Modernization and Migration of the Umbrella Network Monitoring System – Case Study

Document Information

Title	Modernization and Migration of the Umbrella Network Monitoring System – Case Study
Version | State	1.0 | Final
Author	Łukasz Gałczyński
Contact Details	E-mail: balgalczynski@gmail.com
Date of Creation	16.02.2026
Date of Last Modification	24.02.2026

Version History

Version No.	Publication Date	Author	Details
1.0	24.02.2026	Łukasz Gałczyński	Final version.

Table of Contents

Document Information 1
Version History 1
1. Business Context 5
2. Project Scope 5
3. Stakeholders 6
4. Problem Analysis (As-Is State) 8
5. Proposed Solution (To-Be State) 9
5.1. Architectural Vision & Design Principles 9
5.2. Target Architecture Overview 10
5.3. Integration Layer 10
5.4. Alarm Processing Pipeline 10
5.5. Scalability, Performance & Resilience 11
5.6. Observability & Monitoring 11
5.7. Security & Maintainability 11
5.8. User-Facing Enhancements 12
5.9. Deployment & Infrastructure 12
6. Requirements Documentation 12
6.1. Requirements Elicitation Approach 12
6.2. Functional Requirements – Examples 13
6.2.1. Alarm Ingestion & Processing Requirements 13
6.2.2. User Interaction Requirements 13
6.2.3. Administration & Configuration Requirements 14
6.3. Non-Functional Requirements – Examples 14
6.3.1. Performance & Scalability Requirements 14
6.3.2. Reliability & Resilience Requirements 14
6.3.3. Security Requirements 15
6.3.4. Maintainability Requirements 15
6.4. Domain Model (UML Class Diagram Overview) 15
6.5. Alarm Lifecycle (State Machine Diagram) 17
6.6. Use Case Modelling – Examples 19
6.6.1. Use Case Descriptions (Selected) 20
6.7. User Stories – Examples 22
6.8. Example Test Case Scenarios 25

6.9. Traceability Matrix – Example 27
6.10. Gathering & Documenting Requirements – Summary 28
7. Mock-ups / GUI Prototypes 28
8. Project Outcomes 30
8.1. Technical Outcomes 31
8.2. Operational Outcomes 31
8.3. Business Outcomes 31
8.4. Project & Process Outcomes 31
8.5. Quality Outcomes 32
8.6. Organizational Outcomes 32
8.7. Personal Outcomes 32
9. My Role & Responsibilities 32
9.1. Participation in the Bidding & Offer Preparation Process 33
9.2. Requirements Analysis & Stakeholder Collaboration 33
9.3. Prototyping & Design Support 33
9.4. Project Coordination & Delivery Support 33
9.5. Testing & Quality Assurance 34
9.6. Knowledge Management & Documentation 34
9.7. Ongoing Support & Verification of Incoming Requests 34
10. Lessons Learned 34
10.1. Technical & Architectural Lessons 35
10.2. Development & Delivery Lessons 35
10.3. Collaboration & Stakeholder Lessons 36
10.4. Process & Project Management Lessons 36
10.5. Quality & Testing Lessons 37
10.6. Organizational & Operational Lessons 37
10.7. Security & Compliance Lessons 37
10.8. Personal Lessons 37
11. Summary 38
12. Attachments 39
12.1. List of Figures 39
12.2. List of Tables 39

1. Business Context

The project was initiated by one of the major telecom operators in Poland, whose network infrastructure had been growing steadily in scale and complexity. The operator relied on a legacy Umbrella NMS platform responsible for aggregating alarms and events from multiple underlying Network Management Systems (NMS), which themselves collected telemetry from thousands of network devices across mobile, fixed-line, and transport domains.

Over time, the existing Umbrella NMS had become increasingly difficult to maintain. Several of its core components were outdated, no longer supported by the vendor, and posed operational and security risks. More importantly, the system struggled to handle high-volume alarm bursts generated during severe weather conditions, such as storms or heavy snowfall. These events often led to system slowdowns or temporary unresponsiveness, directly impacting the Network Operations Centre (NOC) and delaying incident resolution.

The operator recognized that the limitations of the legacy platform threatened service reliability and SLA commitments. To ensure operational continuity and prepare for future network expansion, the decision was made to migrate to a modern, scalable Umbrella NMS solution capable of handling significantly higher event loads, improving alarm correlation, and providing better visibility for NOC teams.

2. Project Scope

The project covered a full redesign and migration of the operator's legacy Umbrella NMS platform into a modern, modular, microservices-based solution. The previous system was a monolithic application with outdated components and limited scalability, which made it increasingly difficult to maintain and unable to handle high alarm volumes during severe weather events. The new platform was designed as a distributed architecture composed of multiple specialized modules, each responsible for a distinct part of the alarm processing pipeline.

The scope included the development of dedicated connectors for each integrated NMS system, supporting various communication protocols such as CORBA, SNMP, HTTP, and REST. The new solution introduced several functional engines, including an input rules engine, an enrichment engine integrating with the inventory system, a correlation engine, and a core module responsible for both the front-end and back-end logic. Additional modules were introduced beyond the capabilities of the legacy system, such as a reporting module (e.g., active alarms per connector), a notification module (email/SMS alerts and scheduled reports), a map module for geospatial visualization of network elements, and a heartbeat module for monitoring system health.

From a process perspective, the project covered the migration of all key operational workflows, including alarm ingestion, filtering, enrichment, correlation, deferring, and the creation of "shadow alarms." The initial phase focused on delivering the essential "as-is" functionality with selected enhancements, while subsequent phases introduced optimizations and improvements based on user feedback. Workshops were conducted with two technical stakeholders representing NOC interests and acting as system owners/administrators.

The scope also included data migration activities, API extensions, and protocol upgrades. The system's API was expanded to support REST-based communication for connectors and JSON payloads alongside XML for inventory-related operations. Comprehensive performance, stress, and disaster recovery tests were executed to validate the system's resilience under high alarm bursts and failure scenarios.

As a Business & Systems Analyst, I was responsible for requirements analysis, workshop facilitation, documentation of functional and non-functional requirements, preparing user stories and tasks in JIRA, and maintaining communication with the client. I also contributed to roadmap planning, estimation of change requests, and risk assessment.

Beyond analytical duties, I supported manual testing, verified client-reported issues, coordinated fixes with developers, and acted as a temporary project manager by running daily meetings, aligning delivery timelines, and coordinating deployment activities. The project followed an incremental rollout approach, with the new system gradually taking over functionality from the legacy platform, which remains partially active during the transition. I also participated in UAT and provided post-deployment support.

3. Stakeholders

The project brought together a wide range of stakeholders from both the telecom operator and the delivery team, each contributing distinct expertise and expectations. Their collaboration shaped the direction of the modernization effort, influenced architectural decisions, and ensured that the new Umbrella NMS platform aligned with operational realities. The table below outlines the key stakeholder groups, their roles, and the responsibilities they carried throughout the project.

Party	Stakeholder / Group	Role in the Project	Key Responsibilities & Expectations
Operator	System Administrators (Owners)	Primary business & technical representatives	Providing requirements, validating functionality, supporting UAT, representing NOC interests, ensuring system meets operational needs.
Operator	NOC Representatives	End users (joined during UAT)	Expecting a stable, performant system; reliable alarm handling; clear UI; map support; night mode; reporting issues and CRs post-go-live.
Operator	OSS Maintenance Team	Post-deployment system owners	Taking over daily operations, monitoring system health, coordinating with delivery team on fixes and enhancements.
Delivery Team	System Architect	Technical lead	Designing system architecture, overseeing microservices structure, supporting integration, resolving architectural challenges (e.g., Hazelcast → Redis migration).

Delivery Team	DevOps Engineers	Infrastructure & deployment	Managing Docker Swarm environment, ensuring scalability and resilience, supporting deployments, handling performance and DR testing.
Delivery Team	Project Manager	Delivery coordination	Managing timeline, scope, communication; overseeing sprints; coordinating with client; delegating tasks.
Delivery Team	Business & Systems Analyst (Me)	Requirements, analysis, coordination	Leading workshops, documenting requirements, preparing JIRA tasks, roadmap planning, CR estimation, manual testing, defect triage, client communication, temporary PM duties.
Delivery Team	Developers	Implementation team	Building microservices, connectors, engines (rules, enrichment, correlation), map module, reporting, notifications; resolving defects.
Delivery Team	Automation Tester	Early-phase QA	Preparing automated tests during initial development phase; validating core flows.
Operator	NMS Teams (External Systems)	Data providers	Supplying alarm/event data via CORBA, SNMP, HTTP, REST; providing documentation; supporting integration troubleshooting.
Operator	Field Technicians / On-Site Technical Staff	End users (lower priority)	Rely on accurate, timely alarm data to perform physical interventions such as repairs, equipment replacement, and site visits. High interest in alarm accuracy and clarity, but limited influence on system design.

Table 1 Stakeholders Roles & Responsibilities

Whereas the following matrix positions each stakeholder group according to their influence over the project (power) and their level of involvement or concern (interest). It reflects how their expectations shaped the project's priorities and how communication with each group was managed.

