LOGO

Organisation [Name]

Department [Name]

<Project Charter>

Coreqt – RM tool

Date: <Date>

Doc. Version: 1

Template version: 3.0.1

This artefact template is aligned with the PM² Guide V3.0

For the latest version of the templates visit:
https://www.pm2alliance.eu/publications

The PM² Alliance is committed to the improvement of the PM² Methodology and of its supporting artefact. Project management best practices and community contributions & corrections are incorporated in the PM² Alliance's artefact templates.

Join the PM² Alliance and visit the PM² Alliance GitHub to provide your feedback & contribution:
https://github.com/pm2alliance

Document Control Information

Settings	Value
Document Title:	Project Charter
Project Title:	Coreqt – RM tool
Document Author:	Mehmet Can Ünal
Project Owner:	Mehmet Can Ünal
Solution Provider	Compas Solutions
Project Manager:	Mehmet Can Ünal
Doc. Version:	1
Sensitivity:	High
Date:	02/12/2026

Document Approver(s) and Reviewer(s):

NOTE: All Approvers are required. Records of each approver must be maintained.
All reviewers in the list are considered required unless explicitly listed as Optional.

Name	Role	Action	Date
		<Approve / Review>	

Document history:

The Document Author is authorized to make the following types of changes to the document without requiring that the document be re-approved:

- Editorial, formatting, and spelling
- Clarification

To request a change to this document, contact the Document Author or Project Owner.
Changes to this document are summarized in the following table in reverse chronological order (latest version first).

Revision	Date	Created by	Short Description of Changes

Configuration Management: Document Location

The latest version of this controlled document is stored in <location>.

<These notes should be deleted in the final version :>

Notes for Templates:

- Text in <orange>: has to be defined.
- *Text in <blue>:* guidelines and how to use the Template. Should be deleted in the final version.
- Text in green: can be customised. Should be recolored to black in the final version.

TABLE OF CONTENTS

1 Executive Summary **4**
2 Considerations on the Business Case **4**
3 Project Description **4**
3.1 Scope 5
3.1.1 Includes ("IN" Scope) 5
3.1.1.1 Functional Deliverables 5
3.1.2 Excludes ("OUT" Scope) 6
3.1.3 Scope Statement 6
3.2 Success Criteria 6
3.3 Stakeholder and User Needs 7
3.4 Deliverables 7
3.5 Features 9
3.6 Constraints 10
3.7 Assumptions 10
3.8 Risks 11
4 Cost, Timing and Resources **13**
4.1 Cost 13
4.2 Timing and Milestones 14
4.3 Planned Resources 14
5 Approach **14**
5.1 Methodology 15
5.2 Change Management 15
5.2.1 Project Change 15
5.2.2 Configuration Management 15
6 Governance and Stakeholders **16**
6.1 Structure 16
6.2 Roles and Responsibilities 16
6.3 Other Stakeholders 16

1 Executive Summary

The Compas Coreqt RM Plugin for capella shall provide MBSE users with budget constraints and uncertainties an option to add main features of RM tools into the Capella MBSE architecture. This plugin shall instantiate the first of Compas software development programs which aim in total to an own and independent eco-system. With limited capabilities and resources, the team will thus focus on main functionalities, robustness and cost effectiveness. Expensive features shall remain out of scope, namely integration with other tools outside Capella (e.g., Jira, GIT). The main focus shall thus be to connect to the capella architecture.

2 Considerations on the Business Case

Compas solutions goal of becoming provider of SE solutions with focus on digital engineering and seamless product development, now there is the chance of supplying client with specific tools needed for software development with regard of safety-criticality and certification needs. The focus of this project relies solely on the development of a RM tool to support requirement-based software development. Compas aims to bridge the requirements intense software development operations with Capella MBSE tool to utilize the MBSE leverage and the open-source options in order to create a platform around a systems engineering structure. The current opportunity is that BAYKAR is about to set up its own requirement-based software development framework to certify its UAVs. They desired domestic products more then foreign and our position as a coworker for a certain time together with the embedding of Capella for requirements capture allows us to support them in their endeavour.

