Ref. Ares(2025)11235472 - 16/12/2025

NARRATE

Regenerative Resilient Smart Manufacturing Networks

D3.3 SMN Knowledge Model using a Neurosymbolic DSS (a)

2025/12/15

Grant Agreement No.: 101138094
Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01
Type of action: HORIZON-RIA

D3.3 SMN KNOWLEDGE MODEL USING A NEUROSYMBOLIC DSS NEUROSYMBOLIC DSS (A)

Work package	WP 3
Task	T3.3
Due date	15/12/2025
Submission date	15/12/2025
Deliverable lead	NUNSYS
Version	1.0
Authors	Tomás González, David Correa, Jesús Ávila (NUN)
Reviewers	Jan Torka (FHG), Michael Papazoglou (SERV)
Abstract	The deliverable presents the design and implementation of the Neuro-Symbolic Question Answering (NSQA) Decision Support System within the IMC platform. This system integrates neural language understanding with symbolic reasoning to provide explainable, adaptive, and auditable decision-making for risk management scenarios. The architecture combines semantic parsing, expert rules, static and dynamic response plans, and external data retrieval through REACT-based agents. By merging neural flexibility with symbolic structure, the NSQA module ensures robust interpretation of heterogeneous inputs, transparent reasoning chains, and continual learning through dynamic knowledge creation. This hybrid approach enhances operational intelligence, scalability, and compliance in complex industrial environments.
Keywords	Neuro-Symbolic AI, Decision Support System, Semantic Parsing, Knowledge Representation, Risk Management,

	Explainable AI, REACT Agents, Dynamic Response Plans, Hybrid Reasoning, IMC Platform.

Project co-funded by the European Commission in the Horizon Europe Programme

Nature of the deliverable:	R
Nature of the deliverable:	PU

DOCUMENT REVISION HISTORY

Version	Date	Description of change	List of contributor(s)
0.1	2025/12/05	First Draft	NUN
0.2	2025/12/08	Review	NUN
0.3	2025/12/11	Review	SERV
0.4	2025/12/11	Review	FhG
0.9	2025/12/12	Review	NUN
1.0	2025/12/15	Final review by coordination	AID

STATEMENT ON MAINSTREAMING GENDER

The NARRATE consortium is committed to including gender and intersectionality as a transversal aspect in the project’s activities. In line with EU guidelines and objectives, all partners – including the authors of this deliverable – recognise the importance of advancing gender analysis and sex-disaggregated data collection in the development of scientific research. Therefore, we commit to paying particular attention to including, monitoring, and periodically evaluating the participation of different genders in all activities developed within the project, including workshops, webinars and events but also surveys, interviews and research, in general. While applying a non-binary approach to data collection and promoting the participation of all genders in the activities, the partners will periodically reflect and inform about the limitations of their approach. Through an iterative learning process, they commit to plan and implement strategies that maximise the inclusion of more intersectional perspectives in their activities.

DISCLAIMER

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Climate, Infrastructure and Environment Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

COPYRIGHT NOTICE

© NARRATE Consortium, 2025 - All rights reserved

No part of this publication may be translated, reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the written permission of the publisher or provided the source is acknowledged.

Except otherwise noted, the reuse of this document is authorised under the Creative Commons Attribution 4.0 International (CC BY 4.0) licence. This means that reuse is allowed provided appropriate credit is given and any changes are indicated.

How to cite this report: NARRATE(2025). D3.3 SMN KNOWLEDGE MODEL USING A NEUROSYMBOLIC DSS (A)

The NARRATE Consortium is the following:

Participant number	Participant organisation name	Short name	Country
1	INSTITUTO TECNOLOGICO METALMECANICO, MUEBLE, MADERA, EMBALAJE Y AFINES-AIDIMME	AID	ES
2	SCIENTIFIC ACADEMY FOR SERVICE TECHNOLOGY EV	SERV	DE
3	FRAUNHOFER GESELLSCHAFT ZUR FORDERUNG DER ANGEWANDTEN FORSCHUNG EV	FhG	DE
4	INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON	INSA	FR
4.1	INSAVALOR SA	INSA-V	FR
5	SOFTWARE AG	SAG	DE
6	F6S NETWORK IRELAND LIMITED	F6S	IE
7	SYNESIS-SOCIETA CONSORTILE A RESPONSABILITA LIMITATA	SYN	IT
8	MEDITERRANEAN WOOD FACTORY S.L. - MEDWOOD	MED	ES
9	DHL EXEL SUPPLY CHAIN SPAIN SL	DHL	ES
10	NUNSYS SA	NUN	ES
11	BUDATEC GMBH	BUD	DE
12	BRUNEL UNIVERSITY LONDON	BUL	UK
13	POLICY LAB OU	PL	EE
14	BLUEBRIDGESOLUTIONS UG	BLU	DE
16	SANTER REPLY SPA	REP	IT

TABLE OF CONTENTS

Table of Contents 5

List of figures 7

List of tables 7

Abbreviations 8

1. Introduction 11

2. Theoretical background 12

2.1 Neuro-symbolic artificial intelligence 12

2.1.1. Neural Components: Learning and Interpretation 13

2.1.2. Symbolic Components: Structure, Logic, and Explainability 13

2.1.3. Complementarity Between Neural and Symbolic Layers 14

2.2. Knowledge base question answering 14

2.2.1. Structured and Explicit Knowledge Representation 14

2.2.2. From Natural-Language Input to Knowledge-Based Interpretation 15

2.2.3. Reasoning Over Knowledge and Facts 15

2.2.4. Integration of External Information Sources 15

2.3. Semantic parsing 17

2.3.1 Interpreting Natural-Language Risk Descriptions 17

2.3.2 Mapping Inputs to Risk Types 17

2.3.3 Structuring Inputs for Symbolic Reasoning 18

2.3.4 Supporting Response Plan Selection and Initialization 18

2.3.5 Enhancing Explainability 18

2.4. React agents and stepwise reasoning 18

2.4.1 Relationship with Semantic Parsing 19

2.4.2 Relationship with the Knowledge Base 21

2.4.3 Core Principles of REACT Architectures 22

2.4.4 Benefits for Neuro-Symbolic Reasoning 22

2.4.5 REACT Reasoning Cycles in the NSQA Module 23

2.4.6 Interaction with Response Plans and Expert Rules 23

2.4.7 Natural-Language Reasoning Trace 24

3. Conceptual framework of the nsqa module 24

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

3.1. Interpretation Layer 25
3.2. Knowledge Layer 25
3.3. Reasoning Layer........ 26
3.4. Learning Layer 26
4. Overall system architecture........ 27
4.1. High-level overview 27
4.2. Component roles 27
4.3. Data and knowledge flow........ 28
4.4 Architecture diagram........ 29
5. Knowledge representation and management........ 30
5.1. Expert rules........ 31
5.2. Static response plans........ 31
5.3. Dynamic response plans........ 32
5.4. Facts and external knowledge sources........ 33
5.4.1. Nature of Facts........ 34
5.4.2. Sources of Facts........ 34
5.4.3. Integration of Facts into the Reasoning Process 34
5.4.4. Fact Persistence and Scope 35
6. NSQA processing pipeline 35
6.1. Input modalities........ 35
6.1.1. User Natural-Language Queries 36
6.1.2. Events Received via RabbitMQ........ 36
6.2. Risk type identification 37
6.3. Response plan selection 38
6.4. Rule integration 39
6.5. React agent construction and execution........ 39
6.6. Output 40
6.7. dynamic knowledge creation 40
7. Explainability and traceability........ 41
8. Examples and case studies........ 42
9. Conclusion........ 73

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation,
Horizon Europe Guarantee.

10. Appendix..74
10.1. Graph Data for the use cases..74
11. References..76

LIST OF FIGURES

Figure 1. Data and knowledge flow......................................29
Figure 2. Architecture diagram..30
Figure 3. Monitoring Platform main page to SUBMIT A QUERY In natural language............36
Figure 4. User submitting a query in natural and getting a result.......37
Figure 5.Graph data for the use cases...................................75

LIST OF TABLES

Table 1. Component roles..28

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
API Agents	Agents for accessing external APIs (OpenAPI-basierte Agenten)
BB#x	Building Block # Number (Refers to deliverable D1.4 ARCHITECTURAL REQUIREMENTS[14], tools clustering organization)
CC BY 4.0	Creative Commons Attribution 4.0 International License
DSS	Decision Support System
EC	European Commission
FSC	Forest Stewardship Council
Graph DB	Graph Database
HORIZON-RIA	Horizon Research and Innovation Action
IMC	Intelligent Manufacturing Custodian
IMC Platform	Intelligent Manufacturing Custodian Platform
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
KBQA	Knowledge Base Question Answering
KBGQA	Knowledge Base Question Answering
KPI	Key Performance Indicator
LLM	Large Language Model
LLM-based	Large-Language-Model-based (e.g. LLM-based classifier)
MSx	Milestone
Mx	Month x
NSAI	Neuro-Symbolic Artificial Intelligence
NSQA	Neuro-Symbolic Question Answering
OPENAPI	Open API Specification for RESTful services
PEFC	Programme for the Endorsement of Forest Certification
RAG	Retrieval-Augmented Generation
RAG Agents	Retrieval-Augmented Generation Agents

RAG DB	Retrieval-Augmented Generation Database
REACT	Reason + Act (Hybrid reasoning agent architecture)
REST	Representational State Transfer
RabbitMQ	Message-broker system for event-based communication
Tx.x	Task x.x
SQL	Structured Query Language
SQL DB	SQL Database

EXECUTIVE SUMMARY

This deliverable introduces the Neuro-Symbolic Question Answering (NSQA) Decision Support System developed for the IMC platform, designed to enhance risk management through explainable and adaptive reasoning. Modern industrial environments demand decision-making systems capable of interpreting natural-language inputs, integrating expert knowledge, and reasoning transparently. The NSQA module addresses these needs by combining **neural components** for semantic understanding with **symbolic structures** for rule-based consistency and procedural logic.

The architecture comprises four layers: **Interpretation**, **Knowledge**, **Reasoning**, and **Learning**. It processes heterogeneous inputs—user queries and real-time events—through semantic parsing and risk classification, then applies expert rules and response plans to guide decisions. A REACT-based agent orchestrates stepwise reasoning, integrates external data sources and generates auditable reasoning traces. When predefined knowledge is insufficient, the system dynamically creates new risk types and response plans, validated through user feedback, ensuring continual learning and scalability.

Key outcomes include:

- **Explainable decision-making** through transparent reasoning chains.
- **Hybrid neuro-symbolic integration** for robust interpretation and structured logic.
- **Dynamic knowledge creation** enabling adaptability to novel scenarios.
- Integration with external data sources for real-time contextualization.

By merging neural flexibility with symbolic rigor, the NSQA module provides a foundation for **advanced**, **compliant**, and **operationally intelligent decision support** within IMC. Its design emphasizes **transparency**, **modularity**, and **scalability**, positioning the platform for future extensions in industrial risk management.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

1. INTRODUCTION

The Neuro-Symbolic Question Answering (NSQA) Decision Support System developed for the IMC platform constitutes a hybrid artificial intelligence framework designed to provide robust, explainable, and adaptive reasoning capabilities for risk management scenarios. Modern industrial and operational environments frequently involve complex, dynamic, and heterogeneous sources of information. These environments require decision-making systems that can not only interpret natural-language descriptions of risks but also apply expert knowledge, retrieve factual external data, and reason transparently about the consequences of actions. The NSQA module addresses this need by integrating **neural** and **symbolic** AI components[1]: two complementary paradigms that, when combined, yield a system capable of both deep linguistic understanding and structured, auditable reasoning.

The IMC platform serves as the application ecosystem where risk information is continuously generated. Risks may originate from two primary channels:

1. **Natural-language queries provided by human users**, describing incidents, anomalies, warnings, or operational questions.
2. **Events captured automatically through a RabbitMQ-based message bus**, representing real-time signals from sensors, monitoring systems, or integrated digital services.

Regardless of their origin, these inputs often exhibit ambiguity, incomplete information, or diverse linguistic formulations. The NSQA module must therefore interpret them semantically, classify them appropriately, and decide which response strategy (or *response plan*) is best suited to handle the situation. This decision-making process requires both **contextual understanding** (a strength of neural models such as large language models) and **rule-based consistency**, provided by symbolic knowledge such as expert-defined rules and procedural response plans.

Therefore, the NSQA system is designed as a **multi-layered architecture** incorporating:

- **Neural understanding**, responsible for semantic matching, intent identification, and natural-language interpretation[5].
- **Symbolic reasoning**, based on explicit expert knowledge, stored rules, and structured procedural logic in response plans.
- **External factual grounding**, achieved through agents capable of querying APIs, RAG-based vector stores, and other information retrieval tools.
- **Iterative decision-making**, implemented through a REACT-based reasoning agent that deliberates step by step, justifying its actions and reflections at every stage.
- **Dynamic learning mechanisms**, enabling the creation of new risk types and response plans when predefined knowledge is not sufficient.

This hybrid neuro-symbolic formulation is vital for a domain such as risk management, where decisions must often be justified, traceable, and coherent with organizational policies. Purely neural models, while powerful in understanding language, often lack the consistency, determinism, and explicability required in regulated or safety-critical environments. Conversely, purely symbolic systems are rigid and unable to generalize when confronted with novel risk descriptions, unexpected linguistic variations, or emerging operational contexts. By merging both paradigms, the NSQA module leverages neural models' adaptability without sacrificing the structural transparency of

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

symbolic reasoning frameworks.

Within the IMC ecosystem, the NSQA module plays a central role in supporting informed decision-making processes. When a risk is received, the system must identify its most relevant semantic category (*risk type*), select (or generate) an appropriate response plan, retrieve any necessary external information, and reason through the sequence of actions required to resolve. The outcome is not simply a final answer or recommendation; it includes a **detailed, human-readable explanation** of why each step was taken, which rules or facts influenced the decision, and what external knowledge was consulted in the process.

An additional characteristic of the NSQA architecture is its **capacity for continual learning**. If a new type of risk arises for which no predefined knowledge exists, the system can generate a new risk type and construct a dynamic response plan based on the reasoning process executed by the REACT agent. These dynamically generated plans can then be validated through user feedback—strengthening the system with new expert knowledge when appropriate or discarding inadequate strategies. This mechanism progressively enriches the knowledge base, allowing the system to evolve alongside the operational environment it supports.

Ultimately, the NSQA Decision Support System aims to provide the IMC platform with:

- Semantic understanding of natural-language risks
- Explainable, auditable reasoning chains
- Integration with external knowledge sources
- Procedural consistency through symbolic plans
- Adaptability through dynamic knowledge creation
- Scalability and maintainability of expert knowledge

By combining these capabilities, the NSQA module establishes a foundation for advanced decision-support functionalities in IMC, enabling the platform to address both routine and novel risk scenarios with transparency, rigor, and operational intelligence.

2. THEORETICAL BACKGROUND

This section provides the theoretical foundations that guide the design of the NSQA module and its role within the IMC platform. It introduces key concepts from neuro-symbolic artificial intelligence, knowledge base question answering, semantic parsing, and stepwise agent-based reasoning, and explains why their combination is particularly suitable for risk management scenarios. By outlining these principles, the text establishes the conceptual framework that later sections translate into concrete architectural components, processing pipelines, and implementation choices.

2.1 NEURO-SYMBOLIC ARTIFICIAL INTELLIGENCE

Neuro-Symbolic Artificial Intelligence (Neuro-Symbolic AI) refers to a family of computational approaches that integrate **neural learning systems** (e.g., deep learning models and large language models) with **symbolic reasoning frameworks** (e.g., rules, logic programs, structured plans, and knowledge graphs)[12]. The objective of this integration is to achieve systems that can both *understand and represent complex*

Funded by the European Union
Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions
UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

information in flexible ways, and *reason over it* with structured, interpretable logic.[1][2]

Whereas neural models excel at pattern recognition, semantic generalization, and natural-language processing, symbolic systems provide explicit, human-readable structures that facilitate consistency, formal manipulation, safety, and traceability. Neuro-Symbolic AI therefore seeks to combine the strengths of both paradigms into a unified decision-making substrate.

