ITUPublications

International Telecommunication Union
Standardization Sector

ITU Focus Group Technical Specification

(01/2024)

Focus Group on Autonomous Networks
(FG-AN)

Knowledge management for autonomous networks

Technical Specification ITU FG-AN

Knowledge management for autonomous networks

Summary

This Technical Specification specifies knowledge management for autonomous networks. Knowledge management for autonomous network is to learn network knowledge by using heterogeneous data, store it through formal representation. It can shorten the processing response time of network tasks, enable the network to make reasonable and globally behaviour on demand, and help autonomous networks realize the goals of exploratory evolution, real-time responsive online experimentation and dynamic adaptation. This Technical Specification covers the following aspects:

– Overview;
– Classification of knowledge;
– Requirements of knowledge management in autonomous networks;
– Knowledge management functional architecture;
– Sequence diagrams of knowledge management in autonomous networks;
– Security consideration.

Keywords

Autonomous networks, knowledge graph, knowledge management.

Note

This is an informative ITU-T publication. Mandatory provisions such as those found in ITU-T Recommendations are outside the scope of this publication. This publication should only be referenced bibliographically in ITU-T Recommendations.

Contributors: Leon Wong
FG-AN chair,
Rakuten Mobile, Japan
E-mail: leon.wong@rakuten.com

Ziting Zhang
China Telecom
China
Tel: +86 10 5090 2483
Email: zhangzt9@chinatelecom.cn

Dan Xu
China Telecom
China
Tel: +86 10 5090 2570
E-mail: xudan6@chinatelecom.cn

Haobin Wang
China Telecom
China
Tel: +86 10 5090 2366
E-mail: wanghb11@chinatelecom.cn

Paul Harvey
University of Glasgow
UK
Tel: +86 10 5090 2483
Email: paul.harvey@glasgow.ac.uk

Vishnu Ram
India
Email: vishnu.n@ieee.org

Xiaojia Song
China Mobile
P.R. China
Email: songxiaojia@chinamobile.com

This Technical Specification has been published as approved by the focus group, without any subsequent editorial review.

Table of Contents

Technical Specification ITU FG-AN

Knowledge management for autonomous networks

1 Scope

This Technical Specification describes knowledge management for autonomous network. This Technical Specification covers the following aspects:

– Overview;
– Classification of knowledge;
– Requirements of knowledge management in autonomous network;
– Knowledge management functional architecture;
– Sequence diagrams of knowledge management in autonomous network;
– Security consideration.

2 References

[ITU-T M.3385] Recommendation ITU-T M.3385 (2023), *Intelligence levels evaluation* framework *of artificial intelligence enhanced telecom operation and management.*

[ITU-T X.1217] Recommendation ITU-T X.1217 (2021), *Guidelines for applying threat intelligence in telecommunication network operation.*

[ITU-T Y.2701] Recommendation ITU-T Y.2701 (2007), *Security requirements for NGN release 1.*

[ITU-T Y.3061] Recommendation ITU-T Y.3061 (2023), *Autonomous networks – Architecture framework.*

[ITU-T Y.3101] Recommendation ITU-T Y.3101 (2018), *Requirements of the IMT-2020 network.*

[ITU-T Y.3601] Recommendation ITU-T Y.3601 (2025), *Big data – Framework and requirements for data exchange.*

[ITU-T Y Suppl. 71] ITU-T Y Suppl. 71 (2022), *ITU-T Y.3000 series – Use cases for autonomous networks.*

[IEEE P2807] IEEE P2807-2022, IEEE Standard for Framework of Knowledge Graphs.

3 Definitions

3.1 Terms defined elsewhere

This Technical Specification uses the following terms defined elsewhere:

3.1.1 attribute [IEEE P2807]: Describable features.

NOTE – An attribute can be expressed by string, numerical value, and other literals.

3.1.2 controller specification [ITU-T Y.3061]: A high-level, non-executable representation of a controller with the metadata corresponding to the mandatory functionality of the controller and a utility function to be achieved.

3.1.3 data cleaning [ITU-T X.1217]: A process to delete irrelevant data and duplicate data in the original data set, to smooth the noise data, and process missing values and outliers.

3.1.4 entity [IEEE P2807]: An object existed independently in the real world.

3.1.5 knowledge [b-ETSI GS ENI 005]: Analysis of data and information, resulting in an understanding of what the data and information mean.

NOTE – Knowledge represents a set of patterns that are used to explain, as well as predict, what has happened, is happening, or is possible to happen in the future; it is based on acquisition of data, information, and skills through experience and education.

3.1.6 knowledge base [ITU-T Y.3061]: An environment which manages storage, querying, export, import, optimization and update of knowledge.

3.1.7 knowledge element [IEEE P2807]: Knowledge unit describing a certain object or concept that is independent and inseparable.

NOTE – Knowledge element is an umbrella name for the terms of an entity, concept (entity type), attribute, relation, relation type, event, or rule in this document.

3.1.8 knowledge graph [IEEE P2807]: assemblies of knowledge elements and their relations described in a structured form.

3.1.9 ontology [IEEE P2807]: A model to describe entity types, entity attributes and relations among them.

NOTE – Ontology is also described as ontology model.

3.1.10 raw data [ITU-T Y.3601]: Data from a data source without any alteration

3.1.11 relation [IEEE P2807]: Connection between entities, entity types, combination of entities or combination of entity types.

3.2 Terms defined in this Technical Specification

None.