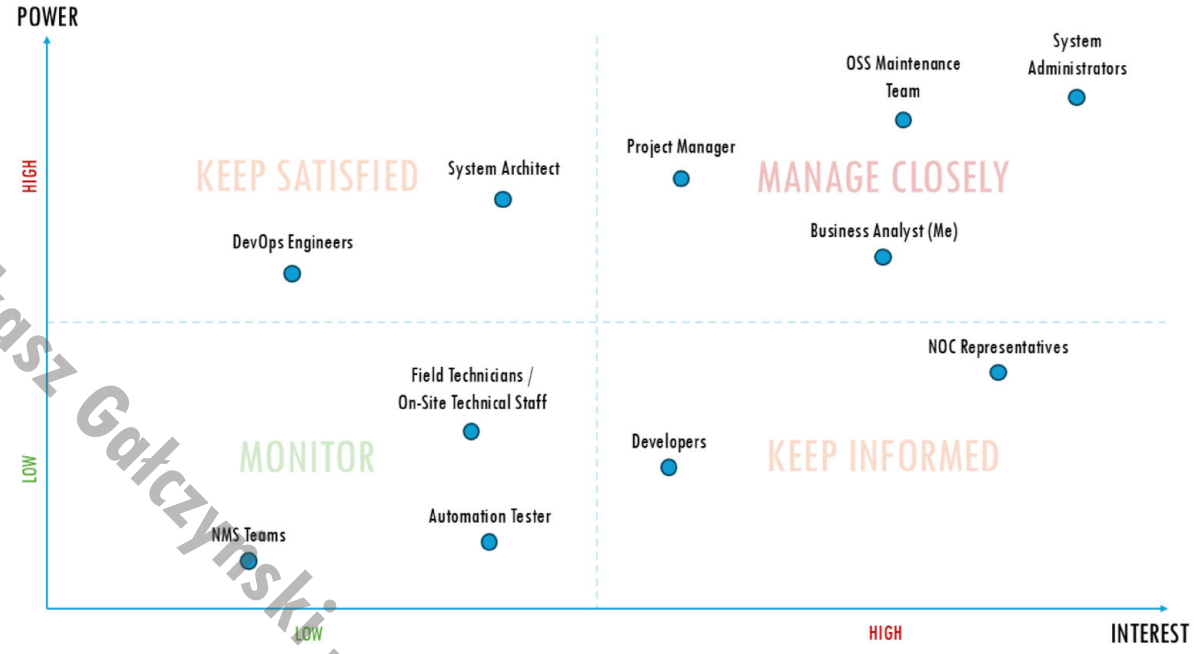

Figure 1 Power-Interest Matrix for Project Stakeholders

4. Problem Analysis (As-Is State)

The modernization initiative was driven by a series of critical limitations and operational risks inherent in the legacy Umbrella NMS platform. The previous system was built as a single monolithic application, which created significant architectural, performance, and maintainability challenges. Any malfunction within a single component – such as a connector – required restarting the entire system, resulting in service interruptions and loss of alarm visibility. This tight coupling made the platform fragile and highly sensitive to failures, especially during periods of elevated alarm traffic.

Performance degradation was one of the most pressing issues. During severe weather events, when alarm volumes surged, the monolithic instance frequently became overloaded, leading to freezes, repeated restarts, or complete unresponsiveness. In such scenarios, alarms were delayed, lost, or duplicated, directly impacting the operator's ability to detect and respond to network incidents. The system's aging technology stack further exacerbated these problems: outdated libraries and components were no longer supported, contained known security vulnerabilities, and limited the ability to extend or evolve the platform.

From an operational perspective, users expressed frustration with system instability and the lack of modern usability features. The absence of a dark mode was a recurring complaint, particularly for NOC operators working night shifts. The legacy system also lacked integrated map visualization, flexible reporting, and configurable notifications – all of which had become essential for efficient network monitoring and field coordination.

Integration with external NMS platforms introduced additional complexity. While HTTP and REST integrations were relatively straightforward, CORBA-based systems frequently caused issues due to unstable subscriptions, inconsistent alarm lists, and unreliable heartbeat signalling. The introduction of SNMP integrations brought its own challenges: the lack of synchronization

mechanisms led to missing alarms, and the system had to be enhanced to handle deduplication of repeated trap messages. Each NMS used its own alarm data model, requiring the creation of mapping scripts to normalize incoming data into a unified internal format.

Architecturally, the monolithic design hindered scalability, high availability, and disaster recovery. Failures in infrastructure components – such as database outages, connector crashes, or disk failures – often caused cascading issues across the entire system. Logging and monitoring capabilities were limited, relying primarily on Grafana, with no centralized ELK-based observability. This made troubleshooting slow and reactive. In contrast, the new system introduced comprehensive monitoring across ELK, Grafana, and additional tools, significantly improving visibility into system behaviour.

Process-wise, the project faced challenges due to limited documentation of the legacy system. Aside from a user manual and NMS integration documents, much of the domain knowledge had to be extracted from system administrators or reverse-engineered from the old codebase. Operational complexity increased further as the operator expanded its network through acquisitions, requiring the legacy system to support a growing number of NMS platforms while simultaneously being phased out.

The business risks associated with maintaining the old system were substantial. Frequent instability, security vulnerabilities, and performance bottlenecks threatened SLA compliance and delayed incident resolution. During high-impact events, the system's inability to reliably process alarms directly affected the operator's ability to dispatch field technicians, perform timely repairs, and maintain service continuity for customers.

These accumulated issues made it clear that the legacy platform could no longer support the operator's operational needs or strategic growth. A complete architectural redesign was necessary to ensure scalability, resilience, and long-term maintainability – ultimately leading to the development of the new microservices-based Umbrella NMS solution.

5. Proposed Solution (To-Be State)

The new Umbrella NMS was conceived as a modern, modular platform designed to overcome the architectural and operational limitations of the legacy system. The primary objective was to deliver a solution that is scalable, resilient, and easier to maintain, while ensuring reliable processing of alarm data and seamless integration with a diverse ecosystem of external network management systems.

5.1. Architectural Vision & Design Principles

The redesign was guided by several strategic goals:

- improving system stability and fault tolerance,
- enabling horizontal scalability during high-load scenarios,
- ensuring rapid recovery from failures,
- simplifying the onboarding of new data sources and integrations,
- and enhancing the overall maintainability of the platform.

To achieve these objectives, the team adopted a distributed, service-oriented architecture, allowing individual components to operate independently and reducing the risk of system-wide failures.

5.2. Target Architecture Overview

The new solution is composed of multiple loosely coupled services, each responsible for a specific part of the alarm processing workflow. This modular structure ensures that issues in one area do not disrupt the entire system and that components can be scaled or updated independently. A central application coordinates the flow of information, while auxiliary services handle tasks such as data processing, enrichment, correlation, reporting, and notifications.

Communication between services is based on asynchronous messaging, which provides buffering during peak load and increases resilience to temporary outages. This design also supports flexible deployment strategies and allows the system to adapt to varying operational demands.

5.3. Integration Layer

The integration layer was re-engineered to support a wide range of external systems with differing communication models. Connectors were redesigned as independent service instances, each responsible for receiving, normalizing, and forwarding alarm data. A unified mapping approach ensures that alarms from different sources are transformed into a consistent internal format, simplifying downstream processing.

The new design also introduces clearer rules for determining whether incoming data represents a new alarm, an update, an action or a duplicate. This significantly improves data quality and reduces noise in the operator interface.

5.4. Alarm Processing Pipeline

The alarm processing workflow was restructured into a series of well-defined stages, including ingestion, normalization, enrichment, rule-based processing, correlation, and persistence. Each stage is handled by a dedicated service, allowing for independent scaling and targeted optimization.

The system also supports bidirectional communication for selected integrations, enabling operators to perform actions directly from the platform when required. The use of asynchronous messaging ensures that high volumes of alarms can be processed reliably, even during large-scale network incidents.

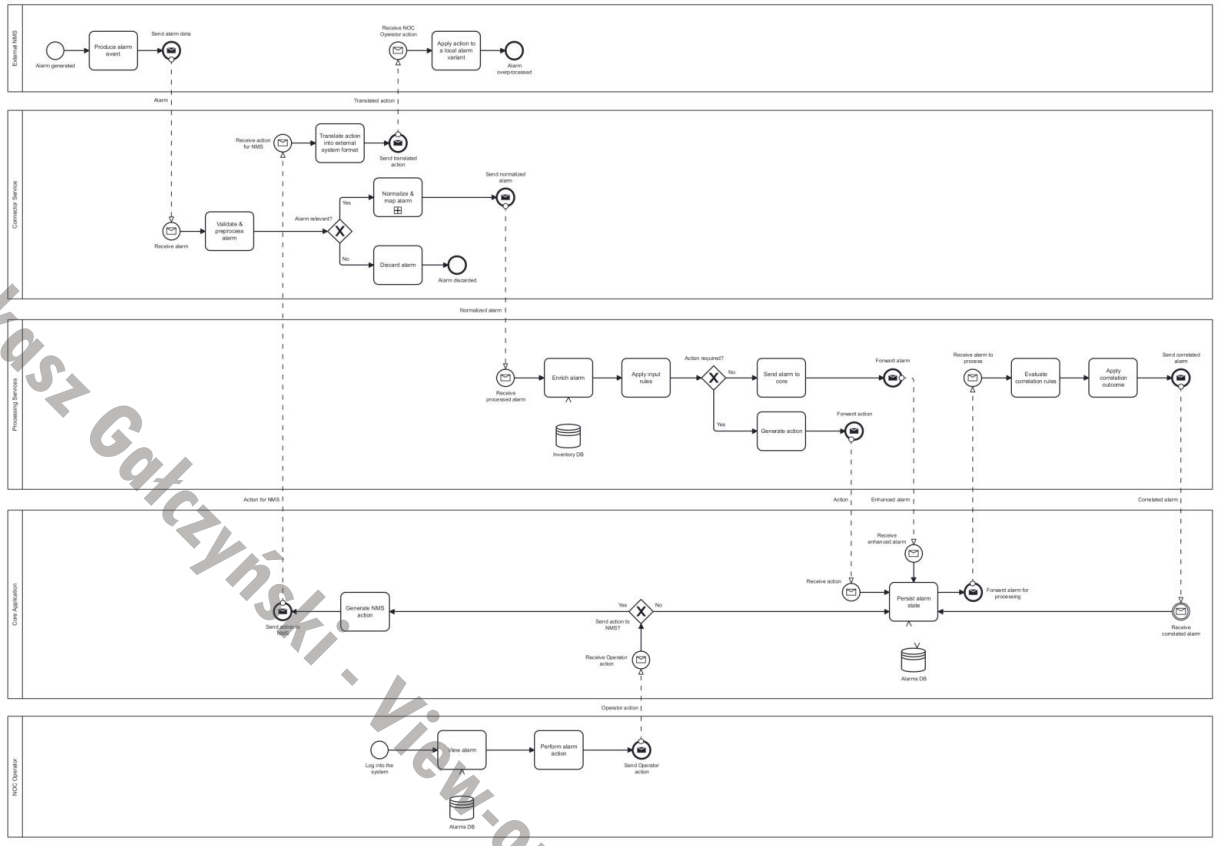

Figure 2 High-Level Alarm Processing Workflow in the modernized Umbrella NMS

5.5. Scalability, Performance & Resilience

The distributed architecture enables horizontal scaling of critical components, ensuring that the system can handle substantial increases in alarm volume without degradation. Buffering mechanisms absorb sudden spikes in traffic, while container-based deployment allows resources to be adjusted dynamically.

These improvements eliminate the bottlenecks associated with the monolithic legacy system and significantly enhance overall system resilience.