2.1.1. Neural Components: Learning and Interpretation

The neural side of a neuro-symbolic system contributes capabilities such as:

- **Semantic understanding:** mapping natural-language inputs to meaningful internal structures (e.g., identifying relevant risk types from a user query).
- **Contextual inference:** using learned representations to relate concepts that may not be explicitly encoded in symbolic rules.
- **Generalization:** recognizing patterns across many forms of phrasing, events, or domain situations.
- **Adaptive knowledge acquisition:** generating new procedural descriptions or plans based on experience.

In the NSQA module, these capabilities are essential for processing natural-language risks and aligning them with the existing or newly generated symbolic knowledge structures of the IMC system.

2.1.2. Symbolic Components: Structure, Logic, and Explainability

Symbolic systems contribute:

- **Explicit knowledge representation**, such as rules provided by domain experts[11] or predefined response plans.
- **Formal reasoning mechanisms** that can be inspected, validated, and audited.
- **Deterministic behaviour** when applying structured sequences of steps or rule-based decisions.
- **Interpretability**, since symbolic representations can naturally map to human-understandable explanations.

Purely symbolic systems, however, rely on manual construction and maintenance of rules, ontologies, and plans. This makes them:

- Inflexible when encountering new or unforeseen scenarios
- Dependent on exhaustive expert specification
- Challenging to scale as domains evolve or diversify

Neuro-symbolic integration alleviates these issues by allowing symbolic reasoning to remain grounded in structured expert knowledge, while neural models provide the flexibility to interpret new conditions and expand the symbolic layer when required.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

2.1.3. Complementarity Between Neural and Symbolic Layers

The power of Neuro-Symbolic AI[12] emerges from the interaction of these two components[2]:

- Neural models handle ambiguity, variability, and natural language.
- Symbolic structures provide formalized decision pathways, rule enforcement, and traceable logic.
- The system cycles between both representations, continually enriching one with information derived from the other.

In practical terms, this means that complex decision processes—such as interpreting the nature of a risk, determining an appropriate response strategy, querying external sources for missing facts, and communicating the reasoning process—can be managed with both robustness and transparency.

This hybrid approach is central to the NSQA module within IMC. The system uses neural semantic parsing and interpretation to identify or create risk types, while symbolic response plans and expert rules guide the reasoning process in a repeatable and auditable way. The interplay of both layers enables a dynamic yet controlled decision-support engine suitable for real-world operational environments.

2.2. KNOWLEDGE BASE QUESTION ANSWERING

Knowledge Base Question Answering refers to the set of techniques, models, and computational strategies that enable a system to answer questions by reasoning over explicitly represented knowledge. Unlike purely neural approaches, which infer answers implicitly through distributed representations (knowledge-base), question answering relies on structured knowledge repositories, logical rules, and explicit relationships between entities. This paradigm is particularly relevant for decision-support contexts, where answers must be reliable, traceable, and grounded in well-defined domain knowledge.

In the context of the NSQA module, the knowledge base is composed of several complementary elements: expert-defined rules, static response plans (BB#2), dynamically generated response plans, and facts and knowledge obtained through external data sources (for example: blueprints in BB#3 and platform library in BB#5). Together, these form a hybrid knowledge substrate that the system can query, interpret, and manipulate during the reasoning process[6].

2.2.1. Structured and Explicit Knowledge Representation

A defining characteristic of knowledge-base–oriented question answering is the emphasis on explicit structure. Knowledge is stored in forms that can be inspected, validated, and reused. This includes:

- **Rules**, capturing expert insights and domain constraints
- **Response plans**, encoding procedural knowledge for risk resolution
- **Entity–attribute–relation structures**, whether stored as database entries, JSON schemas, or logical predicates

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

- **Factual observations and knowledge**, retrieved on demand from external APIs, semantically indexed sources or blueprint repositories.

This explicit structure enables the system to perform step-by-step reasoning and provide verifiable explanations for each decision it makes.

2.2.2. From Natural-Language Input to Knowledge-Based Interpretation

When a user submits a query or an external event triggers a risk evaluation, the system must bridge the gap between natural language and the symbolic structures described above. Knowledge-based question answering traditionally involves two major steps:

- **Interpretation** of the natural-language input into a form that aligns with the structured knowledge.[5] In our system, this corresponds to identifying the most semantically relevant *risk type*.
- **Retrieval and reasoning** based on the knowledge relevant to that interpretation. This includes selecting an appropriate response plan and integrating applicable expert rules.

The NSQA module enhances this process through a semantic-parsing step powered by large language models, allowing the system to map free-form queries into domain-specific concepts even when the phrasing is varied or incomplete[3][5].

2.2.3. Reasoning Over Knowledge and Facts

Once the input is semantically anchored, the system begins reasoning over the available symbolic structures. Knowledge-base reasoning typically involves:

- Pattern matching against rules
- Evaluating conditions and constraints
- Stepwise execution of response plans
- Dynamic incorporation of new information
- Justification of each decision

A key strength of using a structured knowledge base is that the reasoning process becomes auditable. Every step can be traced to a rule, a plan instruction, or a fact retrieved by agents. This aligns directly with the IMC platform's transparency and explainability requirements.

2.2.4. Integration of External Information Sources

In many decision-support scenarios, answers cannot be derived solely from static knowledge. Real-time or context-specific information must be incorporated. The NSQA module addresses this by treating external agents: OpenAPI-based connectors, retrieval-augmented generation agents, and other information providers like blueprint providers in BB#3 and platform library in BB#5, as extensions of the knowledge base. These agents are explained below in **menu point 5.4**.

Each call to an external source enriches the knowledge state with fresh facts. These facts are then used by the symbolic reasoning engine to:

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

- Validate assumptions
- Check rule applicability
- Refine or adjust response plan execution
- Support data-driven decisions

This integration of dynamic factual information distinguishes the NSQA system from traditional, static knowledge-base approaches.

Hybridization with Neuro-Symbolic Reasoning

Although this section focuses on knowledge-base reasoning, the NSQA system's distinctive strength lies in its integration with neural reasoning components. The structured knowledge facilitates deterministic, explainable decision-making, while the neural layers support flexible interpretation and adaptive plan generation.

Through this hybridization, knowledge-base question answering in the NSQA module becomes:

- **More adaptable** (new risk types and plans can be generated dynamically)
- **More expressive** (natural language inputs can be understood without rigid templates)
- **More flexible** (facts can come from heterogeneous sources)
- **More transparent** (symbolic traces can be preserved and reviewed)

Relevance to the IMC Platform

Within the IMC application, knowledge-base question answering allows the system to:

- Advanced Decision Support.

Combining LLMs with the predictive models of digital twins offers improved decision support. For instance, if a disruption is forecasted, the NDSS/LLM could recommend optimized contingency plans by processing historical data and existing supply chain strategies

- Enhanced Predictive Capabilities.

By combining LLMs' natural language understanding with digital twins' data, predictive analytics can be explained in intuitive ways. For example, an LLM can translate a digital twin's predictive maintenance alert into actionable insights, such as:

"Machine X has a 90% chance of failing within 5 days due to rising temperature anomalies. Schedule maintenance now to prevent downtime."

- Automated Reports and Insights.

LLMs can generate detailed, easy-to-understand reports by synthesizing data from digital twins. They can automate processes like preparing summaries of production efficiency, quality trends, or bottlenecks.

- Respond consistently to recurring risk patterns
- Incorporate domain expertise through explicit rules
- Combine predefined strategies with dynamically learned ones
- Provide traceable, explainable decisions that align with compliance and operational requirements

Funded by the European Union
Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions
UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

- Integrate factual data into symbolic reasoning without compromising structure or auditability

This foundation forms one of the core pillars of the NSQA module and underpins its ability to act as a reliable decision-support component within the broader IMC architecture.

2.3. SEMANTIC PARSING

Semantic parsing is a foundational capability within the NSQA Decision Support System, enabling the transformation of natural-language inputs into structured representations that can be used for symbolic reasoning. Its role extends beyond simple text understanding; it acts as the bridge between human language and the system's internal decision-making logic. By mapping unstructured queries or events to machine-interpretable constructs, semantic parsing[3][4] ensures that the downstream reasoning components operate with precise, contextually grounded information.

In the context of the IMC platform, semantic parsing serves several interconnected functions:

2.3.1 Interpreting Natural-Language Risk Descriptions

Risks may enter the system through two channels:

1. **User-provided queries** expressed in natural language.
2. **Automatically ingested events** obtained through message streams such as RabbitMQ.

These inputs can vary significantly in style, vocabulary, and specificity. Semantic parsing allows the system to normalize and interpret them by identifying:

- The underlying **intent** or problem the input describes
- The relevant **entities**, parameters, or conditions expressed
- Temporal or causal aspects that may influence decision-making
- Implicit information that should be made explicit for reasoning

This interpretation step ensures that the diverse forms of input received by the system can be uniformly processed by the reasoning engine.

2.3.2 Mapping Inputs to Risk Types

A core operational task of the NSQA module is the assignment of each incoming risk to a **risk type**, a semantic category representing a class of related risks. Semantic parsing is essential for this mapping process. It enables the identification of the conceptual meaning of the input and supports the LLM-based classification stage by providing structured representations that can be compared against the existing set of risk types.

If no existing risk type adequately captures the semantics of the input, the system—guided by the parsed representation—can generate a new risk type. This dynamic creation of categories allows the knowledge base to evolve organically as the system encounters novel situations.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

2.3.3 Structuring Inputs for Symbolic Reasoning

Symbolic reasoning requires the manipulation of structured data, such as predicates, attributes, or logical primitives. Semantic parsing transforms free-form text into such structures. While the NSQA architecture does not rely on rigid formal grammars, it utilizes semantic parsing to extract:

- Relevant conditions or constraints
- Candidate actions or procedures implied in the risk
- Relationships between entities (e.g., cause–effect, dependency, containment)
- Indicators that may activate expert rules or response plan steps

This process enhances the compatibility of natural-language inputs with the symbolic components of the system, such as expert rules and response plans.

2.3.4 Supporting Response Plan Selection and Initialization

Response plans—whether static or dynamically generated—contain structured logic, decision points, and procedural steps. Semantic parsing provides the contextual information required to align an incoming risk with the most appropriate plan. This involves:

- Identifying which elements of a plan are relevant
- Initializing plan variables based on parsed attributes
- Determining whether the input contains missing information that will need to be collected through the REACT agent's external calls

This alignment ensures that response plan execution begins with a coherent and well-grounded representation of the problem.

2.3.5 Enhancing Explainability

Explainability is a core principle of the NSQA system. Semantic parsing creates interpretable intermediate representations of the input. These representations can later be referenced when generating natural-language explanations of the system's reasoning process. For example, semantic structures extracted from the input inform the explanations of:

- Why a given risk type was selected
- Why expert rules were triggered
- Why certain external data sources were consulted
- How the final decision relates to the user's original query

By grounding all reasoning steps in a parsed understanding of the initial risk, the system's explanations become more transparent and easier for users to audit.

2.4. REACT AGENTS AND STEPWISE REASONING

REACT agents represent a family of hybrid reasoning frameworks that coordinate *Reflection* (deliberation, internal reasoning) and *Action* (interaction with external tools or

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

information sources). This approach has gained substantial traction in neuro-symbolic settings because it provides a structured interface between neural reasoning capabilities and explicit procedural or symbolic steps. In the context of the NSQA module, REACT agents serve as the principal operational mechanism for executing response plans, incorporating expert rules, and integrating dynamically retrieved facts[7]. This is shown in detail in **menu point 8**

2.4.1 Relationship with Semantic Parsing

Semantic parsing provides the initial structured interpretation of an incoming input, transforming natural-language queries or event descriptions into a representation that can be aligned with the system's symbolic knowledge. This parsed representation directly conditions the behaviour of the REACT agent in several ways:

- It determines the **initial problem framing** that the agent reflects upon during its first reasoning step.
- It provides the semantic anchors used to select the relevant risk type, response plan, and expert rules.
- It identifies missing or implicit information that may need to be acquired through external actions.

In this sense, semantic parsing defines the **starting state of the REACT reasoning loop**. The agent does not reason directly over raw text, but over the structured, semantically grounded interpretation produced by the parsing stage. This ensures that subsequent reasoning steps are aligned with the user's intent and the domain semantics encoded in the knowledge base.

2.4.1.1 Semantic Parsing in Large Language Models

Semantic parsing is traditionally defined as the task of mapping natural language utterances $x \in X$ formal meaning representations $y \in Y$ as logical forms, executable programs, database queries, or structured action specifications[16]. In the context of **Large Language Models (LLMs)**, semantic parsing is not implemented as an explicit symbolic module but instead emerges as a **latent, distributed computation** induced by large-scale pretraining and task conditioning.

Formally, an LLM parameterizes a conditional distribution[16]:

$$p\theta(y \mid x)$$

where y may itself be a structured representation encoded as a token sequence. When y corresponds to a formal or semi-formal structure (e.g., SQL, JSON, API calls), the model implicitly performs semantic parsing.

2.4.1.2 Distributed Semantic Representations

LLMs encode input sequences into **contextualized token embeddings** via stacked self-attention layers. Unlike classical semantic parsers that rely on discrete symbols and grammar rules, LLMs represent meaning in a **high-dimensional continuous space**.

Each token embedding jointly encodes:

- lexical semantics,
- syntactic role,

Funded by the European Union
Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions
UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

- discourse context,
- and pragmatic intent.

Crucially, semantic roles and predicate–argument structures are not explicitly represented but are **implicitly encoded in activation patterns** distributed across layers and attention heads.

2.4.1.3 Semantic Composition via Self-Attention

Semantic composition in LLMs is primarily mediated by the self-attention mechanism[15]. Attention weights induce soft dependency structures that resemble latent parse trees or semantic graphs.

Given a sequence of token representations $H = \{h_1, \dots, h_n\}$, self-attention computes[15]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$$

This operation enables the model to:

- bind modifiers to predicates,
- associate arguments with events,
- resolve long-range dependencies,
- and model scope phenomena (e.g., negation, comparison).

From a semantic parsing perspective, self-attention functions as a **learned composition operator**, replacing explicit grammar-driven composition with data-driven relational induction.

2.4.1.4 Latent Structural Induction

Although LLMs do not construct explicit symbolic trees, empirical analyses have shown that intermediate layers encode information consistent with:

- dependency relations,
- predicate–argument structures,
- entity linking,
- and hierarchical event composition.

These structures are **latent** and **soft**, but they are sufficient to support downstream generation of executable meaning representations. In this sense, LLM-based semantic parsing can be viewed as the induction of **continuous semantic programs**, which are only discretized at the output stage.

2.4.1.5 Decoding as Meaning Realization

Semantic parsing in LLMs becomes explicit during decoding, where the model generates a target sequence that often corresponds to a formal language. The decoding process can be interpreted as a projection[17]:

$$\phi: \mathcal{H}_{\text{semantic}} \rightarrow \mathcal{Y}_{\text{formal}}$$

where $\mathcal{H}_{\text{semantic}}$ denotes the internal semantic state of the model.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

When constrained decoding or function-calling interfaces are used, this projection enforces partial well-formedness, effectively anchoring latent semantics to explicit symbolic structures.

2.4.1.6 Semantic Parsing in REACT Agents: Incremental and Action-Grounded Interpretation

In **REACT agents**, semantic parsing is performed in an explicitly **incremental and action-grounded manner**, tightly coupled with the agent's reasoning and interaction loop. Rather than mapping an input utterance to a complete meaning representation in a single decoding pass, the LLM iteratively refines its semantic interpretation through alternating phases of **reasoning (thought generation)** and **acting (tool invocation or environment interaction)**[7].

Formally, the semantic parsing process in REACT can be modeled as a sequence of latent semantic states:

$$s_0 \xrightarrow{\text{reason}} s_1 \xrightarrow{\text{act}} s_2 \xrightarrow{\text{observe}} \cdots \xrightarrow{\text{reason}} s_T$$

where each state s_t encodes a partial and evolving semantic interpretation of the task. Each reasoning step performs a local semantic refinement (such as disambiguating intents, introducing constraints, or hypothesizing intermediate goals) while each action step grounds the current semantic hypothesis by producing an executable structure (e.g., a tool call with structured arguments).