3 Project Description

3.1 Scope

3.1.1 Includes ("IN" Scope)

The project includes the design, development, and delivery of a Capella plugin that extends native requirements management capabilities to support compliance-ready, model-centric traceability and configuration control for systems engineering environments.

Specifically, the project will deliver:

3.1.1.1 Functional Deliverables

1. **Bidirectional Traceability Engine**
    - Support for requirement-to-model linking
    - Configurable link types (derives from, satisfies, verifies, impacts, etc.)
    - Multi-level upstream and downstream navigation
2. **Suspect Link Detection**
    - Automatic flagging of impacted links upon requirement or model changes
    - Manual review and clearance mechanism
3. **Traceability Matrix View**
    - Configurable artifact type selection
    - Gap analysis (missing parent, missing verification)
    - Export capability (PDF/CSV)
4. **Impact Analysis Module**
    - Visual dependency graph
    - Change propagation analysis
    - Interactive navigation
5. **Baseline Management**
    - Creation of named baselines at milestones (e.g., SRR, PDR, CDR)
    - Baseline comparison view
    - Read-only baseline storage
6. **Version History**
    - Field-level change tracking
    - Full version history per requirement
    - Version comparison view
7. **Review & Approval Workflow**
    - Configurable workflow states
    - Role-based approval gates
    - Online review sessions
    - Commenting functionality
    - Electronic signature support
    - Audit trail of review decisions
8. **Audit Trail**
    - Immutable logging of changes

- Traceability of approvals and workflow transitions
- Exportable audit reports

9. **Requirement ↔ Test Linking**
   - Support for linking test artifacts to requirements
   - Coverage metrics
   - Identification of unverified requirements
10. **User Access Control**

- Role-based access management within the plugin scope

3.1.2 Excludes ("OUT" Scope)

The following items are explicitly excluded from the scope of this project:

1. Development of a standalone Requirements Management system independent of Capella.
2. Full Application Lifecycle Management (ALM) capabilities (e.g., CI/CD pipelines, defect tracking, DevOps management).
3. Source code repository hosting (Git server functionality).
4. Enterprise-wide identity management systems (SSO, LDAP integration beyond basic authentication).
5. Integration with external RM tools (e.g., Codebeamer, DOORS, Polarion) in Phase 1.
6. Automated test execution frameworks.
7. Regulatory certification of the software tool itself (e.g., DO-330 tool qualification).
8. Product line engineering beyond branch-level variant comparison.
9. Customization for specific customers beyond generic configuration capabilities.
10. Cloud-based SaaS infrastructure (initial release assumed on-premise/plugin distribution).

These exclusions clarify that the project focuses specifically on enhancing MBSE-native requirements management within Capella rather than delivering a full enterprise ALM ecosystem.

3.1.3 Scope Statement

This project will develop and deliver a Capella-integrated Requirements Management plugin that provides compliance-ready traceability, baselining, review workflows, impact analysis, and coverage management to enhance model-centric systems engineering processes.

3.2 Success Criteria

- Delivery of a fully functional Capella plugin implementing the defined in-scope capabilities (traceability, baseline management, version history, workflow & approvals, audit trail, and coverage management).
- Successful integration with the current stable version of Capella without degradation of core modeling performance.
- Stable version control and baseline mechanism verified through system testing.
- Completion of system testing with:
  - Zero critical defects
  - No more than five (5) major defects open at release
- Delivery within:
  - 3 months ±10% schedule tolerance
  - Approved budget ±10% cost tolerance

3.3 Stakeholder and User Needs

ID	Need Description	Priority
1	End-to-end traceability	Critical
2	Baselines	Critical
3	Workflow & approvals	Critical
4	Impact analysis	High
5	Suspect link detection	High
6	Version history	High
7	Test coverage view	Medium–High
8	Variant handling	Medium
9	Automation of reporting	Medium

3.4 Deliverables

ID	Deliverable Name	Deliverable Description
1	Capella Requirements Management Plugin (Core Software Product)	A fully functional Capella plugin providing advanced requirements management capabilities integrated into the MBSE environment.
1.1	Bidirectional traceability engine	
1.2	Suspect link detection mechanism	
1.3	Traceability matrix module with gap analysis	
1.4	Impact analysis visualization module	
1.5	Baseline management capability	
1.6	Version history with field-level tracking	
1.7	Workflow & role-based approval engine	
1.8	Electronic signature support	
1.9	Audit trail logging module	
2	System Architecture & Technical Documentation	Comprehensive technical documentation describing the internal architecture and integration mechanisms of the plugin
2.1	System architecture description	
2.2	Data model specification	
2.3	Interface/API documentation	
2.4	Security and access control design	
2.5	Traceability data model documentation	

