoided to maintain uniformity and clarity.
author: The initials of the individual responsible for writing the requirement. To ensure accountability, the column must be non-nullable, and an appropriate database constraint should be enforced to prevent empty entries.
verification_status: This column shows where the requirement currently stands in the review process. There will only be two possible values, "TBR" (To Be Reviewed) and "Accepted"
verifier: contains the initials of the person(s) responsible for the review. This column must also be non-nullable if the review_status value is "Reviewed".
validation_status: Shows where the requirement currently stands in the verification process. There will only be 4 possible values: "Pending","Pass","Failed" and "Inconclusive".
vv_method: References the method used for validation of the requirement. The Verification_method column is only allowed to be NULL if the verification status is pending.

5.1.4 Table 3: goal_children

This will be a junction table for mapping each goal ID to its respective resulting drone swarm requirement(s). Goals and requirements must be written in such a way that one drone swarm requirement can only result from one goal. If this is discovered to not be the case, the goal/ swarm requirement should be re visited and re-written. This table shall have the following columns:

id: An auto-incrementing integer serving as the table's primary key. It uniquely identifies each record and can also be used as a reference point for a specific relationship.
goal_id: Reference to the associated goal.
swarm_req_id: Reference to the drone swarm requirement that has been derived from the associated goal.

5.1.5 Table 4: swarm_req_children

This will be a junction table for mapping each drone swarm requirement to its respective resulting system requirements. The table shall have the following columns:

id: See ID under section 5.1.4 Table 3: goal_children.
swarm_req_id: Reference to the associated drone swarm requirement.
sys_req_id: Reference to the system requirement that has been derived from the associated drone swarm requirement.

5.1.6 Table 5: system_requirements

This table will contain the system requirements obtained from goals, use case diagrams and other. The anticipated attributes of the goals are described below.

parent_id: Reference to the system requirement from which this requirement was derived. System requirements that are derived from goals will have the value "None".

sys_req_id: The unique identifier for the individual requirement. This ID will also preferably have a meaningful connection to the requirement it pertains to. All related requirements will have the same prefix.

requirement: This is where the requirement will be written.

priority: See Priority under section 5.1.2 Table 1: goals

effect: See effect under section 5.1.3 Table 2: drone_swarm_requirements

author: The initials of the individual responsible for writing the requirement. To ensure accountability, the column must be non-nullable, and an appropriate database constraint should be enforced to prevent empty entries.

verification_status: This column shows where the requirement currently stands in the review process. There will only be two possible values, "TBR" and "Reviewed"

verifier: contains the initials of the person(s) responsible for the review. This column must also be non-nullable if the review_status value is "Reviewed".

rationale: This column enables the author to explain the presence/break-down of a requirement that may not be obvious.

validation_status: Shows where the requirement currently stands in the verification process. There will only be 4 possible values: "Pending","Pass","Failed" and "Inconclusive".

vv_method: References the method used for validation of the requirement. The vv_method column is only allowed to be NULL if the verification status is pending.

comment: allows for free discussion, that is to say questions and suggestions aimed at improving the requirements. In cases where the verification status is inconclusive, it also provides space for explanations.

5.1.7 Table 6: sysreq_children

This is a junction table for mapping the system requirements to their respective resulting subsystem requirements. This table shall have the following columns:

id: See ID under section 5.1.4 Table 3: goal_children

sys_req_id: Reference to the associated system requirement.

sub_req_id: Reference to the subsystem requirement that has been derived from the associated system requirement.

5.1.8 Table 7: subsystem_requirements

This table contains the same attributes as those in section 5.1.6 Table 5: system_requirements. The definitions should be interpreted according to the subsystem context.

5.1.9 Table 8: test_and_verification

This table will include the different test and verification methods that will be used to evaluate the requirements and goals and it will have the following columns:

Method_id: The unique method ID.

Description: Detailed explanation of each test and/or verification method, outlining the procedure,tools,conditions and other important information.

Method_type: This column will have 3 possible values: "Inspection", "Analysis" and "Test"

Document_id: Reference to the appropriate document that proves the existence of the method.

5.1.10 Table 9: documents

This table includes all relevant documentation such as safety analysis documents, simulation results etc. The table will have the following columns:

doc_id: The unique document ID.

title: The title of the document.

description: A summary of the content of the document.
file: Stores the URL, file path, or binary reference to the actual document (any relevant artifacts) associated with the record. This enables retrieval and linkage of documentation within the database.
version: Specifies the version of the linked document.
author: The name of the author of the document.