4 Abbreviations and acronyms

This Technical Specification uses the following abbreviations and acronyms:

AI Artificial Intelligence

AN Autonomous Networks

DAS Dynamic Adaptation subsystem

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

E2E End to End

EES Exploratory Evolution subsystem

ES Experimentation subsystem

KB Knowledge Base

KBS Knowledge Base subsystem

KM Knowledge Management

QoS Quality of Service

RDF Resource Description Framework

VLAN Virtual Local Area Networks

5 Conventions

In this Technical Specification:

The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Technical Specification is to be claimed.

The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement needs not be present to claim conformance.

The keywords "can optionally" indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option, and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with this Recommendation.

The keywords "original data" and "raw data" can be used interchangeably with this Technical Specification.

6 Overview

Knowledge management (KM) is an established concept that involves various processes such as identifying, organizing, storing, and disseminating information within an organization. However, with the advent of autonomous networks, new demands and expectations have emerged, necessitating the updates in knowledge management practices.

An autonomous network is characterized by its ability to generate, adapt, and integrate controllers in real-time, utilizing network-specific information. It enables exploratory evolution, responsive online experimentation, and dynamic adaptation [ITU-T Y.3061]. To effectively support the objectives of an autonomous network, the presence of a knowledge base is crucial. The knowledge base provides production, maintenance and application of knowledge and serves as a foundation to enable decision-making within other subsystems and components of the autonomous networks.

Within the context of autonomous networks, knowledge management encompasses a range of activities such as storage, querying, exporting, importing, and optimizing knowledge. The ultimate goal of these activities is to ensure that knowledge can be effectively managed and utilized. To achieve the goal, a formal and consensual knowledge representation framework becomes essential, facilitating accurate and meaningful knowledge formation, exchange, and utilization.

In summary, knowledge management within autonomous networks goes beyond the traditional understanding of the concept. It involves the utilization of knowledge graph technology to enhance the performance and capabilities of autonomous networks. By effectively managing knowledge through formal representation, autonomous networks can leverage their knowledge base to make informed decisions, adapt dynamically, and achieve optimal performance in various tasks and scenarios.

7 Classification of knowledge

Classification of knowledge in autonomous networks can enhance organization, accessibility, sharing, knowledge discovery, and maintenance of knowledge, so it is recommended to classify knowledge with some methods. The following provides three classification perspectives for knowledge classification.

7.1 Knowledge classification based on knowledge processing level

The classification of knowledge is based on the level of processing which has been applied to the data. Such processing can include raw data, cleaned data, and applied knowledge, as shown in Figure 7-1.

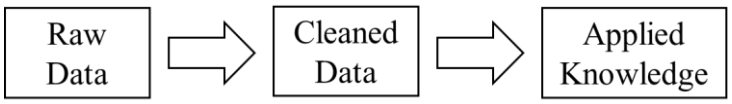

Figure 7-1 – Progressive knowledge processing diagram

- **Raw data** [ITU-T Y.3601]: The original, unprocessed data that is collected or obtained from various sources. Raw data often lacks structure and may contain noise, inconsistencies, or duplicates.
- **Cleaned data**: Cleaned data is the result of data pre-processing, which involves techniques such as data cleaning, data normalization, and handling missing values. In this stage, errors, duplicates, and irrelevant information are removed to improve the quality and reliability of the data. Cleaned data is more suitable for further analysis and application.
- **Applied knowledge**: Applied knowledge is the result based on the processing of cleaned data using knowledge graph for deriving insights and making informed decisions to the knowledge requesters. It includes patterns, models, relationships, and rules derived from data, which can be used for decision-making, prediction, optimization, and other tasks to support practical problem-solving.

7.2 Knowledge classification based on application scenarios

As discussed in the AN use case [ITU-T Y Suppl. 71], the lifecycle of network operation and management in autonomous networks includes planning, construction, optimization, and operation and maintenance scenarios, and involve the following knowledge.

- **Network Planning Scenario**:
  a) **Network requirement analysis**: Analysing the requirements of users and applications to determine required network bandwidth, capacity, and performance.
     – Application requirement analysis: Understanding user and application characteristics, including bandwidth, latency, reliability, etc.
     – Network planning principles: Understanding basic principles of network planning, such as optimal path selection, capacity planning, etc.
     – Business scenario modelling: Modelling based on actual business scenarios to identify key business requirements and network interactions.
  b) **Network topology design**: Designing network topology structure, determining network devices and connectivity methods.
     – Network topology types: Understanding different network topology types, such as star, mesh, ring, or hybrid topologies, and their advantages and disadvantages.
     – Equipment selection: Understanding different types of network devices, including switches, routers, firewalls, etc., and selecting appropriate devices based on requirements.
     – Connection technologies: Understanding different connection technologies, such as Ethernet, fiber optic, etc., and their applicable scenarios and characteristics.
- **Network Construction Scenario**:
  a) Device configuration: Configuring network devices to ensure normal operation according to planning requirements.
     – Device operating systems: Understanding device operating systems, including command-line interface and graphical user interface, resource requirements, licensing, and common commands and configuration methods.

- Interface configuration: Understanding how to configure device interfaces, including IP addresses, subnet masks, default gateways, etc.
- Routing configuration: Understanding basic principles of routing configuration, common routing protocols, and how to configure and manage routing tables.
- VLAN configuration: Understanding the concept and configuration methods of virtual local area networks (VLANs), and how to partition and configure VLANs.

b) Service deployment: Deploying network services such as DHCP, DNS, web servers, etc., to provide corresponding network functionality and services to users and applications.

- Service principles: Understanding the principles and workings of various network services, including DHCP, DNS, web servers, etc., and their deployment and configuration methods in the network.
- Service optimization: Understanding how to optimize and fine-tune services to improve performance, reliability, and security.