5.6. Observability & Monitoring

The new platform introduces comprehensive observability capabilities, including centralized logging, metrics dashboards, and health monitoring. These tools provide deep insight into system behaviour, support faster troubleshooting, and enable proactive detection of performance issues.

The increased visibility also helps operators better understand system load patterns and optimize resource allocation.

5.7. Security & Maintainability

Security was strengthened through centralized authentication, reduced reliance on local accounts, and improved access control mechanisms. The modular architecture simplifies maintenance and accelerates the onboarding of new external systems, often requiring only configuration adjustments or the deployment of additional connector instances.

5.8. User-Facing Enhancements

The new solution introduces several improvements aimed at enhancing operator efficiency and comfort. These include a modernized interface, additional visualization options, configurable notifications, and more flexible reporting capabilities. Faster correlation and deduplication reduce alarm noise, enabling operators to focus on the most relevant events.

5.9. Deployment & Infrastructure

The platform is deployed in a containerized environment, allowing for consistent environments across development, testing, and production. This approach supports rapid deployment, simplified scaling, and streamlined operational management.

6. Requirements Documentation

This section summarizes the key functional and non-functional requirements that guided the design and implementation of the modernized Umbrella NMS platform. The goal is not to reproduce the full internal specification, but to present representative examples and modelling artefacts that illustrate the analytical approach, the structure of the requirements, and the rationale behind the solution.

6.1. Requirements Elicitation Approach

The requirements for the modernized Umbrella NMS were gathered through a structured, multi-stage process designed to ensure a complete understanding of operational needs and technical constraints. Before development began, a series of workshops was conducted with key stakeholders, including System Owners (Administrators) and NOC Operators. These workshops served as the foundation for identifying functional gaps in the legacy system, capturing user expectations, and defining the initial scope of the modernization effort. The outcomes were documented, reviewed, and refined collaboratively.

The development phase followed an agile delivery model, organized into recurring two-week sprints. This iterative approach allowed the team to continuously validate assumptions, refine requirements, and incorporate feedback from stakeholders as the system evolved. Regular sprint reviews and backlog grooming sessions ensured that the requirements remained aligned with operational priorities and that emerging insights from integration and testing were promptly reflected in the specification.

In addition to stakeholder workshops and agile iteration cycles, the requirements process included:

- Analysis of the legacy Umbrella NMS, including reverse engineering of selected components.
- Review of integration documentation provided by external NMS vendors.
- Ongoing consultations with domain experts to clarify business rules and operational workflows.
- Iterative refinement of requirements during development, integration, and UAT phases.

This combined approach ensured that the requirements were comprehensive, validated, and grounded in real operational needs.

6.2. Functional Requirements – Examples

Below are representative examples of functional requirements that shaped the system's behaviour. They are expressed in a concise "The system should…" format commonly used in enterprise documentation. And they are grouped into thematic areas reflecting the core capabilities of the system.

6.2.1. Alarm Ingestion & Processing Requirements

ID	Description	Details	Source
REQF-001	The system should validate, normalize, and enrich all incoming alarms before they are processed by input rules or correlation logic.	This ensures that alarms entering the system follow a consistent structure and contain the contextual information required for downstream processing.	System Owners
REQF-002	The system should allow Administrators to easily create, modify, and maintain mapping scripts that transform incoming alarm data into the internal alarm model.	Mapping logic should be configurable without requiring code changes.	System Owners
REQF-003	The system should provide a straightforward mechanism for defining and updating both input rules and correlation rules.	These rules should be manageable through configuration to support rapid adaptation to operational needs.	System Owners

Table 2 Alarm Ingestion & Processing Requirements – Examples

6.2.2. User Interaction Requirements

ID	Description	Details	Source
REQF-004	The system should allow Operators to browse alarm lists using predefined or custom filters.	Filters should support dynamic creation and modification to accommodate different operational scenarios.	NOC Operators
REQF-005	The system should allow Operators to search within the alarm list using combinations of alarm parameters and values, including logical operators such as AND, OR, and IN.	This enables precise and flexible querying of large alarm datasets.	NOC Operators
REQF-006	The system should allow Operators to perform key actions on alarms, including acknowledge/unacknowledge, defer/undefer, correlate, clear, add comments, update parameters, and escalate.	These actions should be reflected immediately in the user interface and, where applicable, propagated to external systems.	NOC Operators

Table 3 User Interaction Requirements – Examples

6.2.3. Administration & Configuration Requirements

ID	Description	Details	Source
REQF-007	The system should allow Administrators to onboard new external NMS platforms by updating the configuration of existing connectors or by replicating and configuring connector instances.	This process should be lightweight and should not require code-level changes.	System Owners
REQF-008	The system should authenticate standard users through integration with an LDAP identity provider.	Access control should follow organizational policies and role-based permissions.	System Owners
REQF-009	The system should allow local users, such as local administrators or API users, to authenticate based on predefined system configuration.	Local accounts should be limited to specific operational scenarios and managed securely.	System Owners

Table 4 Administration & Configuration Requirements – Examples

6.3. Non-Functional Requirements – Examples

The modernization of the Umbrella NMS placed strong emphasis on performance, resilience, security, and maintainability. The following non-functional requirements illustrate the expected quality attributes of the system.

6.3.1. Performance & Scalability Requirements

ID	Description	Details	Source
REQNF-010	The system should validate, normalize, and enrich all incoming alarms before they are processed by input rules or correlation logic.	This ensures that alarms entering the system follow a consistent structure and contain the contextual information required for downstream processing.	System Owners
REQNF-011	The system should support horizontal scaling of critical components.	In this case, it needs to allow additional instances to be deployed dynamically as the load increases.	System Owners
REQNF-012	The system should maintain consistent response times for core operations.	Especially for operations like alarm ingestion, correlation, and UI updates, even during high-volume scenarios.	System Owners, NOC Operators

Table 5 Performance & Scalability Requirements – Examples

6.3.2. Reliability & Resilience Requirements

ID	Description	Details	Source
REQNF-013	The system should tolerate failures of individual components without interrupting the overall alarm processing pipeline.	It should automatically reroute processing through healthy components to ensure continuous data flow during partial outages.	System Owners, NOC Operators
REQNF-014	The system should ensure that no alarm data is lost during temporary outages.	This can be accomplished by using buffering, retry	System Owners,

		mechanisms, or other resilience strategies.	NOC Operators
REQNF-015	The system should recover automatically from component failures in less than 20 minutes.	After recovery, full functionality should be restored without manual intervention whenever possible.	System Owners, NOC Operators

Table 6 Reliability & Resilience Requirements – Examples

6.3.3. Security Requirements

ID	Description	Details	Source
REQNF-016	The system should authenticate users through a centralized identity provider.	It should ensure compliance with organizational access policies.	System Owners
REQNF-017	The system should restrict the use of local accounts to specific administrative or API-related scenarios, with secure configuration and management.	Such accounts should be tightly controlled, monitored, and limited in scope to minimize security risks and ensure compliance with organizational access policies.	System Owners
REQNF-018	The system should enforce secure communication between internal components and external systems.	This must be achieved by employing suitable authentication and authorization mechanisms.	System Owners

Table 7 Security Requirements – Examples

6.3.4. Maintainability Requirements

ID	Description	Details	Source
REQNF-017	The system should allow Administrators to update configuration, mapping logic, and rules without requiring code changes or redeployment.	This flexibility enables rapid adaptation to evolving operational needs and minimizes the dependency on development cycles.	System Owners
REQNF-018	The system should provide clear separation of responsibilities between components, enabling independent updates and reducing operational risk.	This modular structure ensures that changes in one area do not introduce unintended side effects in others, simplifying long-term maintenance	System Owners
REQNF-019	The system should support onboarding of new external NMS platforms through configuration rather than custom development whenever feasible.	This approach shortens integration timelines and reduces the overall cost and complexity of expanding the system's coverage.	System Owners

Table 8 Maintainability Requirements – Examples

6.4. Domain Model (UML Class Diagram Overview)

The following UML class diagram provides a high-level view of the core domain concepts that form part of the modernized Umbrella NMS platform. For demonstration purposes, this case study includes only a selected subset of classes, attributes, and methods that illustrate the system's structural and behavioural foundations. The diagram does not represent the full production model; several classes, relationships, and attribute definitions have been intentionally simplified or adapted to fit the narrative scope of this document. As such,

the model should be interpreted as an illustrative extract rather than a complete one-to-one representation of the original system architecture.