3	User Documentation & Installation Package	Complete user-facing documentation enabling installation, configuration, and operational use.
3.1	User manual	
3.2	Installation guide	
3.3	Configuration guide	
3.4	Administrator guide	
3.5	Quick-start tutorial	
4	Test & Validation Report	Formal verification and validation documentation demonstrating that the plugin meets defined functional and performance requirements.
4.1	Test plan	
4.2	Test cases and results	
4.3	Regression test summary	
4.4	Performance test report	
4.5	Defect resolution log	
5	Demonstration & Pilot Deployment	Deployment of the plugin in at least one pilot environment to demonstrate operational capability within a real MBSE project.
5.1	Installed and configured pilot instance	
5.2	Demonstration session for stakeholders	
5.3	Pilot evaluation summary report	
6	Compliance Reporting Templates	Pre-configured reporting templates supporting regulated environments.
6.1	Traceability matrix export template	
6.2	Audit trail export template	
6.3	Coverage report template	
6.4	Baseline comparison report template	
7	Training Session & Training Materials	Structured training provided to designated users within the Project Owner's organisation.
7.1	Training presentation material	
7.2	Recorded training session (if applicable)	
7.3	Example demonstration project model	

3.5 Features

<This section should define the expected features of the outputs that will be delivered to the users and to the project owner organisation. Think of features as the high-level capabilities associated to the outputs that are necessary to deliver the expected benefits to the users. At this moment of the project, keep

feature descriptions at a high level and focus on capabilities needed and why (not how) they should be implemented. These features will be expanded in greater detail later in the project as it becomes necessary to detail how these features will be implemented by the project core team.

To structure the way the features are identified and described, align them with the previously defined stakeholders and user needs. Keep in mind that one need may be answered by implementing several features.

Because this document is reviewed and read by a wide variety of stakeholders, the level of detail should remain general enough for everyone to understand it. However, enough detail must be available to provide the next stages of the project with the information needed to detail how the outputs will respond to the stakeholders and user needs.

Note: An example of a feature for an issue tracking system (IT System) might be the ability to provide a specific type of report. As the use-case model takes shape, it is recommended that you update their descriptions to refer to the use cases that detail them.>

Function ID	Function Name	Description	Related Need	Deliverable
F-01	Requirement Retrieval	Access and read requirement objects stored using the Requirements VP add-on.	1	1
F-02	Requirement–Model Linking	Create and maintain trace links between requirements and model elements using existing Capella requirement mechanisms.	1	1.1
F-03	Traceability Matrix Generation	Generate matrix views showing relationships between requirements, architecture elements, and tests.	1, 4	1.3, 6.1
F-04	Gap Detection	Identify requirements without verification, without parent, or without allocation.	1, 7	1.3
F-05	Impact Analysis	Identify downstream model elements affected by requirement change.	4	1.4
F-06	Suspect Link Identification	Detect links potentially invalid after requirement or model modification.	5	1.2
F-07	Requirement Version Access	Access historical versions provided through plugin mechanisms (without modifying Requirements VP internal storage).	6	1.6
F-08	Baseline Snapshot Creation	Create read-only baseline snapshots referencing requirements and links.	2	1.5
F-09	Coverage View	Display requirement verification and test coverage status.	7	1.10
F-10	Export of Traceability Data	Export matrices and reports (CSV/PDF).	9	6.1, 6.3

F-11	Audit Logging	Log changes to trace links, baselines, and approvals.	3	1.9
F-12	User Role Enforcement	Restrict editing and approval functions based on roles.	3	1.7

<Example:

- *Need: consumers face problems in selecting the best and cheapest energy company that provides electricity and gas for homes.*
- *Deliverable: a website to inform customers with up-to-date information on energy providers, prices, and terms & conditions.*
- *Feature: website should allow the consumer to enter the most relevant selection criteria to select the best fitting energy provider in the market.*

Note: Features and needs are examples of Requirements. For advanced Requirement Management, you can refer to the PM² Requirement Management Plan and/or use the Requirement Traceability Matrix artefact.>

Note:

The plugin does not implement a standalone requirements repository.