• **Network Optimization Scenario**:

a) Network performance optimization: Optimizing network performance and reliability by adjusting network configurations and parameters.

- Traffic analysis tools: Understanding the principles and usage of traffic analysis tools such as Wireshark, NetFlow, etc., to analyse network traffic and identify bottlenecks.
- Bandwidth management: Understanding the principles and methods of bandwidth management, including bandwidth control, traffic shaping, QoS configuration, etc., to optimize bandwidth allocation and utilization.
- Optimization strategies: Understanding common network optimization strategies such as load balancing, link aggregation, etc., to improve network performance and availability.

• **Network Operation and Maintenance Scenario**:

a) Fault troubleshooting: the activities are to implement monitoring, diagnosis and analysis of network and service faults, deal with complaints, generate fault recovery solutions, etc. [ITU-T M.3385].

- Fault localization: Understanding fault troubleshooting methods and tools such as ping, tracert, log analysis, etc., to quickly locate the causes and locations of network faults.
- Fault repair: Understanding fault repair methods and techniques such as device reboot, configuration changes, etc., to restore normal network operation.
- Cutover maintenance: Understanding cutover principles to implement cutover, upgrade and maintenance operations for network equipment, pipeline resources or services [ITU-T M.3385].
- Emergency response: Understanding the process and measures of emergency response, including emergency plans, backup recovery, etc., to minimize the impact of network faults on business operations.

b) Performance monitoring: Monitoring the performance of network devices and links to promptly detect and resolve performance issues.

- Performance metrics: Understanding common network performance metrics such as latency, packet loss rate, bandwidth utilization, etc., to evaluate network performance and health.

- Monitoring tools: Understanding the principles and usage of network monitoring tools such as SNMP, Nagios, etc., to monitor the performance of network devices and links in real-time.
- Alarms and notifications: Understanding the settings and configuration of alarms and notifications, and how to handle and respond to various performance alarms and anomalies.

NOTE – The classification of knowledge in this Technical Specification specific to telecom operation and maintenance domain.

7.3 Knowledge classification based on source of knowledge

According to the source of knowledge, it can be classified into internal knowledge and external knowledge.

- **Internal knowledge** includes knowledge obtained or derived from other subsystems or components in autonomous networks [ITU-T Y.3061], for example:
  i) Data: Structured or unstructured data collected from various data sources involved in a specific use case or obtained through experiments conducted within the autonomous network's sandbox environment.
  ii) Models: Model is representation of the entities of a system, including their relationships and dependencies, using an established set of rules and concepts [b-ETSI GS ENI 005]. The AI/ML models developed and trained to perform specific tasks or provide insights and predictions based on the available data.
  iii) Workflow Representations: Representations of processes or workflows within the network, enabling efficient coordination and interaction between different subsystems or components.
  iv) Policies: Rule sets, guidelines, or protocols that dictate how the autonomous networks should operate, make decisions, and manage the lifecycle of controllers and other entities.
  v) Experimentation: Knowledge obtained from conducting experiments within the sandbox environment, which helps in refining and improving the performance of the autonomous networks.

NOTE 1 – The internal knowledge mentioned in this Technical Specification comes from the reference points defined in AN-arch-fw document [ITU-T Y.3061], including RP-AN-1, RP-AN-2, RP-AN-3, RP-AN-6 and RP-AN-7. These sources serve as the foundation for the internal knowledge presented herein.

- **External knowledge** includes network topology, guide of maintenance manuals, expert experience, etc obtained externally to AN. More specifically,
  i) Network Topology: It refers to the common network structure and related maintenance strategies within different network domains. It includes information about network layouts, configurations, connectivity, and common fault handling procedures.
  ii) Operational Manuals: It is a guide provided by network equipment vendors and operators to standardize the processes of using network resources. They typically cover procedures for software and hardware updates, troubleshooting methods, and other operational aspects.
  iii) Expert Experiences: Expert experiences refer to the professional knowledge and expertise provided by experts. This includes their expertise in managing different network structure, as well as their understanding and experience of network faults.

NOTE 2 – This Technical Specification focuses on internal knowledge. The reference points involved in external knowledge are out of the scope of the present Technical Specification.

8 Requirements of knowledge management in autonomous networks

8.1 Data processing requirements

Data processing is crucial for effective knowledge management, as it allows for the cleaning, integration, and extraction of knowledge from data, ensuring accuracy, completeness, and usability of knowledge.

AN-km-dp-req-001: It is recommended that knowledge base subsystem supports the integration and transformation of multi-source and heterogeneous data, unifying the data into a consistent format and structure for subsequent analysis and processing.

NOTE 1 – Data integration and transformation refer to the process of integrating data from different sources and transforming it into a unified format, structure, or representation. For example, transforming CSV files, database tables, JSON data returned by APIs, etc., into the same data structure, enabling effective data integration and processing.

AN-km-dp-req-002: It is recommended that the knowledge base subsystem support the unification of formats for multi-source and heterogeneous data to ensure consistency and comparability of data during processing and subsequent use.

NOTE 2 – For example, data representing a data should be unified into a specific date format. Other examples include the naming rules for a device's name, the format of IP address, etc.

AN-km-dp-req-003: It is recommended that knowledge base subsystem supports data cleaning, which involves cleaning missing values, error values, duplicate data, etc. [ITU-T X.1217].

AN-km-dp-req-004: It is recommended that knowledge base subsystem supports data linking, bringing together relevant datasets, establishing associations between entities, and providing a consistent and comprehensive view to support more granular querying and analysis.