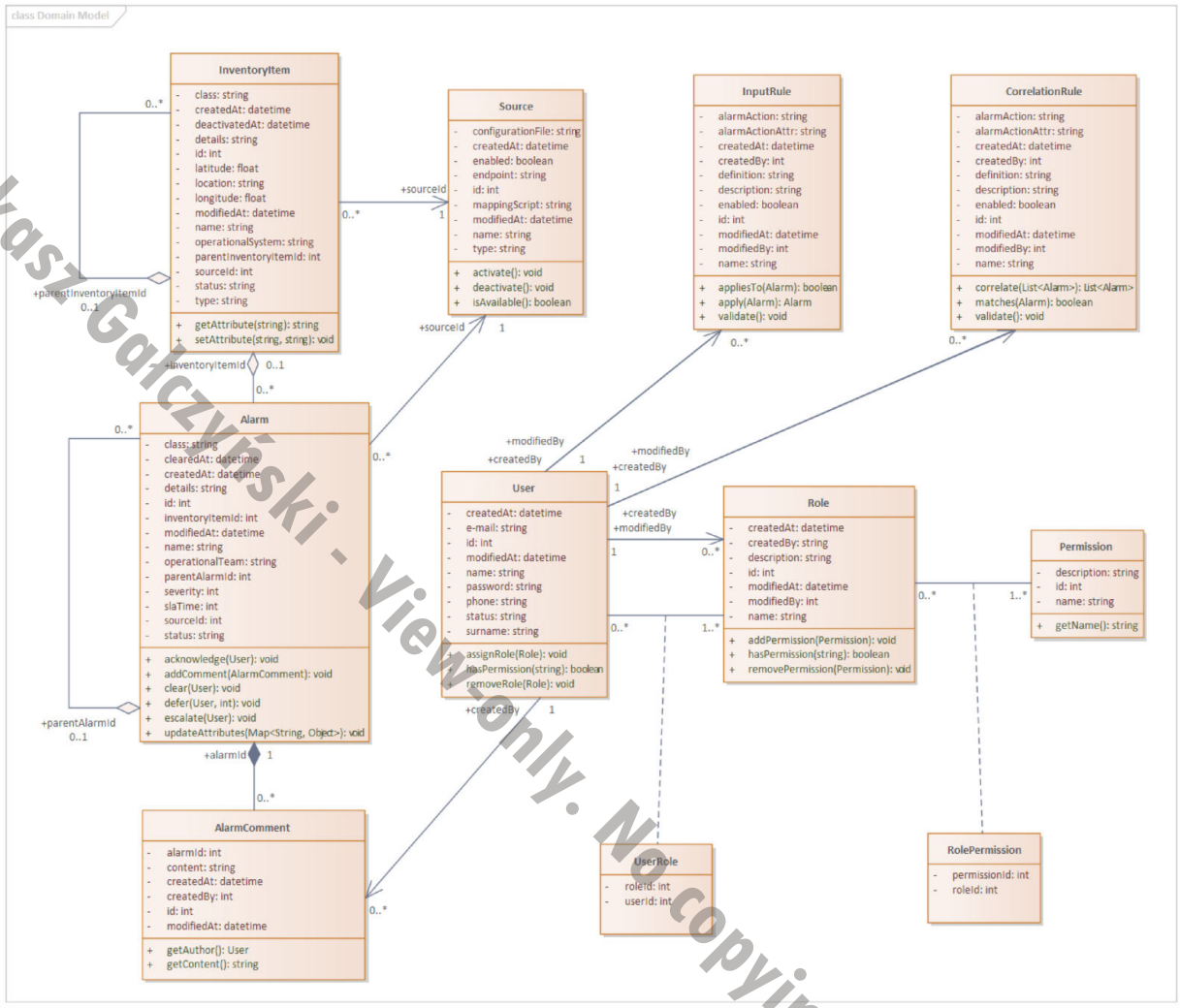

Figure 3 High-level UML Class Diagram with Selected Classes, Attributes & Methods

Below is a summary of the key domain classes included in the diagram, outlining their primary responsibilities and role within the system.

1. **Alarm**

Represents a single alarm event received from an external NMS or connector. It stores core alarm attributes such as severity, status, and descriptive information. The class also exposes high-level behavioural operations (e.g., acknowledge, clear, escalate) that reflect actions performed by NOC Operators.

2. **AlarmComment**

Captures Operator-provided comments associated with a specific alarm. Each comment is linked to both the alarm and the user who created it, providing contextual information for operational workflows and audit purposes.

3. **InventoryItem**

Represents an element of the network or service inventory (e.g., node, interface, service). Inventory items may be referenced by alarms for enrichment purposes, enabling Operators to understand the affected infrastructure and its attributes.

4. **Source**

Models an external NMS or connector responsible for sending alarms into the system. It contains configuration details such as endpoint, type, and activation status, and provides basic operational methods (e.g., activate, deactivate).

5. **InputRule**

Defines a rule applied to incoming alarms during the normalization and preprocessing stage. Input rules determine whether they apply to a given alarm and can transform alarm attributes before further processing.

6. **CorrelatorRule**

Represents a correlation rule used to relate multiple alarms based on defined logic. These rules evaluate alarm sets and produce correlation outcomes that support incident detection and noise reduction.

7. **User**

Represents a system user, including Operators and Administrators. Users are associated with roles and permissions that determine their access rights and allowed actions within the platform.

8. **Role**

A grouping of permissions assigned to users. Roles simplify access management by bundling related permissions into reusable sets aligned with operational responsibilities.

9. **Permission**

Represents a single, granular access right within the system. Permissions are associated with roles and ultimately determine what actions a user is authorized to perform.

10. **UserRole (association class)**

Represents the assignment of a role to a user. This class captures the many-to-many relationship between users and roles in a structured and traceable way.

11. **RolePermission (association class)**

Represents the assignment of a permission to a role. It models the many-to-many relationship between roles and permissions, forming the foundation of the RBAC model.

6.5. Alarm Lifecycle (State Machine Diagram)

The following state machine diagram illustrates the high-level lifecycle of an alarm within the Umbrella NMS platform. It presents both the technical processing stages and the operational actions performed by users, showing how an alarm progresses from initial reception through validation, normalization, enrichment, operator handling, and eventual

resolution. For demonstration purposes, the diagram reflects a simplified extract of the full production lifecycle.

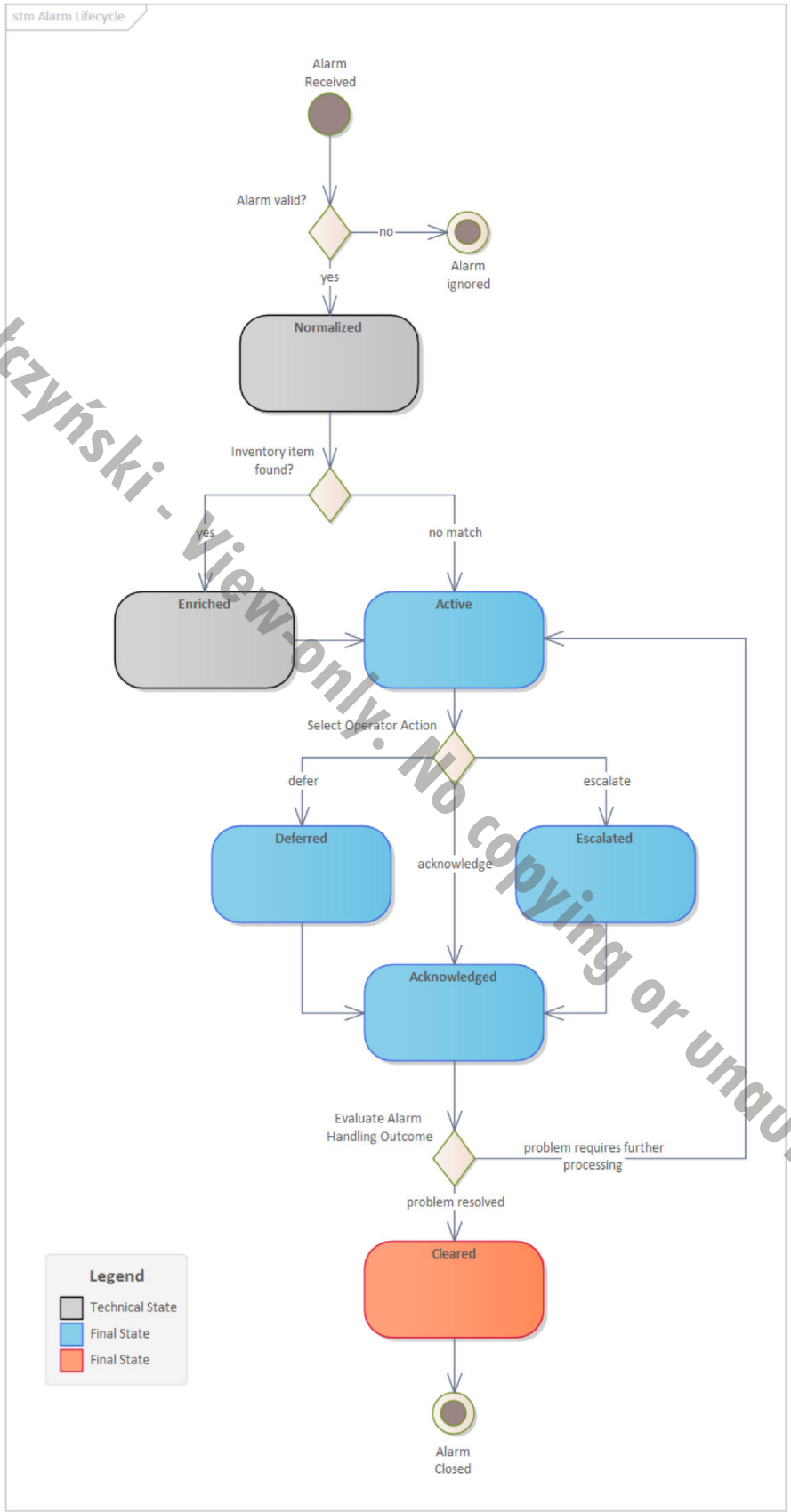

Figure 4 State Machine Diagram with the High-level Lifecycle of an Alarm

The diagram depicts the sequential flow of an alarm as it moves through the system:

- After an alarm is received, the platform first validates its structure and content.
- Valid alarms proceed to the **Normalized** state, after which the system attempts to associate them with an existing inventory element.
- If a matching item is found, the alarm transitions to **Enriched**; otherwise, it continues directly to **Active**, where it becomes visible to Operators.
- From the **Active** state, Operators may choose to **Acknowledge**, **Defer**, or **Escalate** the alarm.
- Deferred and escalated alarms ultimately converge into the **Acknowledged** state, indicating that the alarm has been taken into operational handling.
- A subsequent decision evaluates whether the underlying issue has been resolved. If so, the alarm transitions to **Cleared** and then to the final **Alarm Closed** state. If the issue persists, the alarm returns to **Active** for further processing.

6.6. Use Case Modelling – Examples

This subsection presents a simplified use case model illustrating how key actors interact with the modernized Umbrella NMS platform. The diagram prepared in Enterprise Architect (in the UML notation) and the accompanying list of use cases represent a selected subset of the system's full functionality, included here solely for demonstration purposes within this case study. The use cases are mapped to the previously defined functional requirements (i.e. Functional Requirements – Examples) to ensure traceability and clarity of scope.
The subsection concludes with detailed specifications of two representative use cases, which will be expanded in the following part.