In this framework, **actions function as explicit semantic commitments**. Selecting an action corresponds to instantiating a partial semantic parse into an executable form, effectively externalizing part of the latent meaning representation. Observations returned by the environment then serve as additional semantic evidence, enabling the agent to revise or extend its interpretation.

From a semantic parsing perspective, REACT agents thus implement a form of **interleaved parsing and execution**, analogous to:

- incremental semantic parsing with environment feedback,
- execution-guided decoding,
- or grounded program induction.

Crucially, semantic correctness in REACT is not evaluated solely at the final output level, but continuously through the success or failure of intermediate actions. This creates a feedback loop in which semantic parsing is **validated, corrected, and refined online**, rather than completed prior to execution.

Overall, REACT agents transform semantic parsing from a static text-to-structure mapping into a **dynamic, interactive process**, where meaning is progressively constructed, grounded, and verified through action in an external environment.

2.4.2 Relationship with the Knowledge Base

The knowledge base, composed of expert rules, response plans, and accumulated facts, provides the **symbolic substrate** over which REACT agents operate. During each reasoning cycle, the agent explicitly consults and manipulates this knowledge:

Funded by the European Union
Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions
UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

- **Response plans** act as procedural guides that inform the agent about expected sequences of actions, decision points, and objectives.
- **Expert rules** provide constraints, priorities, and domain-specific heuristics that shape the agent's choices.
- **Facts**, whether pre-existing or dynamically retrieved, define the current state of the world against which rules and plan steps are evaluated.

The REACT agent thus functions as a **controller that navigates the knowledge base**, deciding which symbolic elements to apply, when to apply them, and how to adapt them based on newly observed information. This tight coupling ensures that neural reasoning remains grounded in explicit, auditable knowledge structures.

2.4.3 Core Principles of REACT Architectures

A REACT architecture is generally characterized by the alternation of two primary phases[7]:

1. **Reason (Reflect):** The agent evaluates the current problem state, considers available knowledge, and determines the optimal next step[8]. This includes:
   - interpreting the current goal and subgoals,
   - referencing symbolic knowledge (rules, response plans),
   - assessing prior observations or incoming data,
   - generating hypotheses or intermediate conclusions,
   - deciding what type of external information may be needed.
2. **Act (Execute):** The agent performs concrete actions that modify its knowledge state or advance the procedure. These actions may include:
   - invoking an external API,
   - retrieving contextual information from a vector store,
   - evaluating or applying a rule,
   - updating facts,
   - interacting with another system component.

The alternation between these phases continues until the agent converges on a resolution to the task, produces a final answer, or reaches a termination criterion. In this sense, REACT agents behave similarly to symbolic problem solvers, but they are guided by a neural reasoning process that generates the internal steps.

2.4.4 Benefits for Neuro-Symbolic Reasoning

The REACT methodology is particularly suited to neuro-symbolic systems for several reasons:

- Structured Decision Paths:

Each reasoning step is explicit and logged, enabling transparency and thorough auditability. This aligns strongly with the goals of explainable decision support in the NSQA system.

Funded by the European Union
Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions
UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

- Interoperability with Symbolic Knowledge:

Response plans, rules, and other symbolic elements can be naturally embedded into the reasoning loop. The agent can reflect on the structure of these artifacts, choose when to apply them, and integrate their constraints into subsequent steps.

- **Incremental Knowledge Accumulation:** REACT agents can continuously incorporate new facts obtained from external calls, allowing the reasoning process to adapt dynamically as more information becomes available.
- **Controlled Task Decomposition:** By reasoning step-by-step, the system can decompose complex risk scenarios into manageable subproblems, which can then be mapped onto symbolic procedures, external queries, or direct logical operations.

This hybrid flow allows neural models to guide the reasoning process while preserving the interpretive advantages of symbolic structures.

2.4.5 REACT Reasoning Cycles in the NSQA Module

In the NSQA setting, each REACT cycle plays a role in solving a risk scenario:

1. **Initial Reflection:** The agent inspects:
    - the selected risk type (or a newly generated one),
    - the associated response plan,
    - expert rules,
    - the initial input description.

From here, the agent determines the first actionable step.

2. **Action Execution:** Typical actions include:
    - querying an OpenAPI agent for real-time data,
    - retrieving contextual documents through RAG agents,
    - checking symbolic constraints from rules,
    - advancing the prescribed steps of a response plan.
3. **Observation Integration:** Results from the action are absorbed back into the system:
    - new facts are added to the local reasoning context,
    - rule conditions may now be satisfied or invalidated,
    - future steps of the response plan may become relevant.
4. **Subsequent Reflection:** The agent re-evaluates the current knowledge state and decides the next best step. This cycle continues iteratively and adaptively.

Because each cycle introduces new knowledge that may influence subsequent decisions, the agent can navigate scenarios in which the initial response plan is incomplete, partially applicable, or requires additional contextualization.

2.4.6 Interaction with Response Plans and Expert Rules

An essential quality of REACT agents in this architecture is their ability to treat symbolic knowledge not as static constraints but as *operational components*. For example:

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

- When a response plan specifies a sequence of steps, the REACT agent does not blindly execute them; instead, it evaluates contextual information at each stage to decide whether:
    - a step should be executed immediately,
    - it should be postponed,
    - it should be modified based on real-time facts,
    - or an alternative step is more suitable.
- For expert rules, the agent may:
    - reflect on which rules apply given the current facts,
    - derive new intermediate conclusions,
    - detect rule conflicts or priority conditions,
    - or refine the decision trajectory based on rule-induced constraints.

This dynamic integration supports a robust neuro-symbolic flow where neural reasoning generates actionable insights, while symbolic structures provide consistency and domain alignment.

2.4.7 Natural-Language Reasoning Trace

One distinctive capability of REACT agents is the automatic generation of a natural-language reasoning trace. For each cycle, the system records:

- the agent's internal reflection,
- the rationale for the next action,
- the action executed,
- the observed outcome,
- and how the outcome changes the problem state.

This trace is central to the explainability of the NSQA module. It allows users to:

- understand why specific external calls were made,
- review how expert rules were applied,
- verify that the system followed (or adapted) the response plan appropriately,
- and assess the overall coherence of the decision.

This transparency is valuable not only for validation but also for constructing dynamic response plans based on the agent's behaviour during execution.

3. CONCEPTUAL FRAMEWORK OF THE NSQA MODULE

The Neuro-Symbolic Question Answering (NSQA) module constitutes the core decision-support capability of the IMC platform. It is designed as an integrated reasoning framework that brings together neural understanding, symbolic knowledge structures, and procedural decision-making into a cohesive operational workflow[6]. The conceptual

Funded by the European Union
Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions
UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

framework presented here outlines the functional layers, their interactions, and the foundational principles that govern the module's behaviour.

At a high level, the NSQA module transforms heterogeneous inputs (natural-language risk descriptions or machine-generated events) into informed and traceable decisions. It achieves this through a structured sequence of interpretation, knowledge retrieval, reasoning, and adaptive learning, implemented through a neuro-symbolic architecture that is both extensible and explainable[2].

The conceptual model comprises **four major layers**, each responsible for a key aspect of the system's intelligence:

1. Interpretation Layer
2. Knowledge Layer
3. Reasoning Layer
4. Learning Layer

These layers interact dynamically during each execution cycle, enabling the module to evaluate risks, identify or construct response strategies, and continually refine its internal knowledge.

3.1. INTERPRETATION LAYER

The Interpretation Layer is responsible for understanding the incoming risk signal and situating it within the domain ontology of the IMC platform. This layer leverages semantic parsing and neural interpretation models to convert user queries or event payloads into structured internal representations.

Key functions include:

- **Risk Understanding:** Interpreting the textual or event-based input and extracting its semantic intent.
- **Risk Type Classification:** Matching the interpreted risk to an existing risk type in the database.
- **Risk Type Creation:** If no semantically appropriate type is found, dynamically creating a new risk type that reflects the conceptual characteristics of the input.

This layer ensures that the subsequent reasoning steps operate on a coherent and domain-relevant representation of the problem, establishing a semantic anchor for all further processing.

3.2. KNOWLEDGE LAYER

The Knowledge Layer functions as the symbolic backbone of the NSQA module. It consolidates and organizes the expert-supplied and system-generated knowledge structures that the reasoning process depends on.

This layer encompasses three primary knowledge components:

1. Expert Rules:

Declarative statements embedded by human experts to encode domain-specific logic, constraints, conditions, or best practices.[11] These rules apply globally across all risks and

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

influence inference throughout the reasoning process.

2. Response Plans:

Structured procedural representations that define how particular categories of risks should be handled.

  - **Static response plans** are authored by experts.
  - **Dynamic response plans** are generated by the system based on prior reasoning experience.

3. **Facts:**

   Contextual information gathered at runtime through external data sources (APIs, databases, vector stores). Facts contribute real-world grounding to the symbolic decision process.

The Knowledge Layer therefore provides both the **domain model** and the **procedural repertoire** that the reasoning engine uses to resolve risks.

3.3. REASONING LAYER

The Reasoning Layer operationalizes the decision-making process. It is orchestrated through a REACT-style agent that integrates both neural and symbolic mechanisms.

This layer performs:

- **Deliberative reasoning:** The agent evaluates which steps or rules are relevant, interprets the evolving situation, and decides on the next reasoning action[7].
- **Action execution:** The agent performs concrete operations, such as calling APIs, retrieving data[10], or applying symbolic constraints.
- **Contextual integration:** New facts, rule implications, and intermediate observations are incorporated into the reasoning state.
- **Trace generation:** Every step is documented in natural language, allowing for transparent post-hoc analysis.

The Reasoning Layer thus combines sequential decision-making with traceable logic, ensuring both effectiveness and explainability.

3.4. LEARNING LAYER

The Learning Layer introduces adaptability into the NSQA module. It allows the system to evolve as it encounters new types of risks or as patterns of reasoning emerge.

Key capabilities include:

- **Risk Type Expansion:** Automatic creation of new risk types when incoming cases do not fit existing categories.
- **Plan Induction:** Conversion of a completed agent execution trace into a reusable response plan, enabling the system to accumulate procedural knowledge over time.
- **Feedback Integration:** Incorporation of user feedback to validate or discard newly generated plans, ensuring quality and domain relevance.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

Through these mechanisms, the Learning Layer closes the loop between experience and knowledge, enabling the NSQA module to become progressively more competent and aligned with real operational needs.

While each layer fulfils a specific role, the NSQA module is fundamentally defined by how the layers interact:

- The **Interpretation Layer** structures the problem.
- The **Knowledge Layer** provides the reasoning substrate.
- The **Reasoning Layer** executes a stepwise solution procedure informed by both neural insights and symbolic knowledge.
- The **Learning Layer** updates the knowledge structures based on outcomes and feedback.

This architecture allows the NSQA system to move fluidly between understanding, reasoning, and learning in a manner consistent with modern neuro-symbolic AI principles.

4. OVERALL SYSTEM ARCHITECTURE

This section presents the overall system architecture of the NSQA module within the IMC platform and explains how its components are integrated into a cohesive decision support workflow. It describes how heterogeneous inputs are processed, transformed into structured knowledge, and combined with neuro-symbolic reasoning mechanisms to deliver explainable and auditable decisions. Furthermore, this section introduces the main building blocks, their roles, and the data and knowledge flows between them, providing a high-level view of how interpretation, knowledge management, reasoning, and learning interact within the system.

4.1. HIGH-LEVEL OVERVIEW

The NSQA system is composed of:

1. Input Processing Component
2. Risk Classification Engine (LLM-based)
3. Knowledge Base (rules, response plans, risk types)
4. Agent Factory (constructs REACT agents)
5. Neuro-Symbolic Reasoning Engine
6. External Data Access Agents
7. Dynamic Learning Module
8. Explanation Generator

4.2. COMPONENT ROLES

The table below summarizes the primary components involved in the NSQA reasoning workflow and their respective responsibilities. Together, these components form a cohesive neuro-symbolic architecture that transforms raw inputs into actionable,

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

explainable decisions. Each module contributes a distinct function, from semantic interpretation to rule application, external data integration, multi-step reasoning, and continuous refinement of knowledge.

Component	Role
Risk Type Classifier	Converts input into domain-specific categories
Response Plan Manager	Retrieves or creates response strategies
Rule Engine	Aggregates expert logic
Agent Factory	Builds REACT orchestration agents
External Agents	Fetch domain data through APIs or RAG
Reasoning Engine	Executes stepwise reasoning
Feedback Module	Validates dynamic knowledge

TABLE 1. COMPONENT ROLES

4.3. DATA AND KNOWLEDGE FLOW

The following diagram illustrates the information flow and processing steps that take place within the **NSQA** module and the Intelligent Manufacturing Custodian (**IMC**). All IMC functionalities that are not related to the NSQA reasoning process or persistent storage are conceptually grouped within the **Orchestrator**, which is responsible for coordinating execution, managing control flow, and interfacing with external components.

The database element shown inside the IMC represents the **Platform Library**, which stores structured knowledge such as risk types, response plans, expert rules, plugins, messages, and execution history. Although depicted within the IMC boundary for clarity, the Platform Library is logically an independent knowledge repository that can be accessed by multiple components.

The diagram also highlights how the NSQA module operates as a specialized reasoning subsystem within the IMC, interacting with the Orchestrator to receive contextual inputs and return structured decisions and explanations. External services, including large language models and intermediate APIs connecting to graph databases, are shown outside the IMC boundary, emphasizing their role as external knowledge and computation providers integrated through well-defined interfaces. This flow is explained in detail in menu point 8.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

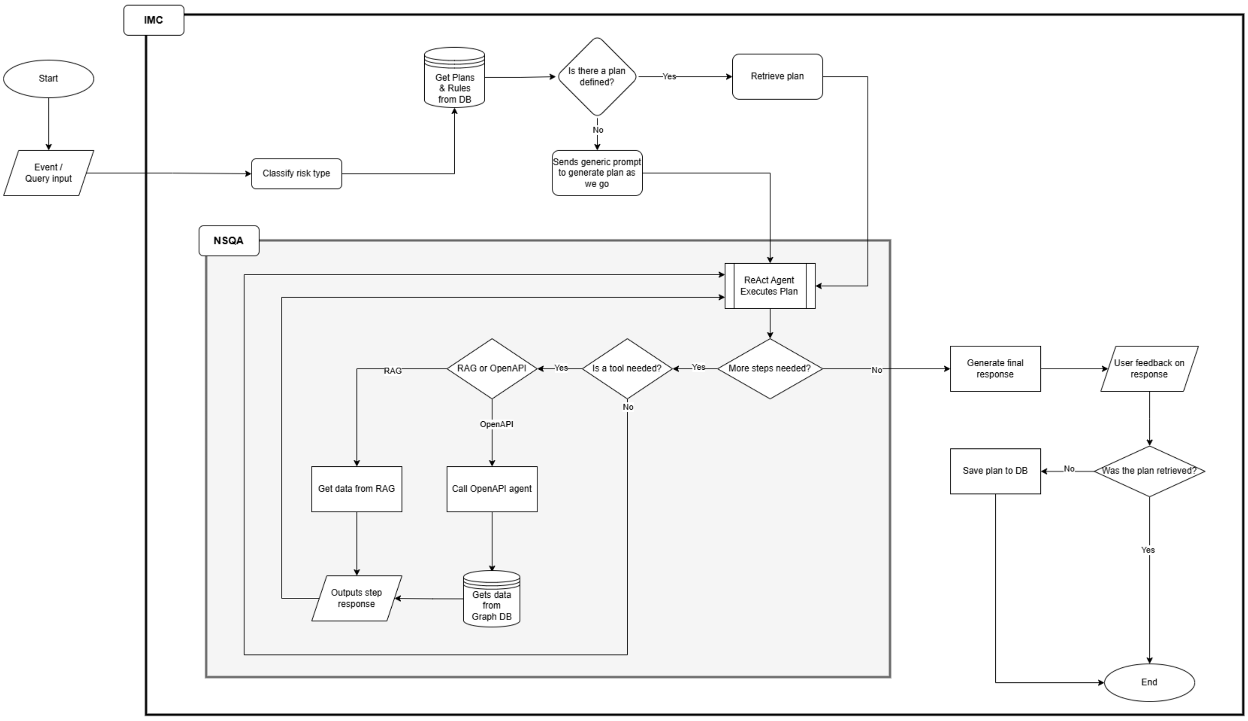


FIGURE 1. DATA AND KNOWLEDGE FLOW

4.4 ARCHITECTURE DIAGRAM

The architecture diagram illustrates how the Intelligent Manufacturing Custodian is internally structured, how it interacts with external services to support neuro-symbolic decision making, and to what building block each element is associated. At the core of the IMC, three principal components cooperate: the database, which stores plans, plugins, rules, messages and other persistent artefacts; the orchestrator, which coordinates reasoning sequences and manages tool invocation; and the NSQA module, which encapsulates the ReAct agent together with its associated tools, such as the RAG retrieval component and the API agent used for external information acquisition. The orchestrator delegates reasoning tasks to the NSQA module, which in turn leverages these tools to gather domain knowledge and execute decision steps. Externally, the IMC communicates with a Large Language Model for semantic interpretation and reasoning support, and with an intermediate API that provides controlled access to a graph database. These connections allow the IMC to enrich its symbolic knowledge, integrate external facts, and maintain a unified flow between data, reasoning and execution across the platform.