8.2 Knowledge processing requirements

Knowledge processing is the process of knowledge representation, information extraction, knowledge verification, knowledge storage, knowledge fusion, knowledge reasoning, etc., which means extracting valuable knowledge from processed data, and converting it into a form that can be applied and utilized.

AN-km-kp-req-001: It is required that knowledge base subsystem supports appropriate knowledge representation models to effectively organize and describe knowledge that can be recognized by machine.

NOTE 1 – The knowledge representation models may include graph-based models, ontologies, relational databases, or other applicable models. The knowledge representation models should be capable of representing entities, attributes, relationships, and semantic information to facilitate knowledge querying, retrieval, and inference.

AN-km-kp-req-002: It is recommended that knowledge base subsystem supports the automatic entity extraction, which involves identifying named entities such as person names, location names, organization names, proper nouns, and so on, from textual data.

AN-km-kp-req-003: It is recommended that knowledge base subsystem supports automatic identification and extraction of relationships between entities from textual data.

AN-km-kp-req-004: It is recommended that knowledge base subsystem supports the automatic extraction of entity attributes from textual data.

NOTE 2 – Examples of these attributes can include features, characteristics, and states of the entities.

AN-km-kp-req-005: It is recommended that knowledge base subsystem supports knowledge fusion by merging entities representing the same concept.

AN-km-kp-req-006: It is recommended that knowledge base subsystem supports knowledge fusion by merging knowledge graph constructed from different sources into one.

NOTE 3 – Entity disambiguation, conflict detection, consistency checking, knowledge reasoning, etc. are recommended for knowledge fusion.

AN-km-kp-req-007: It is recommended that knowledge base subsystem supports the verification of knowledge graph quality such as the correctness, timeliness, and completeness of knowledge.

NOTE 4 – The following is an example to clarify the importance of timeliness of knowledge. Path selection for network traffic is based on the current state of network topology and performance. However, the network environment is dynamic and may change due to failures, fluctuations in traffic, or other factors. Such changes can impact the effectiveness and performance of path selection, making timeliness a key factor directly related to the quality of the knowledge base and the success of operations.

AN-km-kp-req-008: It is recommended that knowledge base subsystem supports the improvement of knowledge graph quality (e.g., knowledge graph confidence and accuracy).

NOTE 5 – Knowledge reasoning is recommended for improving the quality of knowledge graph. Inferring missing information in the knowledge base subsystem based on existing triplets to improve the completeness of knowledge; Knowledge reasoning can also be used to infer contradictory relationships, and then correct the knowledge in the knowledge base subsystem to improve the correctness of the knowledge.

8.3 Knowledge management requirements

Knowledge management aims to effectively manage knowledge, to make it easy to find, retrieve, and use.

AN-km-req-001: It is required that knowledge base subsystem supports the storage of knowledge.

NOTE 1 – The storage methods of knowledge may include relational databases, graph databases, and RDF (Resource Description Framework) triplet databases. It is recommended to choose based on the actual requirement.

AN-km-req-002: It is required that knowledge base subsystem supports the query of knowledge.

NOTE 2 – It is recommended to support queries based on keywords, tags, time ranges, and other conditions.

AN-km-req-003: It is required that knowledge base subsystem supports the import and export of knowledge.

AN-km-req-004: It is required that knowledge base subsystem supports the deletion of knowledge.

AN-km-req-005: It is required that knowledge base subsystem supports the optimization of knowledge.

NOTE 3 – Examples of optimizations applied to the knowledge bases are access policies, granularity of storage, interconnection between various knowledge bases and addition of new knowledge.

NOTE 4 – This requirement refers to AN-UC01-REQ-002 [ITU-T Y Suppl. 71].

AN-km-req-006: It is required that knowledge base subsystem supports the update of knowledge based on the various processes involved in the AN.

NOTE 5 – This requirement refers to AN-arch-knw-req-002 [ITU-T Y.3061].

AN-km-req-007: It is recommended that knowledge base subsystem supports automatic updating of knowledge.

NOTE 6 – The examples of automatic knowledge update include: mechanical update, such as adding new values or information in the database, or fusing, merging or inferring existing data to create new knowledge.

AN-km-req-008: It is required that knowledge base subsystem supports the exchange of knowledge between the involved AN components and other entities in the same administrative domain.

NOTE 7 – Other entities include AN components and network services.

NOTE 8 – This requirement refers to AN-UC01-REQ-004 [ITU-T Y Suppl. 71].

AN-km-req-009: It is recommended that knowledge base subsystem supports the exchange of knowledge between AN components and other entities in different administrative domains.

NOTE 9 – The entities in other administrative domains include AN components and network services.

NOTE 10 – This requirement refers to AN-UC01-REQ-006 [ITU-T Y Suppl. 71].

AN-km-req-010: It is required that knowledge base subsystem supports the creation of reports on the use of knowledge bases for consumption by humans and machines.

NOTE 11 – Example of contents of reports are statistics on access of knowledge bases by various AN components as well as network services in the same or different administrative domains.

NOTE 12 – This requirement refers to AN-UC01-REQ-003 [ITU-T Y Suppl. 71].

AN-km-req-011: It is recommended that knowledge base subsystem supports the knowledge update based on the feedback from other AN subsystems and components.

8.4 Interoperability requirements

AN-km-IO-001: It is recommended that knowledge base subsystem supports obtaining data/knowledge from other AN subsystems or components via reference points specified in AN architecture [ITU-T Y.3061].

AN-km-IO-002: It is recommended that knowledge base subsystem supports other subsystems or components to retrieve data/knowledge from knowledge base via reference points.