5.1.11 Table 10: V_join_Documents

This is a junction table designed to map each verification method to its corresponding document. Since a single verification method may be associated with multiple documents, this table was introduced to properly represent and manage those relationships. The table shall have the following columns:

method_id: Reference the test/verification method.
doc_id: Reference the corresponding document.

5.2 Database Usage Workflow

The workflow described here covers key stages for data entry into the database. It is intended to help streamline the tasks of those who will need to interact with the database and consequently reducing errors and maintaining consistency. The following is the recommended procedure for data entry.

1) **Record the Goals**
   Begin by documenting the project goals in the goals table and assign each goal an ID with a meaningful prefix for example, "SG" for Safety Goal. Go to the id_glossary table and enter the prefix and its meaning under the respective column. The recommended format of the IDs is "AB-01". Thereafter, fill in the appropriate information under each column.

2) **Recording Drone Swarm Requirements**
   Record the drone swarm requirements and fill in the relevant columns while noting their origin i.e. the goals from which it is derived. Populate the goal_children table with the noted relationships, that is to say, the goal_ids and respective swarm_req_ids.

3) **Record System Requirements**
Based on the finalised drone swarm requirements, system requirements will be formulated. For each system requirement, identify and record the origin. This could either be drone swarm requirement(s) or another system requirement. For the system requirements coming from a drone swarm requirement, the IDs should be recorded into the swarm_req_children table and for those coming from other system requirements, the ID of the parent requirement should be written in the parent_id column. Ensure that all child requirements are accurately linked.

4) **Fill the Item table**
Before starting on the subsystem requirements, the primary or relevant component names and their ids must be written into the item table.

5) **Fill the subsys_join_item table**
Before entering the subsystem requirements into the respective table, first record the subsystem requirement ID and its associated item ID into the subsys_join_item table.

6) **Record subsystem requirements**
Document the subsystem requirements while also noting the origin of each requirement. Similarly to the process of recording system requirements, the system requirements from which a subsystem requirement has been derived should be recorded in the sys_req_children table and those from other subsystem requirements should have the ID of the parent ID in the parent_id column. Each requirement should belong to at least one item and this relationship should be recorded in the subsys_join_item table as outlined in step 5.

7) **Record the test and verification**
Each test or verification method should be recorded in the test_verification_table and should reference the relevant document in the V_join_documents table, as proof of the method.

8) **Record in the V_join_documents table**
Similar to all the other junction tables, all the methods and their related documents should be recorded in this table, noting the method ID and associated documents ID.

6 Activities

6.1 Traceability management

This chapter presents the traceability framework adopted for the requirements of the project, outlining how users can navigate the database to locate various requirements and understand the relationships between them, such as which requirements are derived from some and which ones lead to others or which have and haven't been verified or tested.

6.1.1 Searching for requirements

To retrieve a specific requirement, one can look through the id_glossary table to find the appropriate prefix associated with the requirement category of interest. By applying this prefix as a filter in the appropriate table, all requirements whose identifiers begin with that particular prefix can be obtained. Similarly, when searching for requirements related to a particular component, one can look through the item table to obtain the corresponding item_id. This identifier can then be used in the subsys_join_item table to determine the individual or set of subsystem requirement IDs associated with that item.

6.1.2 Searching for the linkage between requirements

Since the SQL code is integrated with Python, additional scripts can be developed to support traceability. For example, by leveraging libraries such as matplotlib and networkx, it is possible to get tree-based diagrams that illustrate the hierarchical structure of requirements and their associated children. These visualizations provide a more intuitive way to trace how high-level goals propagate into drone swarm requirements, system requirements, and eventually subsystem requirements. Alternatively, linkage analysis can also be performed manually by querying the relevant junction tables (e.g., goal_children, swarm_req_children, and subsys_join_item) to identify the parent–child mappings between entities. Manual querying of junction tables is a more convenient way to trace upward relationships (from child to parent) in large and complex trees. SQLite extension in Visual Studio Code demonstrates these relationships in tabular form, which can then be cross-referenced to validate completeness.

Beyond visualisation, traceability can be further strengthened by implementing automated checks. For instance, Python scripts could be used to highlight requirements that lack parent or child relationships. Such checks ensure that gaps in requirement coverage are promptly identified, thereby supporting verification, validation, and overall quality assurance of the system design.

Furthermore, requirements will be linked to their corresponding verification methods in the test_and_verification table. The tree graph may be utilised by responsible parties to ensure that all requirements have been properly verified.

6.2 Change management

The aim of the plan is to minimise the need for changes to any requirement. To achieve this, specific procedures will be implemented and the suggested constraints (section 5 Requirements Organisation & Structure) added to the database to facilitate stability and consistency of all requirements.