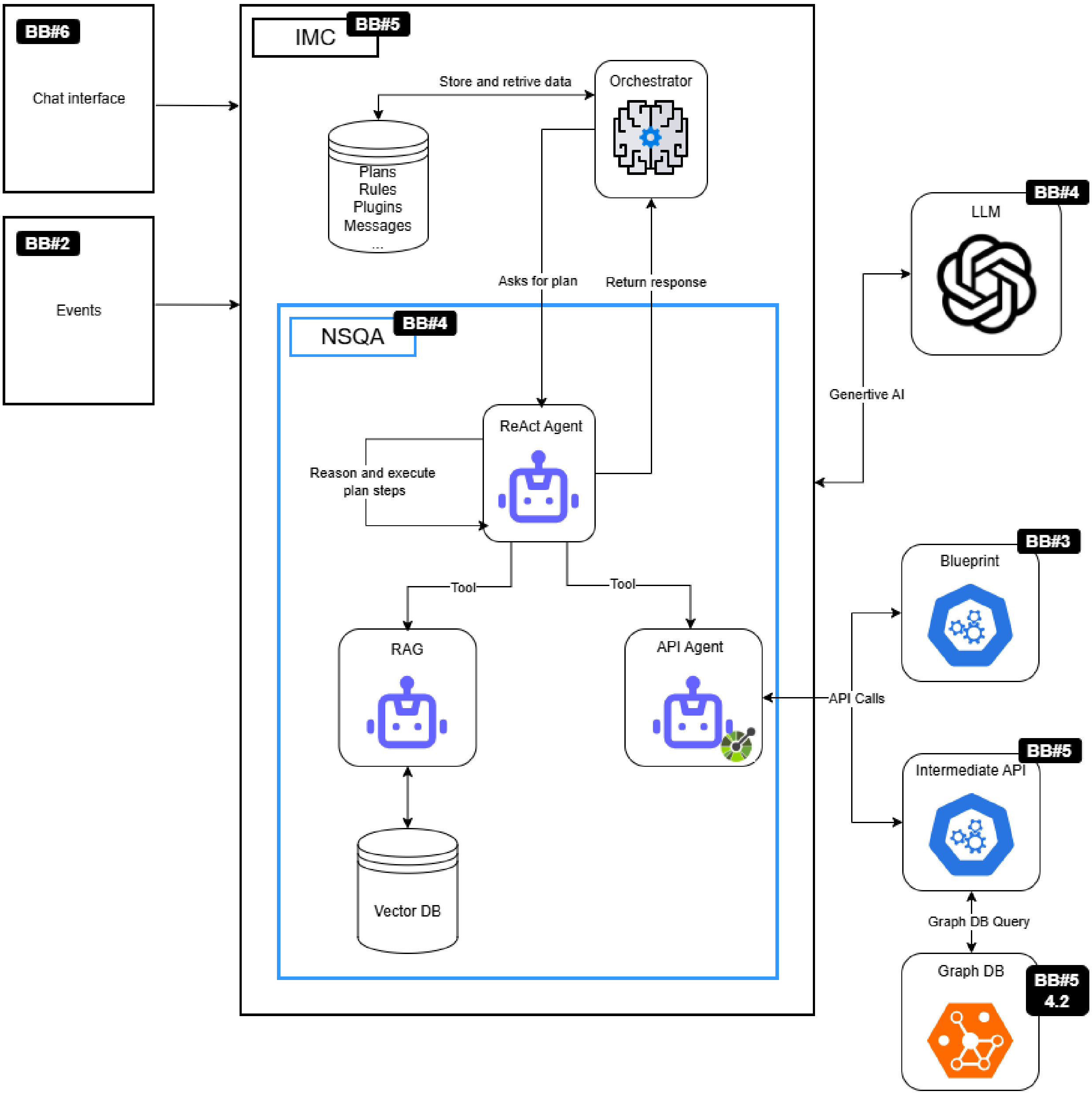


FIGURE 2. ARCHITECTURE DIAGRAM

5. KNOWLEDGE REPRESENTATION AND MANAGEMENT

Chapter 5 introduces the knowledge representation and management layer that underpins the NSQA module's decision-making capabilities. It explains how expert rules, response plans, and factual data are modelled, stored, and maintained so that they can be reliably used by the reasoning processes. The chapter also discusses how static and dynamically generated knowledge are combined, ensuring that the system remains both transparent and adaptable as new risks, contexts, and operational requirements emerge.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

5.1. EXPERT RULES

Expert rules constitute one of the foundational pillars of the symbolic knowledge layer within the NSQA module. These rules encode domain knowledge contributed by human specialists and serve as persistent, globally applicable constraints and guidelines that shape the system's reasoning behaviour. They are stored in natural language within the IMC knowledge base, allowing domain experts to author and maintain them without requiring formal logic programming expertise.

Although expressed in natural language, each rule is interpreted by the system as a declarative statement that influences decision-making throughout the reasoning process. The NSQA engine transforms these statements into an internal representation compatible with its neuro-symbolic reasoning workflow, enabling them to interact seamlessly with response plans, facts retrieved by external agents, and intermediate reasoning steps produced by the REACT agent.

Expert rules fulfil several key functions within the system[11]:

- Add constraints:

Rules may restrict or prioritize certain actions, define conditions that must be met before specific steps are executed, or specify domain requirements that hold across all risks. For example, a rule may state that a particular verification step must always occur before a mitigation action is performed. These constraints help ensure coherence, safety, and alignment with established domain protocols.

- Provide heuristics:

Beyond strict constraints, rules may also encode best practices, heuristics, or preferred strategies that guide the system's reasoning. These heuristics influence how the NSQA module selects information sources, how it structures the execution of a response plan, or how it interprets ambiguous inputs. They enrich the system with expert intuition and serve as a bridge between rigid logic and flexible reasoning.

- Influence plan execution:

During the execution of a response plan, expert rules are continuously evaluated and may modify the system's reasoning trajectory. They can trigger additional steps, override default plan behaviour, or adjust priority among alternative reasoning paths. In this way, rules act as an overlay of expert-driven governance on top of both static response plans and dynamic step-by-step reasoning.

5.2. STATIC RESPONSE PLANS

Static response plans constitute the portion of expert knowledge that has been intentionally and manually designed by domain specialists before system execution. They are structured, predefined procedures that capture established best practices, operational policies, and domain-informed workflows for addressing specific categories of risks. Unlike dynamic response plans—whose structure emerges from the agent's reasoning process—static response plans encode stable, curated expertise that the system can reliably apply across repeated or well-understood scenarios.

A static response plan typically consists of the following components:

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

1. **Execution Steps** Each plan is organized as an ordered sequence of actions or decision points that guide the NSQA agent toward resolving a particular risk. These steps may involve:
    - Verifying preconditions
    - Retrieving or validating relevant information
    - Applying domain rules
    - Taking corrective or preventive actions: Execution steps serve as the backbone of the plan, providing structure and ensuring that the reasoning process unfolds in a logical and domain-consistent manner.
2. **Expected API Calls** Static response plans explicitly reference the external information sources that should be consulted during execution. This may include:
    - OpenAPI-defined endpoints from operational systems
    - Data retrieval mechanisms from partner services
    - Queries to monitoring or asset-management platforms
    - Calls to analytical or scoring services: By specifying expected API interactions, the plan aligns the reasoning process with the organization's available data infrastructure and ensures that decisions are grounded in current, authoritative information.
3. **Domain Reasoning** Each static response plan embeds the expert rationale behind its design. This domain reasoning provides context for why certain steps are required, which conditions matter, and how specific actions contribute to risk mitigation. It reflects:
    - Industry standards or regulatory requirements
    - Organizational policies
    - Historically validated mitigation strategies
    - Best-practice guidelines: Domain reasoning enriches the symbolic component of the NSQA system by making expert knowledge explicit and interpretable. It also supports explainability, as the agent can surface plan-level logic when justifying its decisions to the user.

Conventional response plans are developed in T2.1.

5.3. DYNAMIC RESPONSE PLANS

Dynamic response plans represent the system's ability to **learn structured procedures from its own reasoning traces**. Unlike static response plans— which are manually authored by domain experts— dynamic plans are automatically generated after the NSQA reasoning engine completes a full REACT-style decision cycle for a risk that does not yet have an appropriate predefined plan. They serve as a formalization of the agent's observed behaviour, transforming an implicit reasoning trajectory into an explicit, reusable knowledge artifact.

The creation process begins when the system executes a reasoning sequence without the guidance of a suitable static plan. Throughout this sequence, the REACT agent records every step it takes, including its internal reflections, the external data sources it

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

interacts with, the intermediate facts it generates, and the conditions or rules that influenced its decisions. Once a final resolution to the risk is achieved, the system synthesizes this trace into a structured representation that mirrors the format of static response plans, but with the added benefit of being grounded in real operational experience.

A dynamic response plan typically includes:

- A sequenced set of reasoning and action steps

Each step corresponds to an actionable unit of the agent's previous behaviour— such as querying an external API, invoking a RAG search, applying an expert rule, or evaluating a factual condition. Steps are rewritten into a stable, domain-agnostic form that allows future agents to replicate the logic under similar circumstances.

- Inferred preconditions and contextual triggers

During synthesis, the system analyses which aspects of the input risk, facts, or rule activations led the agent to perform specific actions. These implicit contextual cues are converted into formal preconditions so that the plan becomes operationally applicable in future scenarios.

- Generalized reasoning structures

While the agent's trace may contain internal deliberations specific to a single execution, the system abstracts these into reusable reasoning patterns. For instance, multi-step chains of rule evaluation or iterative queries to an external knowledge source may be transformed into higher-level procedures.

- External data interaction patterns

The system identifies which external endpoints or data retrieval strategies were key to the final decision. These interactions are encoded into the dynamic plan so that subsequent executions can follow the same information-gathering workflow.

- Explanatory metadata

Dynamic plans preserve the system's natural-language explanations, allowing future users to understand not only what the plan does but why its structure emerged. This contributes to transparency and helps domain experts evaluate whether the synthesized plan reflects appropriate domain reasoning.

After creation, dynamic response plans undergo a **validation step**, typically involving human-in-the-loop feedback. Experts or end users evaluate whether the plan is correct, reliable, and suitable for incorporation into the knowledge base. Only after positive validation does the plan become a persistent part of the symbolic layer, where it may operate alongside static plans and influence future decision cycles.

5.4. FACTS AND EXTERNAL KNOWLEDGE SOURCES

Facts represent the contextual, situational, and domain-specific information that the NSQA module retrieves dynamically during its reasoning process. Unlike expert rules or response plans—which encode stable, enduring knowledge—facts correspond to **current states of the world**, gathered through interactions with external systems. These facts provide the empirical grounding required for the neuro-symbolic reasoning process and ensure that decisions remain aligned with real operational conditions rather than relying solely on predefined abstract logic.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

5.4.1. Nature of Facts

Facts may take many forms depending on the type of risk being resolved, including:

- **Current system conditions** (e.g., service status, metrics, configurations)
- **Operational data** retrieved from enterprise APIs
- **Reference knowledge** retrieved from structured or semi-structured sources
- **Unstructured information** retrieved and interpreted through RAG agents
- **Derived facts**, inferred from previous agent steps or from rule activation

Facts are treated as **first-class inputs** within the reasoning process[9]. They can support, trigger, or override certain rule applications, and they often influence the agent's decision about which step to take next.

5.4.2. Sources of Facts

The NSQA module obtains facts through a network of external knowledge connectors, each of which is encapsulated in an LLM-driven agent tailored to a specific retrieval modality. The main categories include:

1. OpenAPI-based Agents

These agents interact with RESTful services described by OpenAPI specifications. They enable the system to query operational systems, monitoring tools, external platforms, or domain services. The retrieved responses are parsed into structured representations and integrated into the reasoning context.

2. Retrieval-Augmented Generation (RAG) Agents

RAG agents enable access to semi-structured or unstructured corpora stored in vector databases[10]. They are beneficial when:

- Domain documentation is extensive.
- Knowledge updates are frequent.
- Information cannot be easily expressed as symbolic rules. The RAG results are converted into natural-language or structured facts usable by the reasoning engine.

3. **Domain-Specific Connectors** Additional agents may interface with:

- Blueprint repository.
- Configuration repositories.
- Knowledge graphs (T4.2/D4.2)
- Monitoring dashboards. These allow the reasoning process to incorporate highly specialized or proprietary data sources.

5.4.3. Integration of Facts into the Reasoning Process

When the REACT meta-agent retrieves a fact, it becomes part of the evolving internal state that guides subsequent decisions:

- Facts can **satisfy conditions** required by response plan steps.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

- They can **activate expert rules** that depend on particular data values.
- They may **shift the direction** of a reasoning trajectory, for example by revealing a new constraint or risk.
- They contribute to the **explanation trace**, making explicit which pieces of information influenced which decisions.

This dynamic interplay between facts and symbolic knowledge is central to the neuro-symbolic paradigm: facts provide situational awareness, while rules and plans provide structure and interpretability.

5.4.4. Fact Persistence and Scope

By design, facts are **contextual** to a specific risk resolution session. They are not permanently stored as part of the knowledge base, ensuring that:

- The reasoning process remains grounded in fresh and relevant data.
- The symbolic knowledge base does not become polluted with ephemeral or time-sensitive information.

However, selected facts—or patterns derived from them—may inspire updates to rules or response plans through expert review or by extending the learning pipeline.

Role of Facts in Ensuring Robustness and Adaptability

The incorporation of external facts ensures that the NSQA module:

- Adapts decisions to real-time circumstances
- Avoids reliance on outdated or static assumptions
- Maintains compatibility with evolving environments
- Leverages organizational knowledge assets without requiring complete symbolic formalization

In this way, facts serve as the connective tissue between the symbolic framework and the operational ecosystem in which the IMC platform functions.

6. NSQA PROCESSING PIPELINE

Chapter 6 describes the end-to-end processing pipeline of the NSQA module, from the moment a risk-related input is received to the generation of a final, explainable output. It outlines how different input modalities are ingested, how risks are classified and matched to response plans, and how rules and agents are orchestrated during stepwise reasoning. The chapter also highlights how intermediate results and dynamically acquired knowledge are integrated at each stage, ensuring that decisions remain context-aware, transparent, and consistent with the IMC platform's overall objectives.