AN-km-IO-003: It is recommended that knowledge base subsystem supports knowledge synchronization with other subsystems or components via reference points.

NOTE 1 – Synchronization mechanism can be chosen based on the frequency of data/knowledge updates, the importance of data/knowledge changes, and the limitations of system resources. For example, periodic synchronization or triggering mechanisms.

AN-km-IO-004: It is recommended that knowledge base subsystem responds to operation requests for knowledge from other subsystems or components via reference points.

NOTE 2 – Examples of such requests may include knowledge update subscription request or knowledge applications request.

9 Knowledge management functional architecture

9.1 Architecture of knowledge base subsystem

In the AN architecture [ITU-T Y.3061], knowledge management is the responsibility of the knowledge base subsystem, as shown in Figure 9-1. The following elaborates on the high-level description in the architecture by providing a more detailed description of the knowledge base subsystem.

Knowledge base is a subsystem which manages storage, querying, export, import and optimization and update knowledge, including that derived from different sources including structured or unstructured data from various components or other subsystems. AN workflows, including exchange of knowledge between AN components, may in turn result in update of knowledge base.

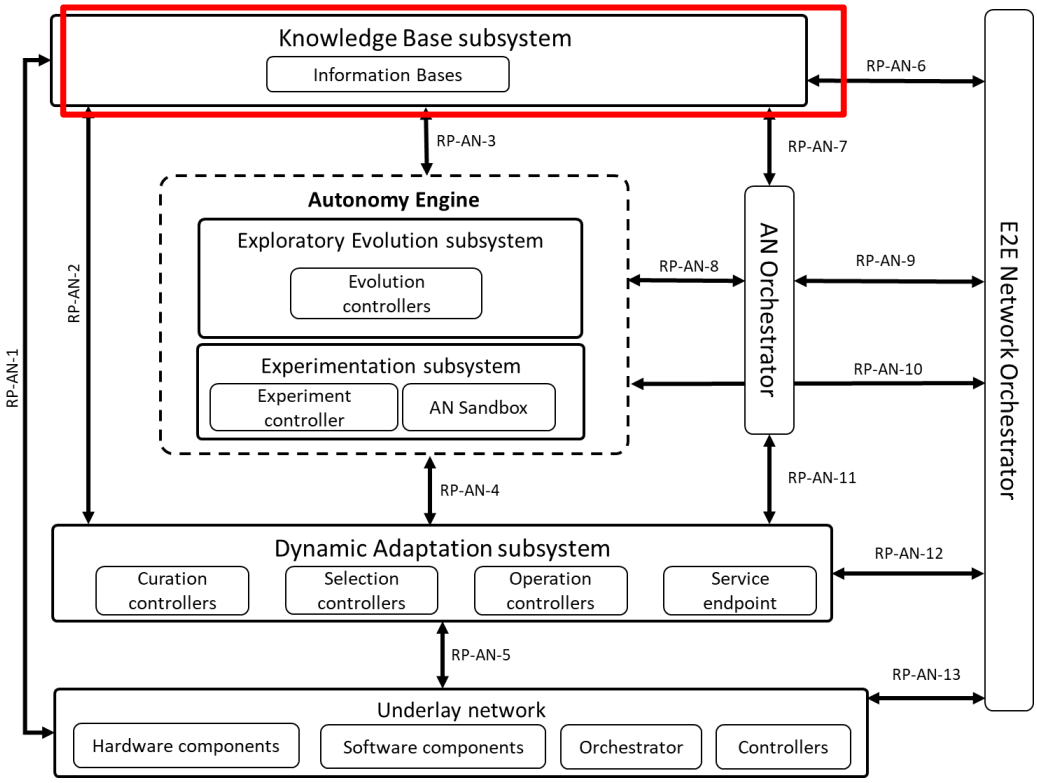

Figure 9-1 – The position of knowledge management in AN architecture

9.2 Detailed functionalities

The detailed knowledge management functionalities in knowledge base system are shown in Figure 9-2, which consists of heterogeneous data processing, knowledge processing controller, knowledge lifecycle management and information bases. The detailed functionality is described in the following clauses.

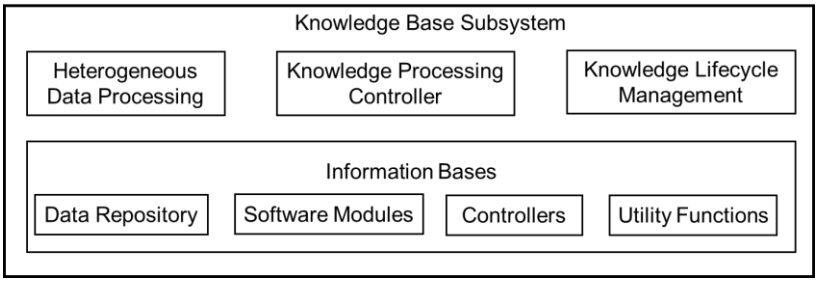

Figure 9-2 – Detailed knowledge management functionalities in knowledge base subsystem

9.2.1 Heterogeneous data processing

The data gathered by autonomous networks is multi-source and heterogeneous, which means the data comes from multiple systems and the data collected by various manufactures without a unified standard. These data may be 1) highly organized and neatly formatted structured data; 2) or semi-structured data, where the model structures that do not conform to the form of relational data tables, but contain relevant tags, which are used to separate semantic elements and to layer records and fields; 3) or unstructured data, such as intent expressed by natural languages [b-ETSI GS ENI 005].