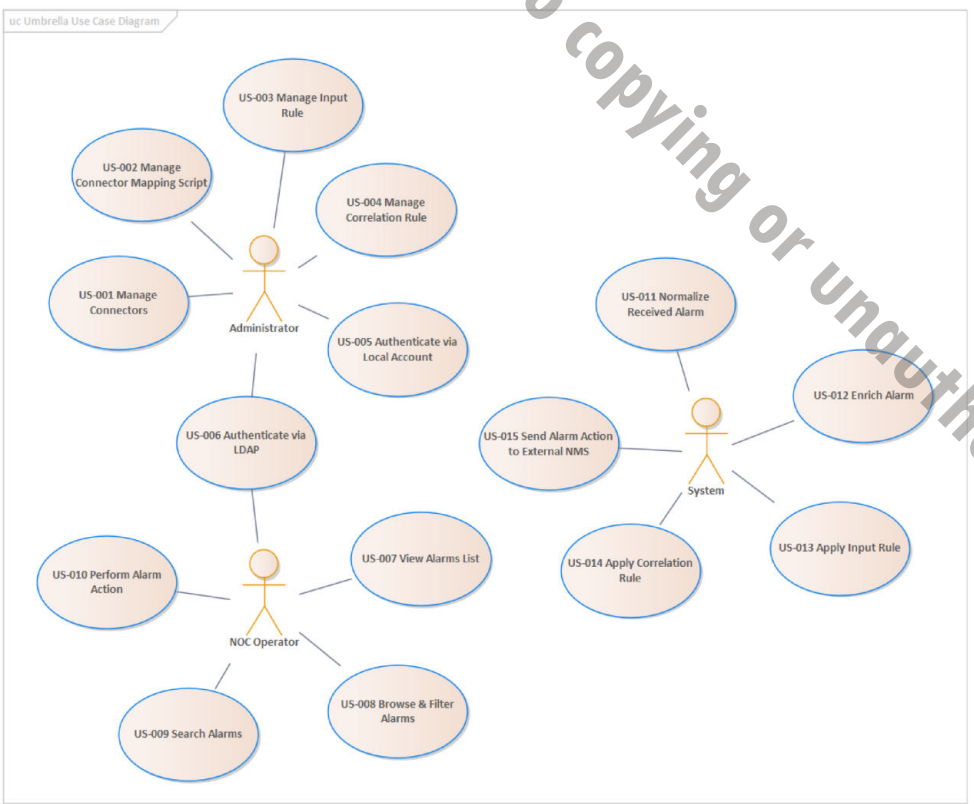

Figure 5 Diagram of Sample Use Cases for the Umbrella NMS

ID	Use Case Header	Source Requirement
US-001	Manage Connectors	REQF-007
US-002	Manage Connector Mapping Script	REQF-002
US-003	Manage Input Rule	REQF-003
US-004	Manage Correlation Rule	REQF-003
US-005	Authenticate via Local Account	REQF-009
US-006	Authenticate via LDAP	REQF-008
US-007	View Alarms List	REQF-004 / REQF-005
US-008	Browse & Filter Alarms	REQF-004
US-009	Search Alarms	REQF-005
US-010	Perform Alarm Action	REQF-006
US-011	Normalize Received Alarm	REQF-001
US-012	Enrich Alarm	REQF-001
US-013	Apply Input Rule	REQF-001
US-014	Apply Correlation Rule	REQF-001
US-015	Send Alarm Action to External NMS	REQF-006

Table 9 Use Case Mapping Matrix for Functional Requirements

6.6.1. Use Case Descriptions (Selected)

US-001 – Manage Connectors

Purpose

To allow Administrators to configure, update, and maintain connectors responsible for receiving alarms from external NMS platforms.

Primary Actor

System Administrator

Supporting Actors

External NMS (indirect)

Preconditions

- Administrator is authenticated and authorized to manage connectors.
- System configuration interface is available.

Main Flow

1. Administrator opens the connector management interface.
2. System displays a list of existing connectors and their statuses.
3. Administrator selects an existing connector or chooses to create a new one.
4. Administrator updates configuration parameters (e.g., endpoint, protocol, authentication, mapping reference).
5. Administrator saves the configuration.
6. Administrator initiates redeployment.
7. System applies the updated configuration.
8. System confirms successful update and activates the connector.

Alternate Flows

A1 – Invalid Configuration

- At step 6, redeployment fails.
- System displays detailed error messages.
- Administrator corrects the configuration and retries.

A2 – Connector Activation Failure

- At step 8, connector cannot be activated (e.g., unreachable endpoint).
- System logs the failure and notifies the administrator.
- Administrator may retry activation or revert changes.

Postconditions

- Connector configuration is updated and active.
- System is ready to receive alarms from the configured external NMS.

US-010 – Perform Alarm Action

Purpose

To allow NOC Operators to execute operational actions on alarms, such as acknowledge, clear, defer, escalate, or add comments.

Primary Actor

NOC Operator

Supporting Actors

External NMS (for action propagation, if applicable)

Preconditions

- Operator is authenticated and authorized to perform alarm actions.
- The alarm exists and is visible in the Operator's current view.

Main Flow

1. Operator selects an alarm from the list.
2. System displays available actions based on alarm state and Operator permissions.
3. Operator chooses an action (e.g., acknowledge).
4. System validates that the action is allowed in the current alarm state.
5. System applies the action and updates the alarm state.
6. System logs the action for audit purposes.
7. If necessary, the system sends the action directly to the connector service, and the connector forwards it to the External NMS.
8. System updates the UI to reflect the new alarm state.

Alternate Flows

A1 – Action Not Allowed

- At step 4, the action is not permitted (e.g., clearing an already cleared alarm).
- System displays an error message explaining the restriction.
- Operator may choose a different action.

A2 – External NMS Propagation Failure

- At step 7, the External NMS does not accept or confirm the action.
- System logs the failure and notifies the Operator.
- Alarm state may remain updated locally or revert based on configuration.

A3 – Insufficient Permissions

- At step 3, Operator selects an action they are not authorized to perform.
- System blocks the action and displays a permission error.

Postconditions

- Alarm state is updated according to the executed action.
- Audit log contains a record of the action.
- External NMS is updated if applicable.

6.7. User Stories – Examples

To complement the formal functional requirements, the following user stories capture the system's expected behaviour from the perspective of its primary users and stakeholders, providing a clear, actionable foundation for implementation and acceptance testing. These user stories were derived directly from the functional requirements defined earlier in this document (i.e. Functional Requirements – Examples) to ensure full alignment between specification, design, and delivery. They also maintain consistency with the use case model presented in the subsection Use Case Modelling – Examples, reflecting the same core interactions and system capabilities.

ID	User Story	Preconditions	Acceptance Criteria	Source Requirement & Use Case
US-001	As an Administrator, I want the system to validate, normalize, and enrich incoming alarms so that downstream rules and correlation logic can operate on consistent and complete data.	- An external NMS sends an alarm to the platform. - The connector for that NMS is configured and active.	- Incoming alarms are checked for structural correctness. - Invalid alarms are rejected or flagged according to configuration. - Valid alarms are transformed into the internal alarm model.	REQF-001; US-011, US-012, US-013, US-014

			- Enrichment data (e.g., inventory attributes) is applied when available. - The resulting alarm is ready for input rule and correlation processing.	
US-002	As an Administrator, I want to easily create and update mapping scripts so that incoming alarms can be transformed into the system's internal model without code changes.	- Administrator has access to configuration tools. - A connector is configured to receive alarms.	- Mapping scripts can be created, edited, and saved through configuration. - Updated mappings are applied to subsequent incoming alarms.	REQF-002; US-002
US-003	As an Administrator, I want to define and update input rules and correlation rules so that alarm processing can adapt to operational needs.	- Administrator has permission to manage rules. - Rule engine is operational.	- Rules can be added, modified, or removed through configuration. - The system validates rule syntax and structure. - Updated rules are applied to new alarms immediately or after a controlled reload. - Correlation outcomes reflect the updated rule logic.	REQF-003; US-003, US-004
US-004	As a NOC Operator, I want to browse alarms using predefined or custom filters so that I can focus on the most relevant events.	- Operator is authenticated. - Alarm data is available in the system.	- Operator can select predefined filters. - Operator can create new filters based on alarm attributes. - Filtered alarm lists update immediately. - Filters can be saved for future use.	REQF-004; US-007, US-008

US-005	As a NOC Operator, I want to search alarms using combinations of parameters and logical operators so that I can precisely locate alarms of interest.	- Operator is authenticated. - Alarm list is loaded.	- Search supports AND, OR, and IN operators. - Search supports multiple alarm attributes (e.g., severity, source, timestamp). - Results update dynamically based on the search query. - Invalid queries return a meaningful error message.	REQF-005; US-007, US-009
US-006	As a NOC Operator, I want to perform actions on alarms (acknowledge / unacknowledge, clear, defer / undefer, correlate, comment, update parameters, escalate) so that I can manage incidents effectively.	- Operator is authenticated and authorized. - Alarm exists and is visible in the UI.	- Operator can execute all supported actions from the UI. - Actions update the alarm state immediately. - Actions are logged for audit purposes. - If applicable, actions are forwarded to the external NMS. - UI reflects the updated alarm state without delay.	REQF-006; US-010
US-007	As an Administrator, I want to onboard new external NMS platforms through configuration so that the system can be extended without custom development.	- Administrator has access to connector configuration. - Documentation for the new NMS is available.	- New NMS can be added by updating connector configuration or replicating an existing connector. - Mapping scripts can be defined for the new NMS. - No code changes are required. - The system successfully receives and processes alarms from the new NMS.	REQF-007; US-001
US-008	As an Administrator, I want the system to ensure that all standard users	- LDAP integration is configured.	- User can log in using LDAP credentials.	REQF-008; US-006

	authenticate through LDAP as the primary login method so that access control remains consistent with organizational identity policies.	- At least one user exists in the LDAP directory.	- Authentication failures return meaningful error messages. - User roles and permissions are applied based on LDAP attributes or system configuration.	
US-009	As a Local Administrator or API User, I want to authenticate using predefined local credentials so that I can perform system-level or automated tasks.	- Local accounts are configured in the system. - User has appropriate permissions.	- Local users can authenticate using system-defined credentials. - All authentication attempts are logged. - Local accounts can be enabled, disabled, or updated by authorized Administrators.	REQF-009; US-005

Table 10 User Stories – Examples (Based on Earlier Functional Requirements)

6.8. Example Test Case Scenarios

The following test case scenarios illustrate how selected system behaviours can be verified in practice. They were derived directly from the use cases presented in the subsection Use Case Descriptions (Selected), ensuring full traceability between functional requirements, behavioural modelling, and validation activities. The scenarios serve as representative examples rather than an exhaustive test suite, demonstrating how key interactions of the modernized Umbrella NMS platform can be tested in a structured and repeatable manner.