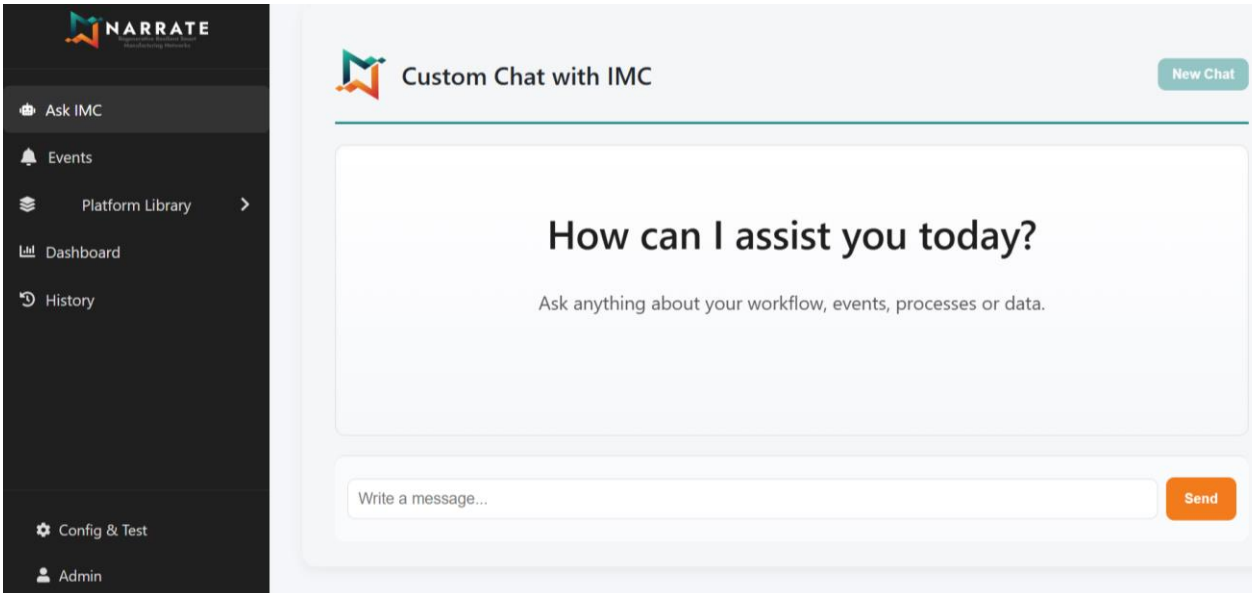
6.1. INPUT MODALITIES

The NSQA module is designed to operate across heterogeneous input channels, ensuring that both human-initiated requests and system-generated events are processed through a unified reasoning pipeline. This flexibility enables the system to support real-time operational monitoring as well as interactive analytical use cases. The primary input modalities are:

6.1.1. User Natural-Language Queries

These queries originate from human users interacting with the IMC application interface. They typically take the form of free-text descriptions of a perceived risk, a question about a system state, or a request for guidance. The NSQA module treats these inputs as semantically rich but structurally unbounded expressions. Upon reception, the system initiates semantic parsing to extract intent, contextual cues, and relevant domain concepts. This modality emphasizes interpretability and natural-language understanding, leveraging neural components to bridge the gap between informal user phrasing and the structured reasoning elements used internally.

The IMC application interface look & feel, named monitoring platform (BB#6), can be seen deeply in deliverable D6.1 AI Platform & Intelligent Manufacturing Custodian Early Release (R1) [13]. This Interface has been created for the users to have an Interaction and administration point with the IMC platform where users can Interact with the IMC via chat, can see the history of questions, all types of events and their resolution, risks, response plans, etc.



Custom Chat with IMC

How can I assist you today?

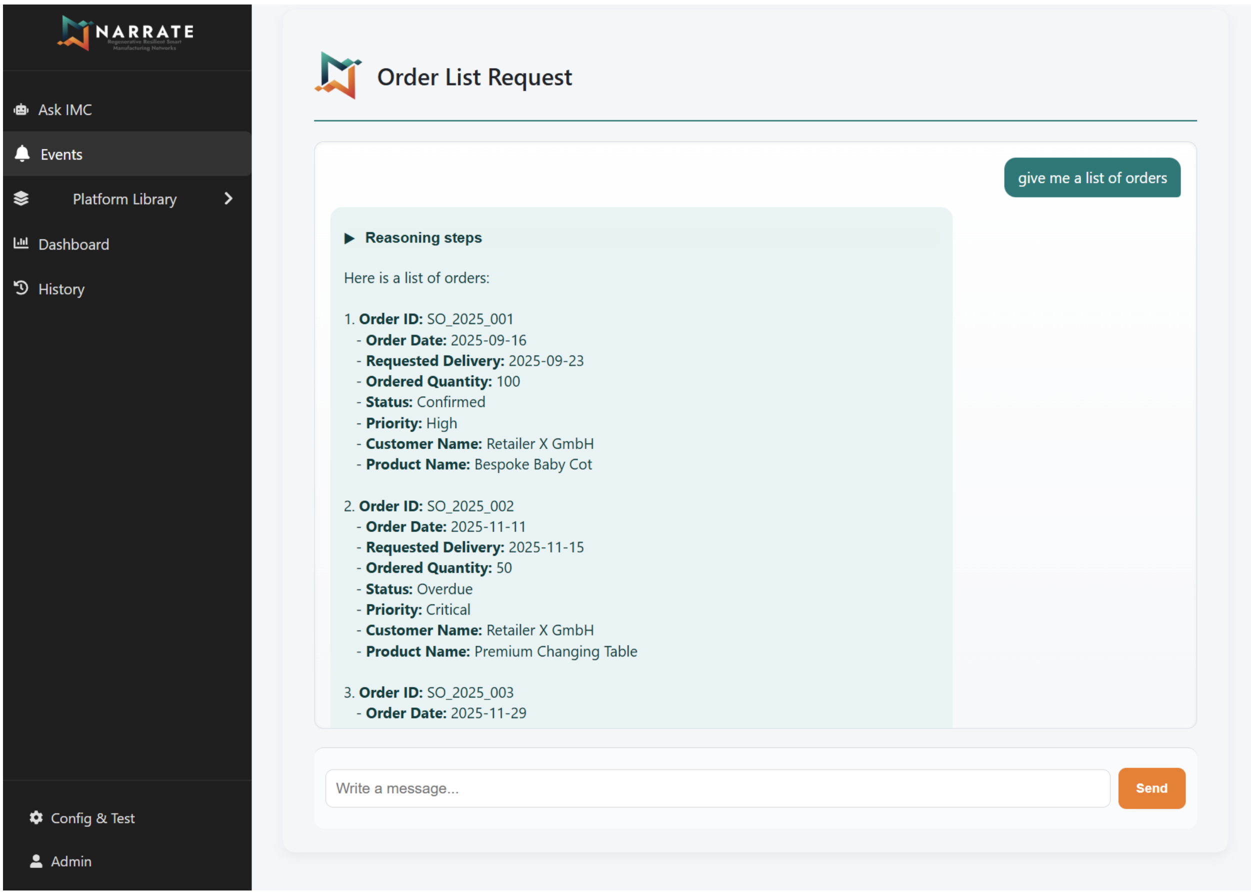
Ask anything about your workflow, events, processes or data.

FIGURE 3. MONITORING PLATFORM MAIN PAGE TO SUBMIT A QUERY IN NATURAL LANGUAGE

6.1.2. Events Received via RabbitMQ

Events represent system-level signals automatically generated by other components, monitoring services, or integrated platforms. They often encode machine-readable descriptions of anomalies, thresholds exceeded, or operational alerts. Although these events are typically less ambiguous than user queries, they still require semantic interpretation to determine their correspondence with known risk types. The NSQA module processes these events through the same risk classification and reasoning stages as human-generated inputs, ensuring consistency in decision logic across both automated and manual workflows.

Regardless of origin, all inputs are normalized into a shared internal representation and passed through the same **risk assessment pipeline**, which includes risk type selection, rule application, response plan retrieval or construction, and neuro-symbolic reasoning. This uniform treatment guarantees coherent behaviour and simplifies downstream auditing, traceability, and explainability. The following screenshot illustrates how a user can submit a natural language query through the platform interface.



give me a list of orders

▶ **Reasoning steps**

Here is a list of orders:

1. **Order ID:** SO_2025_001
   - **Order Date:** 2025-09-16
   - **Requested Delivery:** 2025-09-23
   - **Ordered Quantity:** 100
   - **Status:** Confirmed
   - **Priority:** High
   - **Customer Name:** Retailer X GmbH
   - **Product Name:** Bespoke Baby Cot

2. **Order ID:** SO_2025_002
   - **Order Date:** 2025-11-11
   - **Requested Delivery:** 2025-11-15
   - **Ordered Quantity:** 50
   - **Status:** Overdue
   - **Priority:** Critical
   - **Customer Name:** Retailer X GmbH
   - **Product Name:** Premium Changing Table

3. **Order ID:** SO_2025_003
   - **Order Date:** 2025-11-29

Write a message...

FIGURE 4. USER SUBMITTING A QUERY IN NATURAL AND GETTING A RESULT

6.2. RISK TYPE IDENTIFICATION

Risk type identification is the first structured interpretation step applied to any incoming input. Its purpose is to map a user query or event description to one of the **existing conceptual categories** (risk types) maintained within the system's knowledge base. This mapping provides essential semantic context that determines which response strategies, rules, and external information sources are relevant for the subsequent reasoning process.

In cases where the input does not semantically align with any of the stored risk types, the system proceeds by constructing and executing a coherent response strategy based solely on the available expert rules, external knowledge sources, and the general reasoning capabilities of the neuro-symbolic engine. This ensures that the system remains operational and capable of generating meaningful responses even when encountering entirely new or unclassified situations.

It is also important to note that the term risk type is used in a broad and flexible sense within the platform. Although originally conceived to categorize operational risks, a risk type may represent any recurrent situation, question, or domain-specific scenario that benefits from having an associated response plan. This includes routine queries, informational lookups, or procedural checks that are not risks in the traditional sense. By treating these diverse situations under the unified risk type abstraction, the system achieves greater consistency, reusability of knowledge, and extensibility of its reasoning framework.

The identification workflow unfolds as follows:

1. Retrieval of Available Risk Types

The system begins by loading the full set of risk types stored in the database. Each risk type represents a domain-relevant class of situations or problems, typically defined by experts through concise descriptions or example scenarios. These stored definitions provide the semantic grounding needed for comparison with incoming inputs.

2. Semantic Similarity Assessment via an LLM

A large language model is then used to evaluate the semantic closeness between the input risk and each stored risk type.

- The LLM performs *context-aware similarity ranking*, considering not only lexical similarity but also conceptual and pragmatic relationships.
- This allows the system to correctly match inputs even when they are phrased in unfamiliar or highly variable language.[4]

3. Selection of the Most Relevant Risk Type

Based on the similarity ranking, the system selects the risk type that best captures the meaning of the input. This selection determines which response plans and agents will be considered during the reasoning phase.

4. Automatic Creation of a New Risk Type (When Needed)

If the system determines that none of the existing risk types is sufficiently aligned with the input, it initiates the creation of a *new* risk type.

- The LLM synthesizes a concise semantic description that encapsulates the essential characteristics of the input.
- The new risk type is stored in the database so it can be reused for future queries or events with similar properties.
- This mechanism enables the NSQA module to expand its conceptual coverage over time and adapt to new types of risks that were not foreseen during initial expert configuration.

6.3. RESPONSE PLAN SELECTION

Once a risk type has been identified, the system proceeds to determine the most appropriate response strategy by examining the set of response plans[13] associated with that risk type. This phase constitutes a critical bridge between semantic interpretation and actionable reasoning, as it establishes the initial structure and constraints under which the neuro-symbolic reasoning process will operate.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

The NSQA module begins by **retrieving all response plans linked to the chosen risk type**, as stored in the knowledge base. These plans (typically defined by domain experts) represent structured procedures composed of ordered steps, expected interactions with external systems, and embedded domain-specific reasoning patterns. By analysing the available plans, the system attempts to select the one that best matches the characteristics of the incoming risk. Similarity assessment may rely on metadata, plan scope, historical usage, or textual annotations associated with the response plans.

Once the available plans are collected, the system **chooses the most appropriate match**. The goal is to anchor the reasoning process in a structured, expert-defined pathway whenever possible, ensuring consistency, reproducibility, and adherence to established operational protocols. If a suitable static plan is identified, it becomes the backbone of the subsequent REACT agent execution and guides how external information should be gathered or how decisions should be made.

However, if **no predefined response plan exists for the selected risk type**, the system transitions into a fallback mode in which the absence of a plan is explicitly recorded. This condition signals the need for **dynamic plan creation**, which will occur after the REACT agent completes its reasoning process. In this dynamic approach, the agent's sequence of steps, tool calls, and intermediate decisions will be distilled into a reusable response plan that may be stored for future risks of the same type.

6.4. RULE INTEGRATION

All expert rules stored in the system's knowledge base are injected into the reasoning context as soon as a risk has been classified and its initial response plan determined. During this stage, the rules are transformed into a structured internal representation that the reasoning engine can evaluate alongside the response plan's steps. These rules remain active throughout the entire decision-making process, influencing action selection, prioritization, and constraint enforcement. By embedding expert knowledge at this early stage, the system ensures that subsequent reasoning remains aligned with established domain practices and organizational policies.

6.5. REACT AGENT CONSTRUCTION AND EXECUTION

Once the system has identified the relevant risk type, selected an appropriate response plan (whether static or dynamically generated), and aggregated all applicable expert rules and retrieved facts, it proceeds to construct a **REACT agent**: a composite reasoning entity that orchestrates decision-making in a stepwise, transparent, and tool-augmented manner.

The construction process integrates several key components:

- **The selected response plan**, which provides an initial scaffold of intended actions, domain expectations, and reasoning patterns relevant to the identified risk.
- **All external agents associated with the chosen risk type**, such as OpenAPI agents, RAG-based retrieval agents, or other domain-specific connectors[9]. These agents give the REACT agent access to real-time information, external systems, and domain knowledge not explicitly encoded in rules or plans.
- **Expert rules**, which act as globally applicable constraints, heuristics, or contextual modifiers that shape the reasoning process at each decision step.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

- **Current facts**, including both previously known facts and those retrieved during earlier stages of processing. These facts populate the agent's working memory and inform subsequent decisions.

Once instantiated, the REACT agent follows a structured cycle of reasoning and action[7]:

1. **Reasoning** The agent evaluates the current state of information—plans, rules, and facts—to determine the next best step. This involves interpreting the intent of the response plan, assessing which constraints are active, and identifying any missing information that must be gathered.
2. Acting Through External Tools

Based on its reasoning, the agent may invoke one or more external agents to retrieve additional data, query APIs, execute domain operations, or validate assumptions. Each call is logged and incorporated into the evolving reasoning trace.

3. **Reflecting** After receiving observations from external calls, the agent reflects on the results, integrating newly acquired facts into its working memory and reassessing the situation. Reflection may trigger rule application, adjustments to the proposed sequence of actions, or reconsideration of strategy.
4. Iterating Until a Solution Emerges

This cycle of reasoning, acting, and reflecting continues until the agent reaches a well-justified resolution to the risk. The final output includes not only the solution itself but also a fully documented chain of reasoning that explains how decisions were made and why specific actions were taken.

Through this iterative process, the REACT agent operationalizes the neuro-symbolic paradigm: neural reasoning (e.g., interpretation, reflection, strategy adjustment) is tightly interwoven with symbolic structure (rules, plans, explicit steps), producing decisions that are both informed and explainable.

6.6. OUTPUT

At the conclusion of the reasoning process, the system produces a structured output that captures both the substantive resolution of the risk and the full explanation of how that resolution was reached. The output includes the final recommended action or decision derived from the executed response plan, accompanied by a clear natural-language justification that summarizes the rationale behind the chosen steps. Additionally, the system provides a detailed step-by-step reasoning trace that records all intermediate thoughts, tool interactions, retrieved facts, and rule applications. This trace ensures transparency, supports validation by domain experts, and enables future refinement of response plans.

6.7. DYNAMIC KNOWLEDGE CREATION

When the system encounters a situation where no predefined response plan offers adequate guidance, the reasoning steps taken by the agent become the foundation for constructing a new dynamic response plan. This plan encodes the sequence of actions, information retrieval operations, and inferred decision points that proved effective during execution.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

Once generated, the new response plan is stored for future reuse, where it can serve as expert knowledge for similar risks. To ensure quality and relevance, the system solicits user feedback after presenting the resolution. If users confirm that the automatically generated plan resolved the risk appropriately, it remains part of the knowledge base. If the feedback indicates that the plan was insufficient or misaligned with domain expectations, it is removed to prevent it from influencing future reasoning processes.