Multi-source and heterogeneous data cannot be directly used for subsequent analysis, reasoning, and decision-making. Thus, it is necessary to process multi-source heterogeneous data to improve data availability.

The heterogeneous data processing functionality pre-processes the data collected by the autonomous network (Data cleansing, data filtering, etc.) and converts to a unified format, and sends the processed available data to the knowledge processing controller.

9.2.2 Knowledge processing controller

Knowledge processing controller includes a series of capabilities such as knowledge representation, information extraction, knowledge verification, knowledge storage, knowledge fusion, knowledge reasoning, etc. The knowledge processing controller receives processed data and constructs high-quality knowledge graphs, thus can provide knowledge search, knowledge recommendation, network decision-making knowledge, etc. for use by other subsystems and components in the autonomous networks.

We now describe the capabilities of the knowledge processing controller:

- **Knowledge representation**: Knowledge representation defines mechanisms for the representation of the characteristics and behaviours of the set of entities being modelled; this enables the autonomous networks to plan actions and determine consequences by reasoning using the knowledge representation, as opposed to taking direct action on the set of entities [b-ETSI GS ENI 005].

  NOTE – Graph is not the only way to represent knowledge. There are also structured data represented by table and unstructured data represented by text. Currently, the graph-based approach is commonly used and has good results. RDF (Resource Description Framework) [b-ITU-T F.750], OWL (Web ontology language) [b-ITU-T Y.4563] and other similar knowledge representation methods are recommended for the knowledge representation methods in this Technical Specification.

- **Information extraction**: Information extraction is the process of extracting specific events or factual information from natural language text and other structured, semi-structured or unstructured data (e.g., documents, pictures, audio, video, PDF, etc.). The output of this process usually includes entity, relationship and event information. Information extraction mainly includes three sub-tasks: relationship extraction, Named-entity recognition, and event extraction. It abstracts domain knowledge into entities, relationships, attributes, constraints, etc.
- **Knowledge verification**: Knowledge verification refers to the process of verifying and confirming the information in a knowledge graph. By evaluating data quality, source credibility, logical consistency, and conducting diversity checks, knowledge validation ensures the accuracy and reliability of the knowledge, thus improving the quality and dependability of the knowledge graph.
- **Knowledge storage**: Knowledge storage refers to the process of storing validated knowledge in a knowledge graph in a structured and queryable manner. Through tasks such as data modelling, storage structure design, indexing and retrieval, scalability, and performance optimization, knowledge storage effectively organizes and manages the information in the knowledge graph for convenient access and utilization by users and applications.
- **Knowledge fusion**: Knowledge fusion integrates knowledge from different sources to solve problems such as synonyms, and conflicting information, to obtain unified knowledge. Knowledge fusion supports quickly identifying and eliminating redundant entities and supports subgraph fusion of multi-source knowledge subgraphs, to achieve efficient integration of structural information and semantic information between knowledge graphs. In addition, it needs to provide real-time fusion of new knowledge.
- **Knowledge reasoning**: Automated reasoning and expansion of knowledge are carried out by using semantic relations in the form of knowledge representation to realize automated reasoning and expansion [b-GB/T 42131-2022]. The typical types of reasoning include classification reasoning, synonymous reasoning, etc. In the knowledge reasoning stage, new knowledge is discovered through rule iteration or model representation of constructed graph

and logic iteration of controllers, including knowledge question and answering, association analysis, rule reasoning and representational reasoning, etc.

- **Knowledge evaluation**: Knowledge evaluation is the process of evaluating the knowledge graph quality based on the knowledge utilization feedback. If knowledge evaluation result is poor compared to some pre-defined evaluation metrics (e.g., knowledge graph confidence and accuracy), then knowledge graph needs to be iterated and updated.

9.2.3 Knowledge lifecycle management

Knowledge lifecycle management consists of two phases: i) knowledge graph design time phase and ii) knowledge graph runtime phase.

In the knowledge graph design time phase, knowledge lifecycle management function monitors the knowledge processing and construction whole process including knowledge representation, information extraction, knowledge verification, knowledge storage, knowledge fusion, knowledge reasoning. This phase helps to build and enrich knowledge base.

In the knowledge graph runtime phase, the constructed knowledge graph is utilized in specific autonomous networks scenarios to generate solutions. Examples of such scenarios include generating network planning schemes, network optimization schemes, network fault recovery schemes, etc. In this phase, knowledge application result is required to be fed back to the knowledge processing controller for knowledge evaluation and iterative optimization of the knowledge graph.

Figure 9-3 illustrates the workflow of knowledge lifecycle management.