All requirement objects remain managed by the Requirements VP add-on of Capella.

The plugin extends functionality by providing traceability analysis, reporting, baselines, and workflow capabilities while preserving compatibility with native requirement structures.

3.6 Constraints

Constraint Area	Limitation
Budget	€2,000 maximum

Resources	3 persons × 6h/week
Schedule	3 months
Technology	Capella/Eclipse plugin architecture
Coding	Heavy dependency on single main developer
Deployment	Plugin only (no standalone RM tool)
Compliance	Must support audit-ready traceability

3.7 Assumptions

It is assumed that required libraries to code in the capella and eclipse environment are available. Also,it becomes assumed that the AI tool antigravity is capable of programming in the eclipse/capella style. Addtionally, it has been assumed that plugins can be provided by anyone. Licensing is also an option which has assumed to be true.

3.8 Risks

ID	Risk Description & Details	Status	Likelihood	Impa	Risk	Risk Owner	Risk Response Strategy	Action Details

				ct	Level			
R1	Unstable plugin behavior corrupts Capella model data (broken references, overwritten attributes, inconsistent trace links).	Open	Medium	High	High	Lead Developer (Antigravity)	Mitigation / Prevention	Implement backup before write operations; use transactional EMF operations; perform regression testing; validate on duplicate models; add confirmation prompts for destructive actions.
R2	Critical items (version history, baselines, control data) become deleted or corrupted due to malfunction or user action.	Open	Medium	High	Critical	Technical Lead	Mitigation / Avoidance	Make baselines immutable; implement soft-delete mechanism; restrict deletion to admin role; add consistency checks; enable export backup; log all deletion/restore actions in audit trail.
R3	Single-point-of-failure due to heavy coding dependency on one primary developer.	Open	High	High	Critical	Project Manager	Mitigation	Maintain architecture documentation; enforce modular code structure; conduct weekly knowledge sharing; use shared version control repository; avoid undocumented logic.
R4	Scope exceeds feasible delivery within 216 total estimated hours.	Open	High	Medium–High	High	Project Manager	Mitigation	Prioritize MVP features; defer non-critical features; implement incremental releases; freeze scope after Month 1 unless critical changes required.
R5	Performance degradation with large models (>5,000 requirements), causing slow trace or impact analysis.	Open	Medium	Medium	Medium	Technical Lead	Mitigation	Optimize traversal algorithms; implement lazy loading; early performance testing; avoid unnecessary full model scans.
R6	Incompatibility with future or updated Capella versions.	Open	Medium	Medi	Medi	Technical Lead	Mitigation	Develop against stable Capella version; avoid deep internal API dependency; document integration points;

				um	um			validate against minor updates.
R7	Insufficient validation/testing due to limited resources and no dedicated QA.	Open	Medium	High	High	Project Manager	Mitigation	Define minimum mandatory test coverage; allocate fixed weekly testing time; prioritize critical feature testing; prepare structured test cases.
R8	Low user adoption due to usability issues or complex workflow enforcement.	Open	Medium	Medium	Medium	Product Owner	Mitigation	Involve pilot users early; follow Capella UI conventions; limit mandatory constraints in initial release; collect structured feedback.
R9	Budget overrun due to underestimation of effort or technical complexity.	Open	Medium	Medium	Medium	Project Manager	Mitigation	Track effort weekly; avoid non-essential purchases; reassess scope mid-project; maintain 10% contingency margin.