Through this mechanism, the system gradually expands and refines its symbolic knowledge base, leveraging operational experience to enhance future decision making while maintaining human oversight over knowledge consolidation.

7. EXPLAINABILITY AND TRACEABILITY

A fundamental requirement of the NSQA system is to ensure that every decision, inference step, and action taken by the IMC is fully interpretable and auditable. Given that the platform operates in environments where trust, accountability, and regulatory compliance are essential, explainability is not treated as an optional attribute but as a core architectural principle. The system therefore captures the entire reasoning process that leads from an initial risk input to a recommended response, exposing both the symbolic and neural components involved. This transparency enables users, auditors, and domain experts to understand how conclusions were reached, verify the correctness of the reasoning, and refine the underlying knowledge models when needed.

The system provides:

- A transparent chain of reasoning[8] that outlines how the system progressed from the initial risk input to the final recommended resolution. Each step of the reasoning process is written in natural language and grounded in the contextual information available at that moment.
- Detailed annotations for every decision taken during the reasoning cycle. These annotations clarify why a particular action was selected, which alternatives were considered, and what criteria guided the selection.
- Justification for each call to external agents, including API endpoints and retrieval components. This helps users understand how external evidence contributed to the system's evolving understanding of the risk.
- Symbolic rule activation logs that indicate which expert rules influenced the decision process. This includes the conditions that triggered each rule and the logical effects those rules had on subsequent reasoning steps.
- Generated natural-language summaries that synthesize the entire chain of reasoning into a concise and readable explanation. These summaries can be used for auditing, documentation, and user-facing communication.

Together, these explainability and traceability mechanisms create a comprehensive audit trail that documents every stage of the NSQA reasoning cycle. Users can reconstruct the logical pathway from input to output, inspect the interplay between expert rules, dynamic facts, and external data, and validate that each decision aligns with operational goals and domain constraints. This level of transparency not only strengthens trust in the system but also supports continuous improvement, enabling experts to refine rules, adjust response plans, and enhance overall decision quality over time.

8. EXAMPLES AND CASE STUDIES

The following examples illustrate how the NSQA system operates when confronted with real-world inputs drawn from the NARRATE industrial context. Each case study presents the initial user query or event, the reasoning steps executed by the NSQA pipeline, and the final outcome produced by the system. These examples demonstrate how expert rules, external knowledge sources, and response plans are mobilized to support decision-making, as well as how the neuro symbolic reasoning layer maintains transparency and traceability throughout the process.

By examining these scenarios, readers can better understand how the NSQA module behaves across different types of requests, ranging from operational supplier evaluation to knowledge retrieval and risk alert interpretation. The examples highlight both the flexibility and consistency of the reasoning framework, showing how structured logic and neural inference jointly contribute to actionable results.

USE CASE 1: ALTERNATIVE PROVIDER

Input: The provider of the material beechwood for order SO_2025_001 won't be able to deliver on time. How do I solve it?

Steps:

1. Get input via event or query

The system receives incoming information either as a natural-language query from a user (BB#6) or as a structured event (BB#2) generated by another component through RabbitMQ.

Regardless of format or origin, the input is normalized and transformed into a consistent internal representation. This unified preprocessing ensures that both human-initiated requests and machine-triggered alerts enter the same reasoning pipeline for risk analysis.

```
async def query(query_input: QueryInput, db: Session) -> QueryResponse:
    """

    """
    event = get_event_from_query(query_input)


    chat_manager = ChatManager()
    plan_manager = PlanManager()
    rules_manager = RulesManager()
    plugin_manager = PluginManager()
    risk_type_manager = RiskTypeManager()
    event_manager = EventManager()
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

```
    orchestrator_agent = OrchestratorAgent(
        chat_manager=chat_manager,
        plan_manager=plan_manager,
        rules_manager=rules_manager,
        plugin_manager=plugin_manager,
        risk_type_manager=risk_type_manager,
        event_manager=event_manager
    )

    response = await orchestrator_agent.get_response(event, db)

    return response

async def process_message(msg, db: Session):
    """
    """
    event = get_event_from_message(msg)

    chat_manager = ChatManager()
    plan_manager = PlanManager()
    rules_manager = RulesManager()
    plugin_manager = PluginManager()
    risk_type_manager = RiskTypeManager()
    event_manager = EventManager()

    orchestrator_agent = OrchestratorAgent(
        chat_manager=chat_manager,
        plan_manager=plan_manager,
        rules_manager=rules_manager,
        plugin_manager=plugin_manager,
        risk_type_manager=risk_type_manager,
        event_manager=event_manager
    )
    return await orchestrator_agent.get_response(event, db)
```

2. Select risk type

The system loads all existing risk types from the knowledge base (BB#2) and uses a large language model to compare the incoming input against these definitions.

Through semantic similarity assessment, it identifies which risk type best matches the meaning and context of the input.

If no suitable match exists, a new risk type is automatically generated and stored, enabling the system to adapt to previously unseen risks.

```
async def get_response(self, event_query: EventModel, db: Session) -> QueryResponse:
        """ """

        plan, decision, chat_id = await self._select_plan_advanced(
            event_query=event_query, db=db
        )

[...]

    async def _select_risk_type(
        self, possible_risk_types: List[RiskTypeData], event: EventModel
    ) -> tuple[str, str]:
        """
        Use an LLM to select the most appropriate risk_type given the user's query.
        """

        user_query = event.query

        risk_blocks = "\n\n".join(
            f"PLAN ID {rt.id}:\n{rt.description}" for rt in possible_risk_types
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
)

template = f"""
 You are an expert orchestrator that decides which risk type description best fits a user's question.

Here are the available risk type descriptions:

{risk_blocks}

User's question:
"{user_query}"

Decide which risk type description is most related to this question.
Respond ONLY with the ID (example: "1").
"""

prompt = PromptTemplate(template=template)

try:
    result = await (prompt | self.llm).ainvoke({})
    decision = result.content.strip() if result else ""
except Exception as e:
    print(f"Plan selection failed via LLM: {e}")
    decision = ""

return decision
```

3. Get plans from DB based on risk

After the risk type is determined, the system retrieves all response plans associated with that category from the database (BB#2).

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

It evaluates these plans—considering metadata, usage history, or semantic annotations—to select the one that most closely fits the situation.

If none is appropriate, the system records this and prepares for dynamic plan creation during reasoning.

```
class PlanManager(PlanManagerBase):
    async def get_plans(self, db: Session) -> list[PlanData]:
        plan_list_object: list[Plan] = db.scalars(select(Plan)).all()
        return [
            PlanData(
                id=plan.id,
                name=plan.name,
                description=plan.description,
                instructions=plan.instructions
            ) for plan in plan_list_object
        ]

    async def get_plan(self, id: int, db: Session) -> PlanData:
        plan: Plan | None = db.scalars(select(Plan).where(Plan.id == id)).first()

        if plan is None:
            return None

        return PlanData(
            id=plan.id,
            name=plan.name,
            description=plan.description,
            instructions=plan.instructions
        )

class RulesManager(RulesManagerBase):
    async def get_rules(self, db: Session) -> list[RuleData]:
        rule_object: list[Rule] = db.scalars(select(Rule)).all()
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

```
    return [
        RuleData(
            id=rule.id,
            name=rule.name,
            description=rule.description,
            expression=rule.expression,
            priority=rule.priority
        ) for rule in rule_object
    ]
```

4. Plan and execute using ReAct agent

A ReAct agent (as part of BB#4) is constructed using the chosen plan (or an empty scaffold), all expert rules, external agents, and known facts.

It proceeds through iterative cycles of reasoning, tool-calling, and reflection, gathering information and adjusting steps as needed.

The process continues until the agent reaches a well-justified response, producing both an actionable recommendation and a transparent reasoning trace.

```
async def get_response(self, event_query: EventModel, db: Session) -> QueryResponse:


[...]


    elif str(decision) == "1":
        react_agent = ReActAgent(
            question=event_query.query,
            plan_prompt=plan,
            criteria_prompt=rules,
            history=history,
            tools=tools,
        )
        react_agent.build_agent()
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
    def build_agent(self):
        if not self.plan_prompt:
            self.plan_prompt = default_plan_template
        self.final_prompt = self._build_react_prompt(
            self.plan_prompt, self.criteria_prompt, self.history
        )
        self.planner = load_chat_planner(self.llm, self.final_prompt)
        self.executor = load_agent_executor(tools=self.tools, llm=self.llm)
        self.react_chain = PlanAndExecute(planner=self.planner, executor=self.executor)
        self.logger.info("Successfully build ReAct agent")
```

```
[...]
base_prompt = """
You are an intelligent reasoning agent designed to answer user questions accurately and efficiently.
You have access to various tools to help you complete and execute your reasoning.

Your objective:
- Follow the provided plan to reason through the question and use tools to gather data to answer.
- Evaluate your reasoning and final answer against the provided criteria.
---

PLAN:
__PLAN__
---

CRITERIA:
__CRITERIA__
---

CHAT HISTORY:
```

```
__HISTORY__

USER'S QUERSION:
{question}
---
"""


[...]


def plan_and_execute(self):
    self.logger.info(f"{self.__class__.__name__}: Getting agent response.")
    max_retries = int(settings.max_retries_parser_error_func_call)
    attempts = 0

    callback_handler = ReActCallbackHandler(self.plan_actions)

    while attempts < max_retries:
      try:
        result = self.react_chain.invoke(
          {"input": f"Use the available tools to generate a response: {self.question}"},
          config={"callbacks": [callback_handler]},
        )

        self.logger.info(result)

        reasoning_steps = self._get_reasoning_steps()

        return ResponseModel(
          question=self.question,
          answer=result["output"],
          context=ContextModel(
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

```
                    reasoning_steps=reasoning_steps,
                    success=True
                )
            )

        except OutputParserException as e:
            attempts += 1
            if attempts == max_retries:
                try:
                    reasoning_steps = self._get_reasoning_steps()
                except:
                    reasoning_steps = None
                return ResponseModel(
                    question=self.question,
                    answer=None,
                    context=ContextModel(
                        reasoning_steps=reasoning_steps,
                        success=False,
                        error_message=str(e)
                    )
                )

            self.logger.warning(
               f"Retrying {attempts}/{max_retries} times openapi_chain response because of Exception {e}... "
            )

        except Exception as e:
            if "list index out of range" in str(e):
                try:
                    reasoning_steps = self._get_reasoning_steps()
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
    except Exception:
        reasoning_steps = []

    return ResponseModel(
        question=self.question,
        answer=(
            "The system could not generate a tool-based plan, "
            "so here is a direct answer:\n\n" + self._fallback_answer()
        ),
        context=ContextModel(
            reasoning_steps=reasoning_steps,
            success=False,
            error_message=f"Planner error: {str(e)}"
        )
    )

self.logger.error(
    f"{self.__class__.__name__}: An exception ocurred: {e}"
)
import traceback
self.logger.error(f"Exception type: {type(e)}")
self.logger.error(traceback.format_exc())
# Returning something for testing
try:
    reasoning_steps = self._get_reasoning_steps()
except:
    reasoning_steps = None
return ResponseModel(
    question=self.question,
    answer=None,
    context=ContextModel(
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

```
                        reasoning_steps=reasoning_steps,
                        success=False,
                        error_message=str(e)
                    )
                )
```

4.1 Calls to OpenAPI agent

During execution, the ReAct agent may identify the need for external information or system interaction and invokes the appropriate OpenAPI agent.

These agents act as structured connectors to external services, enforcing schema validation and standardized request formats.

By delegating calls through these agents, the system ensures reliability, safety, and consistent integration with operational APIs.

```
def get_tool(plugin: PluginInfo):
    agent = OpenApiAgent(plugin=plugin)
    agent.build_agent()

    tool = Tool(
        name=plugin.title,
        description=(
            plugin.description
        ),
        func=lambda question: agent.ask_agent(question)
    )
    return tool

[...]

class OpenApiAgent:
    def __init__(self, plugin: PluginInfo):
        self.logger: Logger = logger
        self.id = plugin.id
        self.title = plugin.title
        self.url = plugin.url
        self.schema_url = plugin.schema_url
        self.verify_ssl = plugin.verify_ssl
        self.prompt = plugin.prompt
        self.auth_type = plugin.auth_type
        self.auth_params = plugin.auth_params
        self.config_params = plugin.config_params
        self.llm = llm

    def build_agent(self):
        self.logger.debug(f"{self.__class__.__name__}: Building agent.")

        secret_key = settings.encrypt_secret_key

        # Load and parse OpenAPI schema
        schema = self._load_openapi_schema()
        self.spec = OpenAPISpec.from_spec_dict(schema)

        # Set headers based on auth type
        self.headers = self._build_headers(secret_key)

        # Load optional config parameters
        self._load_config_params(secret_key)

        # Construct final prompt
        self._build_prompt_template()

        self.openapi_chain = self._build_chain()
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
    def ask_agent(self, question):
        self.logger.critical(f"{self.__class__.__name__}: Getting agent response.")
        max_retries = int(settings.max_retries_parser_error_func_call)
        attempts = 0
        while attempts < max_retries:
            try:
                response = [
                    self.openapi_chain.invoke(
                        {"question": question}
                    )
                ]
                self.logger.info(f"Raw chain output: {response}")
                return response

            except OutputParserException as e:
                print(e)
                attempts += 1
                if attempts == max_retries:
                    return []
                self.logger.warning(
                    f"Retrying {attempts}/{max_retries} times openapi_chain response because of
Exception {e}... "
                )

            except Exception as e:
                self.logger.error(
                    f"{self.__class__.__name__}: An exception ocurred while querying the plugin:
{e}"
                )
                return []

    def _load_openapi_schema(self):
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
        try:
            schema = get_openapi_schema(self.schema_url, self.url)
        except Exception as e:
            raise HTTPException(
                status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
                detail=f"Error while retrieving the schema: {e}",
            )
        return schema

    def _build_headers(self, secret_key):
        if self.auth_type == "bearer_token":
            decrypted = EncryptDecrypt.decrypt_secret_key(
                self.auth_params, secret_key=secret_key
            )
            parsed = json.loads(decrypted.replace("'", '"'))
            self.auth_token = parsed.get("bearer_token")
            return {
                "Authorization": f"Bearer {self.auth_token}",
                "Content-Type": "application/json",
                "Accept": "application/json",
            }

        self.auth_token = None
        return {
            "Content-Type": "application/json",
            "Accept": "application/json",
        }

    def _load_config_params(self, secret_key):
        self.parameters = {}
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

```
        if self.config_params:
            decrypted = EncryptDecrypt.decrypt_secret_key(
                self.config_params, secret_key=secret_key
            )
            if decrypted:
                self.parameters = json.loads(decrypted.replace("'", '"'))

    def _build_prompt_template(self):
        self.prompt_template = PromptTemplate(
            input_variables=["question"], template=self.prompt
        )

    def _build_chain(self):
        return get_custom_openapi_chain(
            self.spec,
            self.llm,
            verbose=True,
            headers=self.headers,
            params=self.parameters,
            verify_ssl=self.verify_ssl,
            prompt=self.prompt_template,
        )
```

4.2 OpenAPI calls to intermediate API (BB#5)

When an OpenAPI agent is triggered, it formulates and sends requests directly to the intermediate API or related backend endpoints (BB#5).

This enables the retrieval of domain-specific data, validation of system states, or execution of predefined operations required by the reasoning cycle.

The results returned by the intermediate API are then fed back into the ReAct agent's working memory as new facts or observations.