ID	Source Use Case	Objective	Preconditions	Steps	Expected Results
TS-001.01	US-001	Create a New Connector with Valid Configuration	- Administrator is authenticated. - System configuration interface is available.	1. Navigate to the connector management section. 2. Select “Create new connector”. 3. Enter valid configuration parameters (endpoint, protocol, credentials, mapping reference). 4. Save the configuration. 5. Redeploy	- Connector is created and activated. - Confirmation message is displayed. - Connector appears in the list with status “Active”.

				the core component of the system.	
TS-001.02	US-001	Update Existing Connector with Invalid Configuration	- Administrator is authenticated. - At least one connector exists.	1. Open an existing connector. 2. Modify configuration parameters with invalid values (e.g., malformed URL). 4. Save the configuration.	- System rejects the configuration. - Detailed validation errors are displayed. - Connector remains unchanged and active.
TS-001.03	US-001	Connector Activation Fails Due to External Endpoint Unavailability	- Administrator is authenticated. - External NMS endpoint is intentionally unreachable.	1. Create or update a connector with valid syntax but unreachable endpoint. 2. Save the configuration. 3. Redeploy the core component of the system.	- System validates syntax successfully. - Activation attempt fails. - System logs the failure and informs the Administrator. - Connector status is set to “Inactive” or “Error”.
TS-010.01	US-010	Acknowledge an Active Alarm Successfully	- Operator is authenticated and authorized. - Alarm exists in “Active” state.	1. Select the alarm from the list. 2. Choose the “Acknowledge” action from the context menu. 3. Confirm the action if prompted.	- System validates that the action is allowed. - Alarm state changes to “Acknowledged”. - Action is logged. - UI updates immediately. - If configured, action is propagated to the External NMS.
TS-010.02	US-010	Attempt to Clear an Already Cleared Alarm	- Operator is authenticated. - Alarm exists in “Cleared” state.	1. Select the cleared alarm. 2. Attempt to perform the “Clear” action from the context menu.	- System blocks the action. - Error message indicates that the action is not allowed. - Alarm state remains unchanged. - No propagation to External NMS occurs.

TS-010.03	US-010	Alarm Action Propagation Fails on External NMS	- Operator is authenticated. - Alarm exists and supports propagation. - External NMS is temporarily unavailable.	1. Select an alarm. 2. Perform an action that requires propagation (e.g., acknowledge). 3. System attempts to send the action to the External NMS.	- Local alarm state is updated according to system configuration (either committed or rolled back). - System logs the propagation failure. - Operator is notified of the issue. - Audit log contains the attempted action.

Table 11 Example Test Case Scenarios (Based on Earlier Functional Requirements)

6.9. Traceability Matrix – Example

The following traceability matrix provides a consolidated view linking functional requirements, user stories, use cases, and example test scenarios included in this case study. It illustrates how selected behavioural and functional aspects of the modernized Umbrella NMS platform are connected across the specification and validation layers.

Functional Requirement	User Story	Use Case	Test Scenario
REQF-001 Alarm validation, normalization, enrichment	US-001 Validate, normalize, enrich incoming alarms	US-011 Normalize Received Alarm US-012 Enrich Alarm US-013 Apply Input Rule US-014 Apply Correlation Rule	*(no example test case)*
REQF-002 Mapping scripts	US-002 Manage mapping scripts	US-002 Manage Connector Mapping Script	*(no example test case)*
REQF-003 Input & correlation rules	US-003 Manage input & correlation rules	US-003 Manage Input Rule US-004 Manage Correlation Rule	*(no example test case)*
REQF-004 Browsing alarms	US-004 Browse alarms with filters	US-007 View Alarms List US-008 Browse & Filter Alarms	*(no example test case)*
REQF-005 Searching alarms	US-005 Search alarms	US-007 View Alarms List US-009 Search Alarms	*(no example test case)*
REQF-006 Alarm actions	US-006 Perform alarm actions	US-010 Perform Alarm Action	TS-010.01Acknowledge Active Alarm TS-010.02 Clear Already Cleared Alarm TS-010.03 Propagation Failure to External NMS

REQF-007 Onboarding new NMS	US-007 Onboard new NMS	US-001 Manage Connectors	TS-001.01 Create Connector TS-001.02 Update Invalid Connector TS-001.03 Activation Failure
REQF-008 LDAP authentication	US-008 LDAP as primary login	US-006 Authenticate via LDAP	*(no example test case)*
REQF-009 Local authentication	US-009 Local admin/API login	US-005 Authenticate via Local Account	*(no example test case)*

Table 12 Traceability Matrix (Consolidated View)

Since the case study focuses on a demonstrative subset of the system's capabilities, the matrix includes only those requirements, user stories, and use cases for which detailed modelling and test scenarios were prepared. This example highlights the end-to-end traceability path (from requirement definition through behavioural modelling to test execution) ensuring clarity, consistency, and alignment across all artefacts. Verifying the relationships between these artefacts through such a matrix also helps ensure that all planned functionalities are properly addressed, reducing the risk of gaps during implementation and testing.

6.10. Gathering & Documenting Requirements – Summary

The requirements for the modernized Umbrella NMS were gathered through a structured, iterative, and collaborative process that combined stakeholder workshops, analysis of the legacy platform, and continuous refinement during development. Early workshops with System Owners and NOC representatives established the foundation for understanding operational needs, functional gaps, and pain points in the existing system. These insights were complemented by a detailed review of integration documentation from external NMS platforms and reverse engineering of selected components of the legacy solution.

Throughout the project, requirements were validated and expanded using an agile delivery model. Recurring sprint reviews, backlog grooming sessions, and hands-on testing enabled stakeholders to provide ongoing feedback, ensuring that the specification evolved in line with real operational priorities. This iterative approach allowed me and the team to refine functional requirements – such as alarm ingestion, normalization, enrichment, correlation, and user interactions – as well as non-functional requirements related to performance, scalability, resilience, security, and maintainability.

The resulting documentation includes representative functional and non-functional requirements, a high-level domain model, a state machine diagram illustrating the alarm lifecycle, selected use cases, example user stories, test scenarios, and a traceability matrix linking these artefacts. Together, these materials demonstrate a comprehensive and methodical approach to requirements engineering, ensuring that the modernized Umbrella NMS is aligned with stakeholder expectations, operational workflows, and long-term architectural goals.

7. Mock-ups / GUI Prototypes

As part of the analytical and design activities carried out during the modernization project, a set of conceptual mock-ups was created to visualize how the new Umbrella NMS interface

could support key operational workflows. These prototypes were introduced early in the requirements phase to facilitate discussions with stakeholders, validate usability expectations, and ensure that the proposed interaction patterns aligned with real NOC practices. The following screens present both light and dark mode variants, reflecting the diverse working conditions and preferences of Operators.

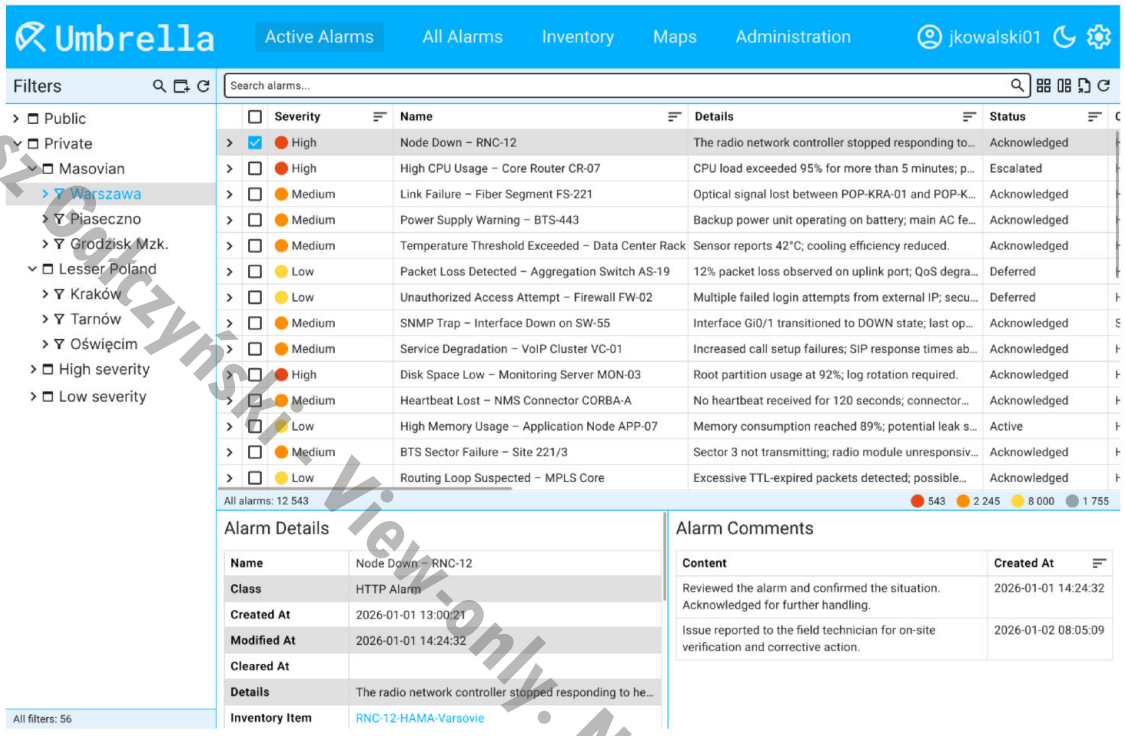

Figure 6 Conceptual mock-up of the Umbrella NMS interface in light mode (the proposed layout for alarm monitoring, filtering, and Operator workflows)

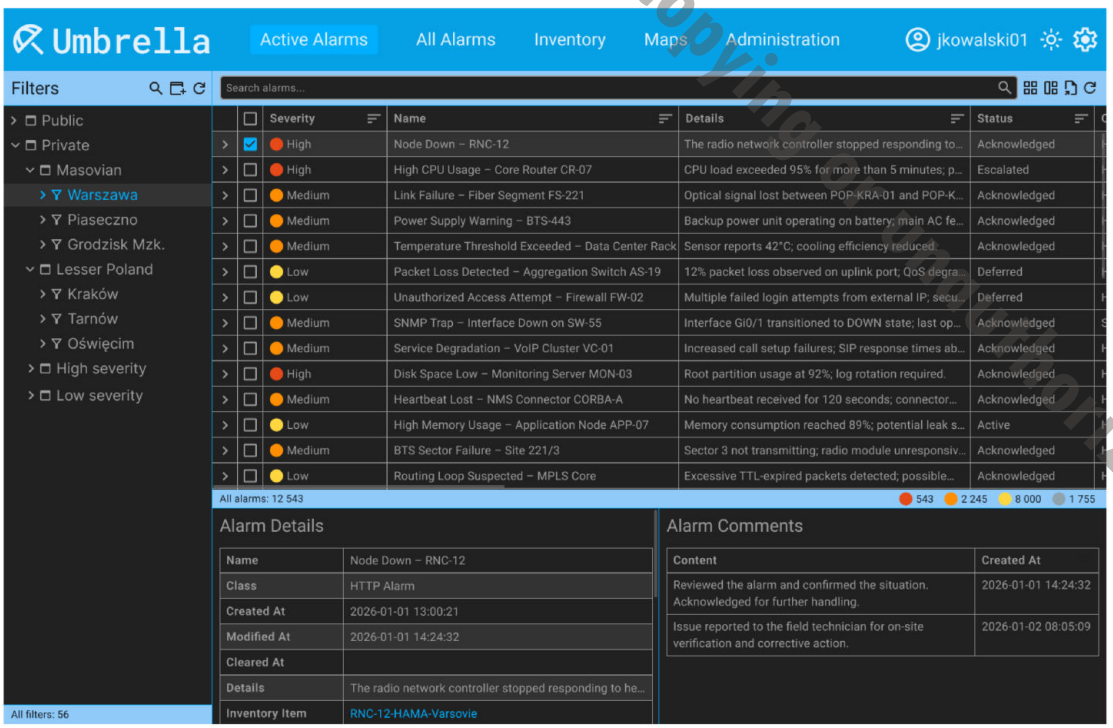

Figure 7 Conceptual mock-up of the Umbrella NMS interface in dark mode (support night-shift operations and improve Operator comfort)

The mock-ups served as an essential communication tool throughout the requirements-gathering stage. By introducing visual concepts early, stakeholders were able to review proposed layouts, provide immediate feedback, and refine expectations before any front-end development began. This iterative approach significantly reduced the risk of misunderstandings, ensured alignment with operational workflows, and ultimately saved development time by giving the engineering team clear, validated UI guidelines.