4 Cost, Timing and Resources

4.1 Cost

Budget Line	Quantity / Effort	Unit Cost	Amount (€)
Team Effort (3 persons)	216 h	20 €/h	4,320 €
AI Coding License (3 months)	1 license	300 €/month	900 €
Google Drive Subscription (3 months)	1 subscription	12 €/month	36 €
Total Economic Project Cost			5,256 €

4.2 Timing and Milestones

ID	Milestone Description	Target Delivery Date
M1	Project Kick-off & Initiation Phase Completed	01 Mar 2026
M2	Project Charter & Initial Risk Register Approved	07 Mar 2026
M3	Planning Phase Completed – Project Work Plan & Technical Concept Finalised	14 Mar 2026
M4	AI Environment & Google Drive Structure Configured	18 Mar 2026
M5	Prototype Version (Core Functionality Implemented)	12 Apr 2026
M6	Mid-Project Review & Scope Validation	20 Apr 2026
M7	Feature Complete Version (All Planned Functionalities Implemented)	17 May 2026
M8	Internal Validation & Stability Testing Completed	28 May 2026
M9	Final Release Candidate Delivered	05 Jun 2026
M10	Documentation & User Guidance Finalised	10 Jun 2026
M11	Closing Phase – Final Review & Acceptance	15 Jun 2026
M12	Project Closure Report & Lessons Learned Documented	20 Jun 2026

4.3 Planned Resources

ID	Resource Requirement	Description
RR1	Project Lead	Responsible for planning, coordination, reporting and governance
RR2	Technical Lead	Responsible for system architecture, integration and AI supervision
RR3	Validation Support	Responsible for testing, model stability checks and verification
RR4	Antigravity AI License	AI coding support for development phase
RR5	Google Drive Subscription	Cloud storage for documentation and backup
RR6	Capella Modelling Environment	Platform for integration and validation
RR7	Version Control System	Code management and traceability control

5 APPROACH

5.1 Methodology

<This section should mention the chosen project management methodology that the project will follow. In case that PM² is not the chosen project management methodology, this section should describe the deviations to the standard/proposed project management methodology.

For the domain-specific work (e.g. information systems development), this section should also mention the chosen methodology or describe the deviations from the standard/proposed domain-specific methodologies (e.g. for information systems development, this section should describe the deviations from the standard and/or the proposed IT information systems development methodology).>

5.2 Change Management

<The sections below should describe how changes to the project will be handled, how changes to project documentation and other deliverables will be handled, and how Organisational changes will be handled.

5.2.1 Project Change

*<This section should describe the adopted project change management process. The changes to the content of this document (the description of the project) are subject to formal approval. Please refer to the PM² **Project Handbook** that describes the standard change control process for the project.>*

5.2.2 Configuration Management

<This section should describe the activities concerned with the creation, maintenance and controlled change of project items throughout the life of the project. All items which can only be modified after authorization by the relevant persons or committee (such as the project scope, budget and milestone dates and other key project documentation) should be identified, along with the process for controlling changes to such items.>

*For more details, please refer to the PM² **Project Handbook** which describes the standard configuration management process for projects.>*

6 GOVERNANCE AND STAKEHOLDERS

6.1 Structure

<This section should describe the organisational structure of the project team and stakeholders, preferably providing a graphical representation. Please also refer to (or simply reference) the PM² Methodology which, for instance, defines and describes the members and responsibilities of the Project Steering Committee (PSC) (i.e. Project Owner (PO), Business Manager (BM), Solution Provider (SP), Project Manager (PM)) .>

6.2 Roles and Responsibilities

*<This section should describe the Roles and Responsibilities of the Project Governance. You can refer to the PM² Methodology and include the assignment of the Standard PM² **Roles and Responsibilities**, or/and define any deviations from the Standard.>*

6.3 Other Stakeholders

<This section should identify all the stakeholders of the project and provide a brief profile description of the key stakeholders (and user groups) involved in the project, and the key problems that they perceive to be addressed by the proposed solution. It does not describe their specific requests or requirements as these should be captured in a separate artefact, however, it provides the background and justification for why the requirements are needed.>

Appendix 1: References and Related Documents

<Use this section to reference (or append if needed in a separate annex) any relevant or additional information. Specify each reference or related document by title, version (if applicable), date, and source (e.g. the location of the document or the publishing organisation).>

ID	Reference or Related Document	Source or Link/Location
1	*<Example of a related document>* *<02.Business_Case.XYZ.dd-mm-yyyy.V.1.0.docx>*	*<Example of a location>* *< U:\ PROJECTS\ProjectX \Documents\>*
2	Project folder	*<Insert project folder location.>*
3		