```
@router.get("/orders", response_model=OrdersResponse)
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

```python
def get_orders():
    """
    Returns detailed sales order information including
    customer, product, and manufacturer data.
    """
    records = run_query(ORDERS_QUERY)
    if not records:
        raise HTTPException(status_code=404, detail="No orders were found.")
    return {"count": len(records), "orders": records}

@router.get("/product/{product_id}", response_model=ProductDetailsResponse)
def get_product_details(product_id: str):
    """
    Returns detailed information of a product by ID.
    """
    records = run_query(PRODUCT_DETAILS_QUERY, {"product_ID": product_id})
    if not records:
        raise HTTPException(status_code=404, detail=f"Product '{product_id}' not found.")
    return {"count": len(records), "productDetails": records}

@router.get("/suppliers/", response_model=SuppliersByMaterialResponse)
def get_suppliers_by_material(workorder_id: str, input_id: str):
    """
    Returns suppliers related to a specific Material or Product Component and Workorder ID.
    The input ID can be either a `materialID` or a `componentID`.
    Workorder ID is the parameter `workOrderID`.
    """
    records = run_query(SUPPLIERS_BY_MATERIAL, {
        "workOrderID": workorder_id,
        "inputID": input_id
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
    })
    if not records:
        raise HTTPException(
            status_code=404,
            detail=f"Material or component id '{input_id}' not found."
        )
    return {"input_id": input_id, "count": len(records), "suppliers": records}

@router.get("/workorders/", response_model=WorkOrdersResponse)
def get_all_workorders():
    records = run_query(ALL_WORKORDERS)
    if not records:
        raise HTTPException(
            status_code=404,
            detail=f"No workorders were found."
        )
    return {"count": len(records), "workorders": records}

@router.get("/all-suppliers/", response_model=SupplierListResponse)
def get_all_suppliers():
    records = run_query(ALL_SUPPLIERS)
    if not records:
        raise HTTPException(
            status_code=404,
            detail=f"No suppliers were found."
        )
    return {"suppliers": records}

@router.get("/all-products/", response_model=ProductsResponse)
def get_all_products():
    records = run_query(ALL_PRODUCTS)
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
    if not records:
        raise HTTPException(
            status_code=404,
            detail=f"No products were found."
        )
    return {"count": len(records), "products": records}
```

4.3 ReAct agent reasoning steps

After receiving responses from external calls, the ReAct agent incorporates the new information into its internal context.

It applies expert rules, interprets the implications of new facts, and decides the next action in the reasoning chain.

This loop of reasoning, acting, and reflecting continues until the agent converges on a stable and well-supported resolution.

Reasoning steps:

1. **Check all orders**

Task: Find the order with ID SO_2025_001 and retrieve the associated product ID and work order ID.

Response:

- **Product ID**: PROD_2025_001
- **Work Order ID**: WO_2025_001

2. **Get product details**

Task: Retrieve product details using the product ID to identify the beechwood material/component ID.

Response:

- **Material Name**: Beechwood (FSC-certified)
- **Material ID**: MAT_2025_BW_001
- **Material Type**: Wood
- **Material Origin**: Germany, Black Forest

3. **Find suppliers**

Task: Using the beechwood material ID and work order ID, find all potential suppliers.

Response:

- **Actual Supplier**:
  - Supplier Name: *EcoWood Partners GmbH*
  - Supplier ID: MS_002
  - Location: Mannheim, Germany
  - Lead Time: 2 days
  - Certifications: FSC, PEFC, ISO 9001
  - Supplying Date: 2025-09-17
- **Potential Suppliers**:
  - *ForestLine Timber AG*:
    - Supplier ID: MS_003
    - Location: Zurich, Switzerland
    - Lead Time: 5 days
    - Certifications: FSC, ISO 45001
  - *GreenWood Supplies Ltd.*
    - Supplier ID: MS_001
    - Location: Freiburg, Germany
    - Lead Time: 3 days
    - Certifications: FSC, ISO 14001

4. **Propose a solution**

Task: Recommend the best supplier to avoid disruptions.

Response:

- **Best Supplier**: *EcoWood Partners GmbH* — shortest lead time (2 days), certified (FSC, PEFC, ISO 9001), geographically close.

- **Backup Supplier**: *GreenWood Supplies Ltd.* — 3-day lead time and close proximity.

5. Final response generation

Once the reasoning process concludes, the system compiles a structured output (BB#4) that includes the recommended action or solution.

Alongside the result, it generates an explanatory narrative clarifying why each step was taken and how the conclusion was reached.

This final response ensures transparency, traceability, and readiness for user review or automated downstream execution.

```
        reasoning_steps = self._get_reasoning_steps()

        return ResponseModel(
            question=self.question,
            answer=result["output"],
            context=ContextModel(
                reasoning_steps=reasoning_steps,
                success=True
            )
        )

[…]
    response: ResponseModel = react_agent.plan_and_execute()
[…]
return QueryResponse(chat_id=chat_id, response=response)
```

Final output:
The supplier currently assigned to order **SO_2025_001**, **EcoWood Partners GmbH**, will not be able to deliver the beechwood on time despite having an average lead time of 2 days and the required certifications (FSC, PEFC, ISO 9001).

To avoid delays in the order fulfillment, a viable solution is to **switch to an alternative supplier**. **GreenWood Supplies Ltd.**, located in Freiburg, Germany, is a suitable replacement due to its **short lead time of 3 days** and geographical proximity.

Recommended action: Reassign the beechwood supply for SO_2025_001 to GreenWood Supplies Ltd. to ensure timely delivery and minimize the impact on the overall order schedule.

USE CASE 2: SIMPLE QUESTION TO GRAPH DB

In this scenario, the system interacts with a graph database, which enables querying highly interconnected manufacturing information in a flexible and semantically rich way. The use of a graph structure allows the NSQA to retrieve relationships across products, components, materials, suppliers, and orders, supporting traceability questions that would be more complex to express in traditional relational schemas.

Graph data for the use cases in menu point 10.1 shows the high-level structure of the graph database used in this example. It provides an overview of how traceability information is modelled and how different entities participate in the manufacturing lifecycle of the COT case (BB#5 4.2)

Input: Give me a list of orders

Reasoning steps:

1. Check the API
   Task: Call the API to answer the question: *"Give me a list of orders."*
   Response:
   - Total Orders: 2
   - Orders:

Order 1

- **Order ID**: SO_2025_001
- **Order Date**: 2025-09-16
- Requested Delivery: 2025-09-23
- Ordered Quantity: 100
- **Status**: Confirmed
- Priority: High
- **Customer**: Retailer X GmbH (Munich, Germany)
  - Contact: Anna Schmidt — anna.schmidt@retailerx.de
- **Product**: Bespoke Baby Cot
  - Product ID: PROD_2025_001
- **Manufacturer**: CotMakers Ltd. (Stuttgart, Germany)
  - Certifications: ISO 9001, FSC Chain-of-Custody
  - Contact: Markus Weber — m.weber@cotmakers.de
- Work Order:
  - Work Order ID: WO_2025_001
  - Creation Date: 2025-09-17
  - Scheduled Start: 2025-09-18
  - Scheduled Completion: 2025-09-22
  - Status: Planned
  - Quantity: 100
  - Department: Assembly Line A

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

    - Work Order Priority: High

Order 2

- **Order ID**: SO_2025_002
- **Order Date**: 2025-11-11
- Requested Delivery: 2025-11-15
- Ordered Quantity: 50
- **Status**: Overdue
- **Priority**: Critical
- **Customer**: Retailer X GmbH (Munich, Germany)
    - Contact: Anna Schmidt — anna.schmidt@retailerx.de
- **Product**: Premium Changing Table
    - Product ID: PROD_2025_002
- **Manufacturer**: CotMakers Ltd. (Stuttgart, Germany)
    - Certifications: ISO 9001, FSC Chain-of-Custody
    - Contact: Markus Weber — m.weber@cotmakers.de
- Work Order:
    - Work Order ID: WO_2025_002
    - Creation Date: 2025-11-12
    - Scheduled Start: 2025-11-13
    - Scheduled Completion: 2025-11-23
    - Status: In Production
    - Quantity: 50
    - Department: Assembly Line B
    - Work Order Priority: Critical

Output:

Here is a list of orders:

1. Order ID: SO_2025_001
    - Order Date: 2025-09-16
    - Requested Delivery: 2025-09-23
    - Ordered Quantity: 100
    - Status: Confirmed
    - Priority: High
    - Customer Name: Retailer X GmbH
    - Customer Location: Munich, Germany
    - Product Name: Bespoke Baby Cot
    - Manufacturer Name: CotMakers Ltd.
    - Work Order ID: WO_2025_001
    - Work Order Status: Planned
2. Order ID: SO_2025_002
    - Order Date: 2025-11-11
    - Requested Delivery: 2025-11-15
    - Ordered Quantity: 50
    - Status: Overdue
    - Priority: Critical
    - Customer Name: Retailer X GmbH
    - Customer Location: Munich, Germany

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

- Product Name: Premium Changing Table
- Manufacturer Name: CotMakers Ltd.
- Work Order ID: WO_2025_002
- Work Order Status: In Production

If you need more details about any specific order, feel free to ask!

USE CASE 3: ALERTS AND RAG DATA

Input: Alert (Nozzle Operation Time Error): MELTIO M600 exceed the threshold value. Actual: 301 h Threshold: 300 h

1. Get input via event or query

The system receives incoming information either as a natural-language query (BB#6) from a user or as a structured event (BB#2) generated by another component through RabbitMQ.

Regardless of format or origin, the input is normalized and transformed into a consistent internal representation.
This unified preprocessing ensures that both human-initiated requests and machine-triggered alerts enter the same reasoning pipeline for risk analysis.

```
async def query(query_input: QueryInput, db: Session) -> QueryResponse:
    """

    """
    event = get_event_from_query(query_input)

    chat_manager = ChatManager()
    plan_manager = PlanManager()
    rules_manager = RulesManager()
    plugin_manager = PluginManager()
    risk_type_manager = RiskTypeManager()
    event_manager = EventManager()

    orchestrator_agent = OrchestratorAgent(
        chat_manager=chat_manager,
        plan_manager=plan_manager,
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
        rules_manager=rules_manager,
        plugin_manager=plugin_manager,
        risk_type_manager=risk_type_manager,
        event_manager=event_manager
    )

    response = await orchestrator_agent.get_response(event, db)

    return response

async def process_message(msg, db: Session):
    """
    """
    event = get_event_from_message(msg)

    chat_manager = ChatManager()
    plan_manager = PlanManager()
    rules_manager = RulesManager()
    plugin_manager = PluginManager()
    risk_type_manager = RiskTypeManager()
    event_manager = EventManager()

    orchestrator_agent = OrchestratorAgent(
        chat_manager=chat_manager,
        plan_manager=plan_manager,
        rules_manager=rules_manager,
        plugin_manager=plugin_manager,
        risk_type_manager=risk_type_manager,
        event_manager=event_manager
    )
    return await orchestrator_agent.get_response(event, db)
```

2. Select risk type

The system loads all existing risk types from the knowledge base (BB#2) and uses a large language model to compare the incoming input against these definitions.

Through semantic similarity assessment, it identifies which risk type best matches the meaning and context of the input.

If no suitable match exists, a new risk type is automatically generated and stored, enabling the system to adapt to previously unseen risks.

```
async def get_response(self, event_query: EventModel, db: Session) -> QueryResponse:
    """ """

    plan, decision, chat_id = await self._select_plan_advanced(
        event_query=event_query, db=db
    )

[...]

  async def _select_risk_type(
    self, possible_risk_types: List[RiskTypeData], event: EventModel
  ) -> tuple[str, str]:
    """
    Use an LLM to select the most appropriate risk_type given the user's query.
    """

    user_query = event.query

    risk_blocks = "\n\n".join(
        f"PLAN ID {rt.id}:\n{rt.description}" for rt in possible_risk_types
    )

    template = f"""
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
     You are an expert orchestrator that decides which risk type description best fits a user's question.


    Here are the available risk type descriptions:


    {risk_blocks}


    User's question:
    "{user_query}"


    Decide which risk type description is most related to this question.
    Respond ONLY with the ID (example: "1").
    """


    prompt = PromptTemplate(template=template)


    try:
        result = await (prompt | self.llm).ainvoke({})
        decision = result.content.strip() if result else ""
    except Exception as e:
        print(f"Plan selection failed via LLM: {e}")
        decision = ""


    return decision
```

3. Get plans from DB based on risk

After the risk type is determined, the system retrieves all response plans associated with that category from the database.

It evaluates these plans—considering metadata, usage history, or semantic annotations—to select the one that most closely fits the situation.

If none is appropriate, the system records this and prepares for dynamic plan creation during reasoning.

```
class PlanManager(PlanManagerBase):
    async def get_plans(self, db: Session) -> list[PlanData]:
        plan_list_object: list[Plan] = db.scalars(select(Plan)).all()
        return [
            PlanData(
                id=plan.id,
                name=plan.name,
                description=plan.description,
                instructions=plan.instructions
            ) for plan in plan_list_object
        ]

    async def get_plan(self, id: int, db: Session) -> PlanData:
        plan: Plan | None = db.scalars(select(Plan).where(Plan.id == id)).first()

        if plan is None:
            return None

        return PlanData(
            id=plan.id,
            name=plan.name,
            description=plan.description,
            instructions=plan.instructions
        )

class RulesManager(RulesManagerBase):
    async def get_rules(self, db: Session) -> list[RuleData]:
        rule_object: list[Rule] = db.scalars(select(Rule)).all()
        return [
            RuleData(
                id=rule.id,
                name=rule.name,
                description=rule.description,
                expression=rule.expression,
                priority=rule.priority
            ) for rule in rule_object
        ]
```

4. Calls RAG tool to get data

When the ReAct agent (as part of the BB#4) determines that additional contextual or historical information is needed, it invokes the RAG (Retrieval-Augmented Generation) tool.
The RAG tool queries embedded knowledge sources (such as document collections, logs, or domain records) to retrieve semantically relevant information.

The retrieved data is then injected into the agent's working memory, enriching its factual context and enabling more accurate reasoning in subsequent steps.

```
        if str(decision) == "2":
            response: ResponseModel = query_chroma_rag(event_query.query)


def query_chroma_rag(question: str):
    """Query an existing Chroma DB using Azure Chat and embeddings."""


    embeddings = AzureOpenAIEmbeddings(
        api_key=settings.azure_openai_api_key,
        azure_endpoint=settings.azure_openai_base_url,
        azure_deployment=settings.azure_openai_embedding_model_deployment_name,
        openai_api_version=settings.azure_openai_api_version
    )
    with open("imc/databases/chroma_mock/mock_data.txt", "r", encoding="utf-8") as f:
        additional_context = f.read()


    vectordb = Chroma(
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

```
        persist_directory="imc/databases/chroma_mock",
        embedding_function=embeddings
    )
    retriever = vectordb.as_retriever(search_kwargs={"k": 3})
    llm = LLM

    prompt = ChatPromptTemplate.from_template(
    """You are an intelligent assistant helping answer user questions based on the provided context.