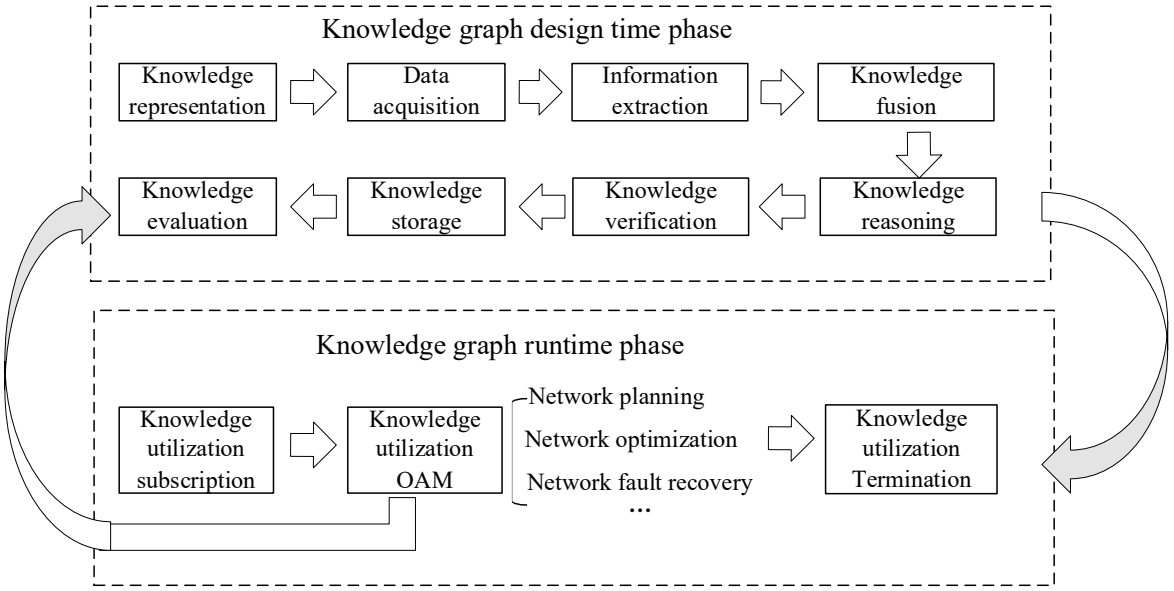

Figure 9-3 – A workflow of knowledge lifecycle management

9.2.4 Information bases for knowledge management

Information base stores the collected data, knowledge, data models, ontologies, policies, etc. to query and use. On the one hand, information base can be continuously updated and serve as a single source of network knowledge for other AN subsystems and be able to provide the required network knowledge for autonomous networks accordingly. On the other hand, based on the network status change, new information generated by other subsystems and components, and newly acquired external knowledge need to be sent to information base for updating the network knowledge.

9.3 Interaction with other subsystems and components

9.3.1 General description

The Autonomous Network Architecture [ITU-T Y.3061] defines/describes the knowledge base subsystem as a subsystem which manages storage, querying, export, import and optimization and update knowledge. Knowledge Base subsystem interacts with Underlay network to collect real-time network data via RP-AN-1 to decide whether to trigger the update of knowledge graph based on the network changes, and interacts with E2E Network Orchestrator to acquire historical network data and provide knowledge which may be used to recommend E2E network orchestration policies via RP-AN-6. Knowledge Base subsystem also can interact with exploratory evolution subsystem (EES), experimentation subsystem (ES), and dynamic adaptation subsystem (DAS) to assist their operation.

9.3.2 Interaction with the exploratory evolution subsystem

From the EES, interactions with the KBS are driven by an evolution controller. During a triggered or periodic evolution of a controller, an evolution controller may request or store raw data or knowledge from the KBS using RP-AN-3.

Examples of raw data to be managed by the KBS for the EES include:

- controller specifications,
- controller descriptions,
- module specifications,
- module descriptions,
- performance and operational data of the EES itself,
- performance data related to generated controller descriptions by the EES.

Examples of knowledge to be managed by the KBS via knowledge graph include:

- knowledge of context for the evolution of a controller considering module composition,
- knowledge of context for the selection of algorithms in the evolution process itself,
- knowledge of context on the current underlay operational status.

The KBS can also use RP-AN-3 to request an evolution in response to the update or optimization of knowledge.

9.3.3 Interaction with the experimentation subsystem

From the ES, interactions with the KBS are driven by an experimentation controller. During the generation of experiments or the execution of experiments, an experimentation controller may request or store raw data or knowledge using RP-AN-3.

Examples of raw data to be managed by the KBS for the ES include:

- controller descriptions,
- results from experiments executed in the sandbox (see clause 8.3.1.2.2 [ITU-T Y.3061]),
- performance and operational data of the ES itself,
- historical experimental configuration data,
- inventory of tools available within the sandbox to be used in experimentation.

NOTE –Examples of such configurations can include configuration of tools in the sandbox.

Examples of knowledge to be managed by the KBS via knowledge graph for the ES include:

- knowledge of context for experimental generation for a controller,
- knowledge of guidance on configuration of tools in the sandbox.

9.3.4 Interaction with dynamic adaptation subsystem

From the DAS, interactions with the KBS are driven by curation controllers and selection controllers. During the curation of controller as candidates for deployment or the selection of controller for deployment in the underlay, both the curation and selection controllers may request or store data or knowledge using RP-AN-2.

Examples of raw data to be managed by the KBS for the curation controller include:

- currently curated operational controllers for a given use case,
- currently validated operational controllers for a given use case,
- operational and performance data for the curation controller itself.

Examples of raw data to be managed by the KBS for the selection controller include:

- currently deployed operation controller,
- available curated operational controllers that are eligible for deployment,
- operational and performance data for the selection controller itself.

Examples of knowledge to be managed by the KBS via knowledge graph for the curation controller include:

- knowledge of the effectiveness of current set of curated controllers,
- knowledge of semantic descriptions of operational controllers.

Examples of knowledge to be managed by the KBS via knowledge graph for the selection controller include:

- knowledge of the current operational environment context,
- knowledge of past performance of operational controllers in the underlay,
- knowledge of past performance of operational controller during experimentation.