The prototypes focus on the main operational screen used by NOC Operators and highlight three core interface areas:

- Alarm List Panel – displays active alarms with key attributes such as severity, name, details, and status. The list supports filtering by region, severity, and category, reflecting the requirement for flexible alarm browsing and rapid triage during high-volume scenarios.
- Alarm Details Panel – shows contextual information for the selected alarm, including timestamps, class, associated inventory item, and descriptive details. This mirrors the enriched data model introduced in the modernized system, where normalization and enrichment ensure consistent and actionable alarm information.
- Alarm Comments Panel – provides a chronological view of Operator comments, supporting collaborative handling, traceability, and operational continuity across shifts.

Both the light and dark mode variants demonstrate how the interface adapts to different working environments. The dark mode mock-up directly addresses a recurring pain point from the legacy system, where the absence of a night-friendly UI caused discomfort during overnight operations. The light mode version offers an alternative suited for daytime use.

Throughout the project, these mock-ups were iteratively updated based on stakeholder input. System Owners and NOC representatives reviewed each version, suggested improvements, and validated the usability of proposed layouts. Their feedback was incorporated into subsequent iterations, ensuring that the final design direction aligned with real operational needs. This collaborative process not only improved the quality of the UI concepts but also streamlined front-end implementation by providing developers with clear, pre-approved visual guidelines.

The resulting prototypes are intentionally high-level and conceptual. Their purpose is not to define pixel-perfect UI, but to illustrate how the redesigned system could present alarm data in a clear, structured, and operator-centric manner, supporting faster decision-making, reducing cognitive load, and enhancing overall user experience.

8. Project Outcomes

The modernization of the Umbrella NMS platform delivered a wide range of technical, operational, and organizational benefits. While the primary objective was to replace an aging monolithic system with a scalable and maintainable architecture, the project ultimately produced improvements that extended far beyond the technical domain. The outcomes below summarize the most significant achievements observed during and after implementation.

8.1. Technical Outcomes

The transition to a microservices-based architecture fundamentally improved the system's flexibility and resilience. Individual services can now be replicated or assigned additional resources directly through Docker Swarmpit, enabling rapid scaling during high-load scenarios. Crucially, failures or restarts of individual components no longer cause system-wide outages, as was the case with the legacy monolith.

The new platform also demonstrated a dramatic increase in processing capacity. While the previous system struggled with several thousand alarms per second during peak events, the modernized solution reliably handles volumes in the hundreds of thousands. This improvement is further reinforced by the fact that the new system currently supports approximately 1.5 times more connectors than the legacy platform.

Although the absolute number of failures has not changed significantly, the system's ability to automatically recover from disruptions has improved substantially. Components restart cleanly, dependencies reconnect without manual intervention, and the overall recovery time after an incident is markedly shorter. The integrated flow spanning connectors, the enrichment module, the input rules engine, the Core service, and the correlation engine proved to be one of the strongest architectural successes.

8.2. Operational Outcomes

From an operational perspective, the new platform has noticeably improved the daily work of NOC teams. Faster system response, a more stable interface, and the introduction of dark mode significantly enhanced operator comfort, particularly during night shifts.

The addition of new modules, such as Maps and Reports, expanded the operational toolkit available to users. These features provided new ways to visualize network issues, analyse alarm patterns, and support decision-making during incidents.

8.3. Business Outcomes

The modernization also delivered measurable business value. A clear example is the synchronization of alarms from some of the CORBA systems: previously, this process could take up to an hour and would block other parts of the system, causing operational delays. In the new architecture, synchronization completes within minutes and no longer impacts the availability of other services.

The simplified onboarding of new NMS platforms, achieved through connector replication and configuration updates, reduced the cost and effort associated with network expansion. As long as the external system communicates via CORBA, HTTP, SNMP, or REST, integration can be completed without code-level changes.

8.4. Project & Process Outcomes

The iterative approach to requirements gathering, combined with early prototyping, proved highly effective. Mock-ups accelerated front-end development by providing developers with clear, validated UI guidelines, while structured requirements made it easier to plan work, distribute tasks, and align the roadmap with the client's priorities.

Stakeholder collaboration was another strong point of the project. System Owners and NOC representatives were engaged, communicative, and open to feedback, which significantly streamlined decision-making. Workshops and ongoing coordination emerged as the most valuable project activities, ensuring alignment across teams and reducing rework.

8.5. Quality Outcomes

Data quality improved notably, especially for SNMP-based integrations, where normalization and deduplication mechanisms reduced noise and inconsistencies. Documentation also reached a higher level of maturity: key materials were consolidated in the Client's Confluence knowledge base, while developers maintained technical documentation directly within the application repository.

Although long-term maintainability will be fully validated by the newly formed OSS maintenance team, the modular architecture and clearer separation of responsibilities already indicate a more sustainable operational model.

8.6. Organizational Outcomes

The modernization effort also brought indirect benefits. The implementation introduced updated libraries, security mechanisms, and authentication workflows, replacing legacy scripts with structured logic embedded in dedicated services. This shift improved both security posture and long-term maintainability.

The project also encouraged better cross-team collaboration and established practices that can serve as a reference for future modernization initiatives within the organization.

8.7. Personal Outcomes

From the perspective of the Business & Systems Analyst role, the project's success stemmed from a combination of thorough requirements analysis, effective coordination, and hands-on involvement in testing. The ability to translate stakeholder needs into actionable development tasks, maintain alignment across teams, and support the project during periods of limited Project Manager availability contributed directly to the quality of the final deliverable.

The new system preserved the familiar user experience of the legacy platform while delivering a faster, more secure, and feature-rich environment. This balance between continuity and innovation played a key role in the positive reception from end users.

9. My Role & Responsibilities

As the Business & Systems Analyst on the modernization project, I played a central role in connecting business expectations with technical execution. My responsibilities covered the full lifecycle of the initiative: from the initial bidding phase, through requirements analysis and design support, to coordination, testing, and post-deployment activities. This broad involvement allowed me to influence both the strategic direction of the solution and the quality of its final delivery.

9.1. Participation in the Bidding & Offer Preparation Process

Before the project formally began, I contributed to the preparation of the commercial and technical offer submitted by my software house. Working closely with a DevOps Engineer and the Technical Lead (both of whom later joined the delivery team) I helped develop the initial solution concept, outline the proposed architecture, and estimate the scope, cost, and effort required for implementation.

This early involvement ensured that the offer was both technically sound and aligned with the Client's expectations. The success of this bidding phase enabled our team to secure the project and proceed with the actual design and development of the modernized Umbrella NMS platform.

9.2. Requirements Analysis & Stakeholder Collaboration

A major part of my role focused on gathering, structuring, and validating requirements. I facilitated brainstorming workshops with System Owners (and at the same time representatives of the NOC team), combining open discussion with walkthroughs of user journeys in the legacy system. This approach helped uncover functional gaps, clarify operational pain points, and ensure that the new solution addressed real-world workflows.

Throughout these sessions, I documented requirements and stakeholder feedback in parallel using multiple tools:

- a Miro virtual whiteboard for collaborative ideation,
- an Enterprise Architect repository for structured modelling,
- and Word exports when formal documentation was required for review or approval.

These materials later formed the basis for detailed user stories and tasks in JIRA, ensuring that the development team had clear, actionable guidance for each increment of work.

9.3. Prototyping & Design Support

To support the design of the new user interface, I prepared conceptual mock-ups illustrating key workflows and interaction patterns. Depending on the context and urgency, prototypes were created using Axure RP, Figma, or (when rapid iteration was needed) by directly modifying HTML / CSS in existing UI components and capturing screenshots of the adjusted views.

These prototypes were reviewed iteratively with stakeholders, who provided feedback incorporated into subsequent versions. This approach accelerated front-end development by giving developers clear, validated guidelines and reducing the need for rework. The mock-ups also supported roadmap planning, helping the client prioritize which UI components and features should be delivered first.

9.4. Project Coordination & Delivery Support

Beyond analytical duties, I actively supported project coordination. During periods when the Project Manager was unavailable, I temporarily assumed PM responsibilities: running daily meetings, aligning timelines, monitoring progress, and ensuring smooth communication between

the Client and the development team. I also contributed to estimating change requests, assessing risks, and planning upcoming increments of work.

This involvement helped maintain project continuity and ensured that the team remained synchronized during critical phases.

9.5. Testing & Quality Assurance

I served as the primary manual tester on the software house side, conducting extensive functional and regression testing across multiple modules. My deep understanding of the system's logic allowed me to identify issues early and verify fixes efficiently. This contributed directly to the stability and reliability of the delivered solution.

I also supported the Client during UAT, helping validate the system's readiness for production and ensuring that reported issues were triaged and resolved promptly.

9.6. Knowledge Management & Documentation

I contributed to building a structured knowledge base for the Operator by documenting key aspects of the system in Confluence. This included functional descriptions, integration details, and operational guidelines. Developers complemented this with technical documentation maintained directly in the application repository, ensuring long-term maintainability and transparency.