Requirements will be reviewed in accordance to the requirements review protocol, and validated before they can lead to any other lower-level requirements [9]. This way we minimise a case where the error propagates so far down. According to the ISO 29148:2018 standard, we must acknowledge that change is inevitable, and this case we will assume worst case scenario [6]. The worst case scenario being that change needs to be done to a high level requirement that has multiple lower-level requirements who also have other lower-level requirements. The following steps should be taken;

1) Run the plot_tree code for visualisation.
2) When prompted, enter the ID of the requirement that is intended to be changed. The tree will then be displayed with the selected requirement as the root node, allowing for a clear view of all related and dependent requirements.
3) Determine how tightly coupled the requirement is with others using the obtained tree.

4) Assess the impact of change and evaluate the scope by answering the following questions:
   - Will it affect other requirements?
   - Will it require change in the preliminary system design?
   - Will it invalidate existing test cases?
   - Will it affect compliance with safety?
5) Document the analysis including the nature and extent of the impact, all affected artifacts, etc.
6) Present the analysis to relevant stakeholders and team members and ask for approval from the Chief Engineer (CE).
7) If the change is approved, a plan for implementing the change will be developed and documented. Documentation will only be required for more complex changes, where detailed planning is essential.
8) Ask for approval from the CE and proceed with the change if approved.

6.3 Quality assurance

This section presents the steps that will be taken to ensure the quality of requirements both individually and as a set. The benchmark for quality is based on the criteria outlined in section 5.2.5 and 5.2.6 of the ISO 29148 standard [6].

6.3.1 Quality assurance for individual requirements

To support an unbiased review process, certain constraints will be implemented in the Python scripts. Specifically, the system will verify that the author's initials and the reviewer's initials are different. If they match, an error will be triggered. This ensures that no one reviews their own requirement, as individuals may overlook their own mistakes.

Each requirement shall be traceable to only one higher-level requirement. However, requirements can have multiple lower-level requirements. This not only promotes traceability, but also simplifies the assessment of the necessity of each requirement, a key quality criteria for a requirement. Additional constraints like the foreign key constraints and NOT NULL-conditions will ensure that every requirement has an origin.

While automated checks can catch many issues, some may only be identified during the manual review process. During this process, the reviewer reads the requirement and evaluates it against the criteria outlined in the review protocol (see RM-04). If a requirement fails to meet any criterion, the reviewer must document the issue in the review table provided in the protocol. Guidance on how to fill in the table for specific cases is included in the review protocol. If further clarification is needed, the RM should be consulted. If the requirement requires change then the steps outlined in section 6.2 Change management should be followed.

6.3.2 Quality assurance for a set of requirements

To ensure consistency across the set of requirements, an "Effect" column will be included in each requirements table (See example section 5.1.3 Table 2: drone_swarm_requirements). This column can be filtered to group requirements with the same effect, allowing for targeted analysis. For example, if multiple requirements are categorised under the effect "Appearance", the person analysing can verify that they do not contradict each other — such as one requirement stating "The system shall be red" and another stating "The system shall be yellow".

The traceability matrix will help in ensuring the completeness of a set of requirements. A set of requirements is considered complete when all necessary requirements have been identified to fully describe the system's functionality as well as constraints, and when each requirement can be traced from its origin (goals) through to its verification or validation evidence. Completeness may be demonstrated by mapping each behavioural step is to at least one requirement, ensuring functional completeness.

Additionally, the requirements shall also be reviewed as a set where aspects like modularity and structure will be targeted under the analysis and review.

References

[1] Modeliosoft, *Modelio*, 2023, version 4.2.1. [Online]. Available: https://www.modelio.org

[2] SQLite Development Team, *SQLite*, 2023, version 3.43.0. [Online]. Available: https://www.sqlite.org

[3] Microsoft Corporation, *Visual Studio Code*, 2023, version 1.83. [Online]. Available: https://code.visualstudio.com

[4] Y. M. Beyene, *Configuration Management Plan*, Intelligent Replanning Drone Swarm, Sep. 30 2025, Version 1.0.

[5] A. Haglund, *Project Plan*, Intelligent Replanning Drone Swarm, Sep. 30 2025, Version 1.0.

[6] "Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering," *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, 2018.

[7] 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.

[8] E. Målqvist, *Safety Management Plan*, Intelligent Replanning Drone Swarm, Sep. 30 2025, Version 1.0.

[9] E. Zainali, *Validation Verification Management Plan*, Intelligent Replanning Drone Swarm, Sep. 30 2025, Version 1.0.

Appendix