    Question:
    {input}

    Context:
    {context}

    Additional context:
    {additional_context}

    Provide a clear and concise answer based only on the context above.
    If the context does not contain enough information, say you don't know.
    """
    )

    combine_chain = create_stuff_documents_chain(llm, prompt)
    rag_chain = create_retrieval_chain(retriever, combine_chain)

    result = rag_chain.invoke({"input": question, "additional_context":additional_context})
    print("Answer:\n", result["answer"])
    print("Context:")
    for d in result.get("context", []):
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
        print("-", d.page_content[:200], "...\n")

    return ResponseModel(
        question=question,
        answer=result["answer"],
        context=ContextModel(
            context_documents=result["context"]
        )
    )
```

RAG Context:

```
            "context_documents": [
                {
                    "id": null,
                    "metadata": {},
                    "page_content": "7. Machine Error – AL06 Cooling Error\n\nSolution:\nThe AL06 error indicates improper cooling pressure or flow (Page 62). Follow the documented procedure: adjust the O2 sensor, probe, and door valves to 1–1.4 L/min and verify that head and gantry flow auto-regulate to approx. 8 L/min and 2.4 L/min.\nIf pressure remains above 0.50 MPa, adjust the chiller bypass until stable operation at 0.45–0.48 MPa is achieved.\n\n8. Nozzle Operation Time – Current: 301 h (Limit: 300 h)\n\nSolution:\nThe wire nozzle has exceeded its typical lifespan of 50–300 hours (Page 178). Replace the nozzle following the inspection and removal instructions to avoid print defects.\nContinued use past this threshold may cause beam misalignment and deposition quality issues.\n\n9. Nozzle Needs Replacement Message",
                    "type": "Document"
                },
                {
                    "id": null,
                    "metadata": {},
                    "page_content": "9. Nozzle Needs Replacement Message\n\nSolution:\nSince the operating hours exceed the recommended limit, replace the nozzle immediately following the 5-step maintenance procedure. Visually inspect the nozzle and compare it to wear levels A, B, and C to confirm the condition.\nReinstall the shield gas nozzle after replacement to restore proper
```

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

```
operation.\n\n10. Power Peak Consumption – Current: 11.2 kW (Limit: 10.6 kW)\n\nSolution:\nA peak consumption of 11.2 kW exceeds the typical maximum noted in the specifications (Page 38). Check the electrical installation, cable sizing, and power distribution to ensure they match the machine requirements.\nHigh peak draw may signal issues with the chiller, lasers, or internal electronics, and should be investigated to prevent electrical overloads.",

            "type": "Document"

        },

        {

            "id": null,

            "metadata": {},

            "page_content": "Regulate chiller bypass\nIf the chiller still displays the error, adjust the chiller bypass by opening the valve little by little and repeat the cooling process until the displayed value is around 0.45-0.48MPa.\nOnce the chiller is running without any alarm, check the flow meters located on the Meltio M600 left side panel:\nFrom left to right, approximately values should be set to:\nHead - 8 - 8.5L/min (error displayed if below 7.8L/min)\nGantry - 2.2 - 2.5L/min (error displayed if below 2.1L/min)\nO2 sensor - 1 - 1.5L/min (error displayed if below 0.7L/min)\nProbe - 1 - 1.5L/min (error displayed if below 0.7L/min)\nDoor - 1 - 1.5L/min (error displayed if below 0.7L/min)\n\n-----------------\n\n## ERROR HANDLING\n\n1. Storage Temperature – Current: 65 °C (Limit: 50 °C)",

            "type": "Document"

        }

    ]
```

5. Return final response

After retrieving and integrating information from the RAG tool, the system synthesizes the enriched context and generates a final response that incorporates the newly retrieved knowledge.
This response includes the context of how the RAG-sourced data influenced the outcome, ensuring transparency and justifying why the final recommendation or answer is valid.

```
    return QueryResponse(chat_id=chat_id, response=response)
```

Final response

The alert indicates that the nozzle operation time for the MELTIO M600 has exceeded the threshold value, with the actual time being 301 hours compared to the threshold of 300 hours. The solution is to replace the wire nozzle, as it has surpassed its typical lifespan of 50–300 hours. Continued use beyond this threshold may lead to beam misalignment and

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.

deposition quality issues. Follow the inspection and removal instructions to replace the nozzle and avoid print defects.

Those 3 use cases are deeply explained in deliverable 6.1[13] as the result of the integration of several NARRATE modules.

Taken together, these case studies show how the NSQA system provides reliable, interpretable, and context-aware decision support across a diverse range of situations. The reasoning traces reveal not only the sequence of actions taken by the system but also the underlying rationale, dependency on expert knowledge, and use of external information channels. This illustrates how the NSQA module transforms raw events or natural language prompts into structured insights aligned with the operational goals of the NARRATE platform.

These examples therefore serve both as validation of the pipeline design and as practical guidance for users and integrators seeking to understand how the system will behave in real deployments. Additional scenarios can be constructed following the same structure to further expand the library of test cases and support ongoing evaluation and refinement of the NSQA reasoning engine.

9. CONCLUSION

The NSQA module presented in this document establishes a comprehensive neuro symbolic foundation for intelligent decision support within the IMC platform. By integrating neural language understanding with structured symbolic knowledge, the system is capable of interpreting heterogeneous risk inputs, selecting or generating tailored response strategies, and grounding its decisions in both expert defined rules and dynamically acquired external information. This hybrid architecture allows the module not only to reason effectively but also to communicate its internal process through transparent, human readable explanations.

Throughout the processing pipeline, the NSQA module preserves a clear separation of concerns. Semantic parsing and risk type identification ensure that incoming information is properly contextualized. Expert rules, static response plans, dynamic response plans, and external facts form a coherent knowledge substrate that guides the system toward informed and consistent decisions. The REACT based reasoning process adds an additional layer of interpretability, enabling stepwise evaluation, reflective action, and detailed traceability of each decision path taken.

The module also supports continual refinement of its knowledge base by learning new response strategies from observed reasoning episodes. This capacity ensures that the NSQA component will evolve alongside the operational environment, increasing its alignment with real world scenarios and expert expectations.

Overall, the NSQA system positions the IMC platform to deliver advanced, explainable, and context aware decision support capabilities. Its design emphasizes transparency, modularity, and adaptability, providing a solid foundation for future extensions such as deeper rule-based generalization, advanced agent coordination strategies, and richer forms of semantic alignment. As the platform grows, the NSQA module can evolve accordingly, reinforcing its role as a central pillar in IMC's intelligent reasoning and decision-making ecosystem.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.

10. APPENDIX

10.1. GRAPH DATA FOR THE USE CASES

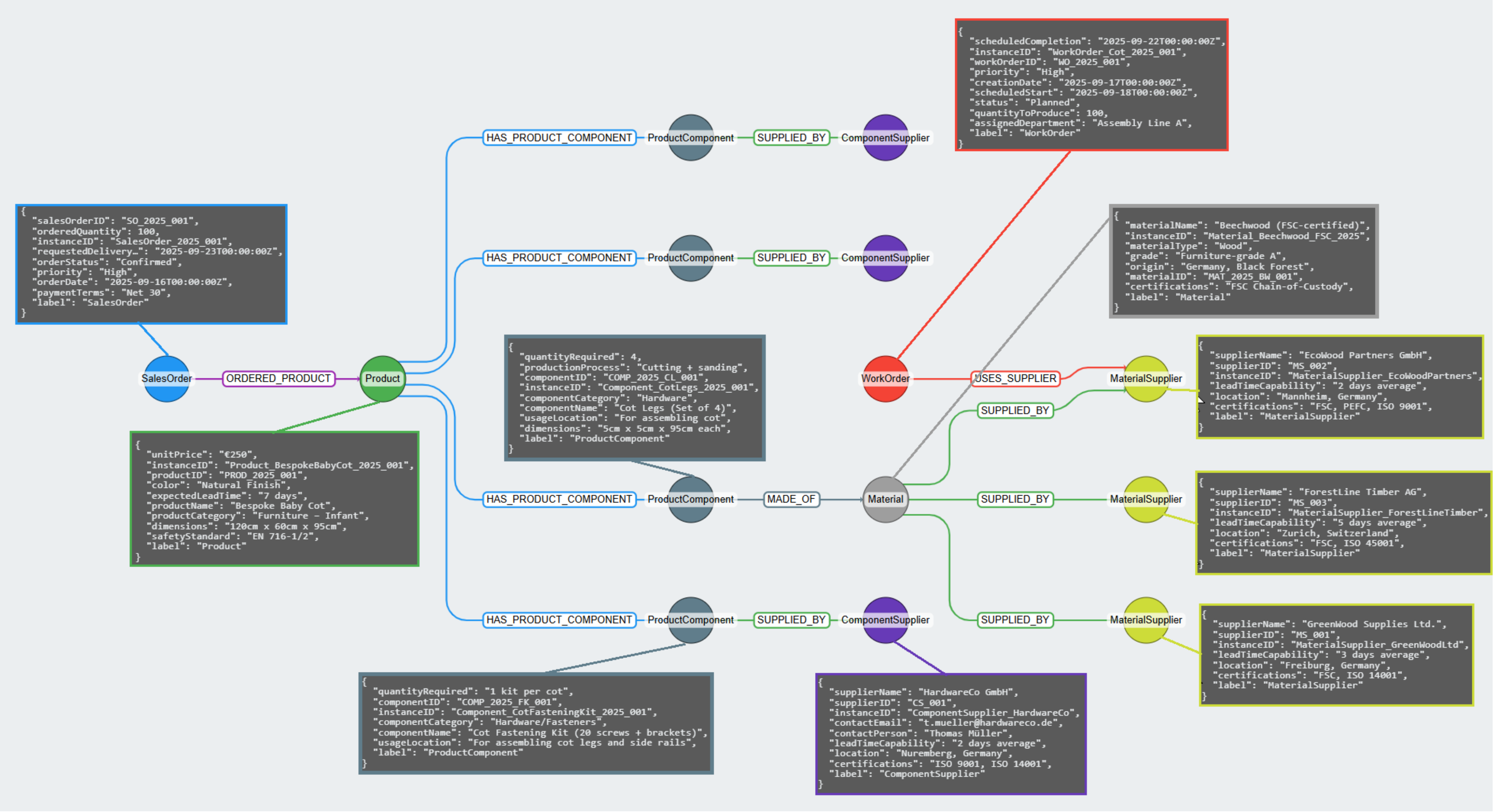
The following figure illustrates a graph-based data model that captures the full traceability chain of a manufactured product (COT case. BB#5 4.2). It shows how a sales order is linked to the ordered product, which is in turn decomposed into several product components. Each component is associated with its respective component supplier and, where applicable, the underlying material from which it is made. Materials are connected to one or multiple material suppliers, including details such as certifications, provenance, and supply attributes. The work order related to the product ties together the operational data, including the specific materials and suppliers used in the manufacturing process. Overall, the diagram visualizes the relationships among orders, products, components, materials, and suppliers, demonstrating how data is interconnected within the graph database.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation

UK Research and Innovation, Horizon Europe Guarantee.



```
{
  "scheduledCompletion": "2025-09-22T00:00:00Z",
  "instanceID": "WorkOrder_Cot_2025_001",
  "workOrderID": "WO_2025_001",
  "priority": "High",
  "creationDate": "2025-09-17T00:00:00Z",
  "scheduledStart": "2025-09-18T00:00:00Z",
  "status": "Planned",
  "quantityToProduce": 100,
  "assignedDepartment": "Assembly Line A",
  "label": "WorkOrder"
}
```

```
{
  "salesOrderID": "SO_2025_001",
  "orderedQuantity": 100,
  "instanceID": "SalesOrder_2025_001",
  "requestedDelivery…": "2025-09-23T00:00:00Z",
  "orderStatus": "Confirmed",
  "priority": "High",
  "orderDate": "2025-09-16T00:00:00Z",
  "paymentTerms": "Net 30",
  "label": "SalesOrder"
}
```

```
{
  "materialName": "Beechwood (FSC-certified)",
  "instanceID": "Material_Beechwood_FSC_2025",
  "materialType": "Wood",
  "grade": "Furniture-grade A",
  "origin": "Germany, Black Forest",
  "materialID": "MAT_2025_BW_001",
  "certifications": "FSC Chain-of-Custody",
  "label": "Material"
}
```

```
{
  "quantityRequired": 4,
  "productionProcess": "Cutting + sanding",
  "componentID": "COMP_2025_CL_001",
  "instanceID": "Component_CotLegs_2025_001",
  "componentCategory": "Hardware",
  "componentName": "Cot Legs (Set of 4)",
  "usageLocation": "For assembling cot",
  "dimensions": "5cm x 5cm x 95cm each",
  "label": "ProductComponent"
}
```

```
{
  "supplierName": "EcoWood Partners GmbH",
  "supplierID": "MS_002",
  "instanceID": "MaterialSupplier_EcoWoodPartners",
  "leadTimeCapability": "2 days average",
  "location": "Mannheim, Germany",
  "certifications": "FSC, PEFC, ISO 9001",
  "label": "MaterialSupplier"
}
```

```
{
  "unitPrice": "€250",
  "instanceID": "Product_BespokeBabyCot_2025_001",
  "productID": "PROD_2025_001",
  "color": "Natural Finish",
  "expectedLeadTime": "7 days",
  "productName": "Bespoke Baby Cot",
  "productCategory": "Furniture – Infant",
  "dimensions": "120cm x 60cm x 95cm",
  "safetyStandard": "EN 716-1/2",
  "label": "Product"
}
```

```
{
  "supplierName": "ForestLine Timber AG",
  "supplierID": "MS_003",
  "instanceID": "MaterialSupplier_ForestLineTimber",
  "leadTimeCapability": "5 days average",
  "location": "Zurich, Switzerland",
  "certifications": "FSC, ISO 45001",
  "label": "MaterialSupplier"
}
```

```
{
  "supplierName": "GreenWood Supplies Ltd.",
  "supplierID": "MS_001",
  "instanceID": "MaterialSupplier_GreenWoodLtd",
  "leadTimeCapability": "3 days average",
  "location": "Freiburg, Germany",
  "certifications": "FSC, ISO 14001",
  "label": "MaterialSupplier"
}
```

```
{
  "quantityRequired": "1 kit per cot",
  "componentID": "COMP_2025_FK_001",
  "instanceID": "Component_CotFasteningKit_2025_001",
  "componentCategory": "Hardware/Fasteners",
  "componentName": "Cot Fastening Kit (20 screws + brackets)",
  "usageLocation": "For assembling cot legs and side rails",
  "label": "ProductComponent"
}
```

```
{
  "supplierName": "HardwareCo GmbH",
  "supplierID": "CS_001",
  "instanceID": "ComponentSupplier_HardwareCo",
  "contactEmail": "t.mueller@hardwareco.de",
  "contactPerson": "Thomas Müller",
  "leadTimeCapability": "2 days average",
  "location": "Nuremberg, Germany",
  "certifications": "ISO 9001, ISO 14001",
  "label": "ComponentSupplier"
}
```

FIGURE 5.GRAPH DATA FOR THE USE CASES

11. REFERENCES

[1] Garcez, A. S., Besold, T. R., De Raedt, L., Földiák, P., Hitzler, P., Icard, T., ... & Silver, D. L. (2019). *Neuro Symbolic AI: The 3rd Wave*. arXiv:1905.06088.

[2] Besold, T. R., d'Avila Garcez, A., Bader, S., Bowman, H., Domingos, P., Hitzler, P., ... & Zaverucha, G. (2017). *Neural Symbolic Learning and Reasoning: A Survey and Interpretation*. Frontiers in Artificial Intelligence and Applications.

[3] Dong, L., & Lapata, M. (2016). Language to Logical Form with Neural Attention. ACL.

[4] Zettlemoyer, L. S., & Collins, M. (2005). *Learning to Map Sentences to Logical Form*. UAI.

[5] Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). *Semantic Parsing on Freebase from Question Answer Pairs*. EMNLP.

[6] Huang, L., Liang, C., & Yu, T. (2023). UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models. ICLR.

[7] Yao, S., Zhao, M., Yu, D., Wang, J., Tian, Y., Narasimhan, K., & Kim, J. (2022). *ReAct: Synergizing Reasoning and Acting in Language Models*. arXiv:2210.03629.

[8] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Le, Q. V., ... & Zhou, D. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv:2201.11903.

[9] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., & Bhosale, S. (2023). *Toolformer: Language Models Can Teach Themselves to Use Tools*. NeurIPS.

[10] Lewis, P., Oguz, B., Karpukhin, V., Wang, W. Y., Chen, D., Khandelwal, U., & Zettlemoyer, L. (2020). *Retrieval-Augmented Generation for Knowledge Intensive NLP Tasks*. NeurIPS.

[11] Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th edition). Pearson.

[12] De Raedt, L., Dumančić, S., & Manhaeve, R. (2020). *Neuro Symbolic Programming*. AAAI.

[13] Gioachin, Fabio, Ronchetti, Matteo, NARRATE (2024). D6.1 AI Platform & Intelligent Manufacturing Custiodian Early Release (R1) 2025/11/01

[14] Papazoglou, Michael, Elgammal, Amal, Kraemer, Bernd. NARRATE (2025). D1.4 ARCHITECTURAL REQUIREMENTS

[15] Vaswani, A., et al. (2017). Attention Is All You Need. NeurIPS.

[16] Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. NeurIPS.

[17] Wang, B. Z., Shin, R., Liu, X., Polozov, O., & Richardson, M. (2018). Execution-Guided Neural Program Decoding. ICLR.

Funded by the European Union

Funded by the European Union under GA No 101138094. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. • Grant Agreement No.: 101138094
Call: HORZON-CL4-2023-TWIN-TRANSITION-01 • Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07 • Type of action: HORIZON Research and Innovation Actions

UK Research and Innovation
UK Research and Innovation, Horizon Europe Guarantee.