9.7. Ongoing Support & Verification of Incoming Requests

Following the deployment of the new Umbrella NMS, I continued to support the project by verifying incoming requests and ensuring they were correctly classified. This included distinguishing between genuine defects and proposals for new functionality.

- For defects, I reproduced and validated the reported behaviour, forwarded confirmed issues to developers, and later tested the implemented fixes.
- For new feature requests, I discussed expectations with stakeholders, proposed feasible solutions, clarified business rules, and prepared preliminary effort estimates.

This ongoing involvement helped maintain system quality, ensured that stakeholder needs were addressed efficiently, and supported the continuous evolution of the platform.

10. Lessons Learned

The modernization of the Umbrella NMS platform was a complex, multi-year initiative that surfaced numerous technical, organizational, and process-related challenges. These experiences provided valuable insights into large-scale system redesign, integration with legacy environments, and cross-team collaboration. The lessons learned below summarize the most significant risks encountered during the project and the strategies used to address them.

10.1. Technical & Architectural Lessons

One of the most impactful lessons concerned the unpredictability of technical constraints that emerge only during real-world implementation. Several architectural assumptions made early in the project required re-evaluation as the system evolved:

- Indexing strategy in ELK: Initially, a single index was used for both active and cleared alarms. This quickly led to performance degradation and excessive index growth. Splitting the index into two separate structures significantly improved query performance and system responsiveness.
- Hazelcast → Redis migration: Hazelcast, originally used for generating unique alarm identifiers, caused instability after service or cluster restarts. The migration to Redis (performed mid-project) proved essential for ensuring consistent recovery and preventing alarm loss.
- Hardware failures during data migration: Unexpected disk failures on the Operator's infrastructure slowed down development and highlighted the importance of proactive monitoring and redundancy, even in pre-production environments.
- High-volume data migration: Repeated synchronization of historical alarm data between the legacy and new systems was more demanding than anticipated. The need to maintain parity during parallel operation required careful planning, incremental migration, and validation.
- Connector integration discrepancies: Differences between alarm counts in the old and new systems often stemmed not from defects but from improved deduplication logic or differences in filter syntax. This reinforced the need for transparent communication with stakeholders and clear documentation of behavioural changes.

These challenges underscored the importance of flexibility in architectural decision-making and the need to revisit assumptions as the system matures.

10.2. Development & Delivery Lessons

The project revealed several insights related to implementation strategy and delivery management:

- Correlation engine complexity: Legacy correlation rules did not behave identically in the new engine. Some rules required optimization, and loop-detection mechanisms had to be introduced to prevent recursive triggering.
- Core service split: The initial assumption that a single Core service could handle both front-end and back-end responsibilities proved inefficient. Splitting the service improved maintainability and reduced coupling.
- Legacy logic migration: Some behaviours implemented as ad-hoc scripts in the old system were poorly documented and not well understood. Re-implementing them as structured logic inside dedicated services required close collaboration with stakeholders.
- Parallel operation of old and new systems: Maintaining synchronization between both platforms introduced additional complexity, especially when external NMS systems behaved inconsistently under dual load.

- Automated testing gaps: Automated tests created early in the project became outdated as the system evolved. The absence of a dedicated automation tester in later phases increased reliance on manual testing and highlighted the need for continuous test maintenance.

These experiences reinforced the value of iterative refinement, early alignment between front-end and back-end teams, and sustained investment in testing throughout the project lifecycle.

10.3. Collaboration & Stakeholder Lessons

The project demonstrated how stakeholder dynamics can evolve over time:

- Expanding integration scope: The original plan included only a limited set of external NMS integrations. As the project timeline extended, additional systems had to be onboarded, increasing complexity and requiring adjustments to performance expectations.
- Late involvement of NOC: Although NOC Operators joined primarily during UAT, their feedback was essential. Their late entry extended the testing and deployment phases by several months, highlighting the importance of involving end users earlier in the process.
- Changing stakeholder landscape: The Operator's internal restructuring introduced new teams and responsibilities, requiring additional knowledge transfer and alignment.

These challenges emphasized the importance of continuous communication, early engagement of all user groups, and adaptability to organizational change.

10.4. Process & Project Management Lessons

Several process-related insights emerged during the project:

- Front-end and back-end should evolve together: Developing both layers independently led to mismatches in data models and additional integration work. Future projects should ensure tighter coordination between UI and API development.
- Unexpected scope expansion: New requirements, such as migrating correlation and input rules, were initially assumed to be handled by the Client. Their eventual inclusion required re-planning and reinforced the need for clear ownership of configuration-heavy components.
- Team members availability: Periods of limited availability of the Project Manager or the Automation Tester shifted coordination duties onto the Analyst, which in turn reduced the time available for analysis and testing. This underscored how essential consistent project management and testing capacity are for maintaining delivery efficiency.

These lessons underline the need for realistic planning, cross-functional alignment, and proactive risk management.

10.5. Quality & Testing Lessons

Ensuring quality in a system of this scale brought its own challenges:

- Comparisons with the legacy system: Users often treated the behaviour of the old system (even when incorrect) as the reference point. This required careful explanation, evidence-based validation, and patience in managing expectations.
- Correlation rule testing: Many correlation rules shared similar logic, leading to multiple tickets for what was essentially the same underlying issue. Consolidating these findings improved efficiency.
- Balancing multiple roles: Acting simultaneously as Analyst, Tester, and occasional PM required disciplined time management and highlighted the value of dedicated QA resources during peak phases.

These experiences reinforced the importance of clear communication, structured testing, and realistic workload distribution.

10.6. Organizational & Operational Lessons

The project also revealed broader organizational insights:

- Operator restructuring: The acquisition of another Operator expanded the user base and increased the number of NOC teams relying on the system. This required additional alignment and training.
- New OSS Maintenance Team: The newly formed team had limited knowledge of the legacy system, requiring extensive transfer of both technical and business knowledge.

These lessons highlighted the importance of structured onboarding and documentation when introducing new teams to complex systems.

10.7. Security & Compliance Lessons

Migrating legacy logic and introducing modern security mechanisms surfaced several challenges:

- Authentication and authorization changes: External systems consuming Umbrella NMS data had to adapt to new authentication requirements, which required coordination and communication.
- Legacy script migration: Some scripts were outdated or poorly understood. Decisions had to be made about which logic to preserve, which to redesign, and which to retire.

These experiences reinforced the need for early security planning and clear documentation of legacy behaviours.

10.8. Personal Lessons

From a personal perspective, the project provided valuable growth opportunities:

- Interdisciplinary work: Balancing roles across analysis, testing, and coordination strengthened time-management skills and adaptability.

- Learning new technologies: Exposure to rule engines, NoSQL databases, and distributed architectures expanded my technical toolkit.
- Communication as a cornerstone: Repeatedly validating assumptions, visualizing ideas, and asking clarifying questions proved essential for maintaining alignment.
- What I would do differently: I would ensure closer alignment between front-end and back-end development from the start, maintain automated tests throughout the project, and secure additional QA support during peak analytical phases.

11. Summary

This case study not only documents the modernization of the Umbrella NMS platform but also illustrates the analytical approach, methods, and practices applied throughout the project. By presenting the full lifecycle: from initial bidding and requirements discovery to architectural design, implementation, testing, and post-deployment support, it demonstrates how structured analysis, iterative collaboration, and disciplined coordination can drive a complex transformation effort to a successful outcome.

The redesigned Umbrella NMS evolved into a scalable, modular, and resilient platform capable of handling dramatically higher alarm volumes, recovering quickly from failures, and integrating new NMS systems with minimal effort. The introduction of microservices, asynchronous messaging, dedicated processing engines, and enhanced observability significantly improved system performance and maintainability. At the same time, user-facing enhancements such as dark mode, map visualization, reporting, and configurable notifications delivered tangible improvements to NOC operations.

Delivering this solution required navigating numerous technical and organizational challenges, including data migration, connector discrepancies, correlation rule optimization, and shifting stakeholder expectations. Through structured workshops, user-journey analysis, prototyping, and continuous refinement, the team ensured that the final system aligned with real operational workflows while remaining adaptable to future needs. The incremental rollout strategy, combined with rigorous testing and strong cross-team communication, enabled a smooth transition from the legacy platform.

The project also strengthened the Operator's internal capabilities by consolidating documentation, establishing a new OSS Maintenance Team, and improving knowledge transfer across departments. As a result, the modernized Umbrella NMS provides a robust foundation for ongoing network expansion, operational excellence, and long-term system evolution.

From an analytical perspective, this case study highlights the value of clear requirements, collaborative design, iterative prototyping, and hands-on involvement across multiple roles. It reflects how a Business & Systems Analyst can contribute not only to defining the solution but also to ensuring its quality, coherence, and alignment with stakeholder needs. Ultimately, the initiative demonstrates how thoughtful architecture, disciplined analysis, and coordinated execution can deliver lasting value in complex enterprise environments.

12. Attachments

12.1. List of Figures

Figure 1 Power-Interest Matrix for Project Stakeholders 8
Figure 2 High-Level Alarm Processing Workflow in the modernized Umbrella NMS..... 11
Figure 3 High-level UML Class Diagram with Selected Classes, Attributes & Methods 16
Figure 4 State Machine Diagram with the High-level Lifecycle of an Alarm 18
Figure 5 Diagram of Sample Use Cases for the Umbrella NMS 19
Figure 6 Conceptual mock-up of the Umbrella NMS interface in light mode (the proposed layout for alarm monitoring, filtering, and Operator workflows) 29
Figure 7 Conceptual mock-up of the Umbrella NMS interface in dark mode (support night-shift operations and improve Operator comfort) 29

12.2. List of Tables

Table 1 Stakeholders Roles & Responsibilities 7
Table 2 Alarm Ingestion & Processing Requirements – Examples 13
Table 3 User Interaction Requirements – Examples 13
Table 4 Administration & Configuration Requirements – Examples 14
Table 5 Performance & Scalability Requirements – Examples 14
Table 6 Reliability & Resilience Requirements – Examples 15
Table 7 Security Requirements – Examples 15
Table 8 Maintainability Requirements – Examples 15
Table 9 Use Case Mapping Matrix for Functional Requirements 20
Table 10 User Stories – Examples (Based on Earlier Functional Requirements) 25
Table 11 Example Test Case Scenarios (Based on Earlier Functional Requirements) 27
Table 12 Traceability Matrix (Consolidated View) 28