9.3.5 Knowledge Interaction with Controllers

9.3.5.1 Controller interaction with knowledge in the KBS in centralized deployment mode

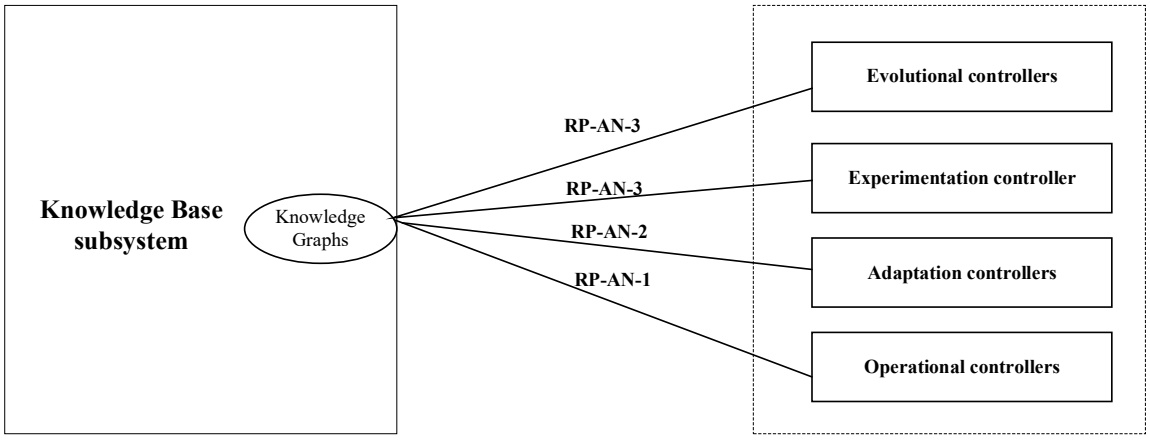

Figure 9-4 – Centralized deployment of knowledge graph

Considering Figure 9-4, a controller may interact with knowledge stored within the KBS to correctly and effectively achieve its goal. Here, the capabilities of knowledge graphs are centralized deployed in KBS and are external to the controllers. To access this information, different controllers would use different reference points:

- Controllers deployed in the underlay network would interact with knowledge stored within the KBS via interface RP-AN-1.

- Controllers deployed in the Autonomy Engine would interact with knowledge stored within the KBS via interface RP-AN-3 for exchanging raw data and knowledge specified in clauses 9.3.2 and 9.3.3.
- Controllers deployed in the Dynamic Adaptation Subsystem would interact with knowledge stored within the KBS via interface RP-AN-2 for exchanging raw data and knowledge specified in clause 9.3.4.

9.3.5.2 Controller interaction with knowledge as an internal module in distributed deployment mode

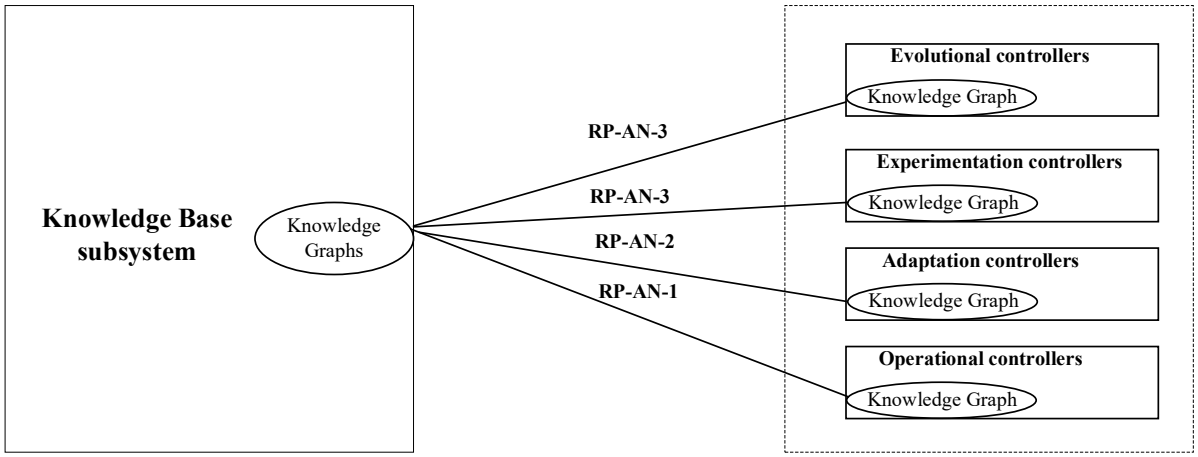

Figure 9-5 – Distributed deployment of knowledge graph integrated in controllers

Considering Figure 9-5, a controller may interact with knowledge generated by the KBS, where this knowledge is integrated in the controller. Here, as self-contained unit of knowledge, such as a knowledge graph, is considered. According to the AN arch, this knowledge graph would be represented as a module which may integrated into the controller. In this deployment mode, controllers can access knowledge by internally module interaction, and controllers will also send the data to KBS for possible knowledge graph iteration. Once knowledge graph is updated, KBS will push the updated knowledge graph to controllers to trigger the corresponding module synchronous update.

10 Sequence diagrams of knowledge management in AN

10.1 General description of knowledge management in AN

Knowledge management in AN mainly includes the design time phase and runtime phase of knowledge graph based on clause 9.2.3. The sequence diagrams of design time procedures consist of network knowledge graph construction and controller knowledge graph construction based on historical network data and controller specifications, which are described in clause 10.2. The sequence diagrams of runtime procedures consist of the request/subscription of network knowledge and knowledge for controllers based on real-time requirements, which are described in clause 10.3.

10.2 Sequence diagrams of knowledge management in knowledge graph design time phase

10.2.1 Sequence diagram of knowledge management in network knowledge graph design time phase

As shown in Figure 10-1, network knowledge graph construction involves heterogeneous data acquisition from the E2E Network Orchestrator, and then Knowledge Base subsystem processes the data to model and construct the network knowledge graph.

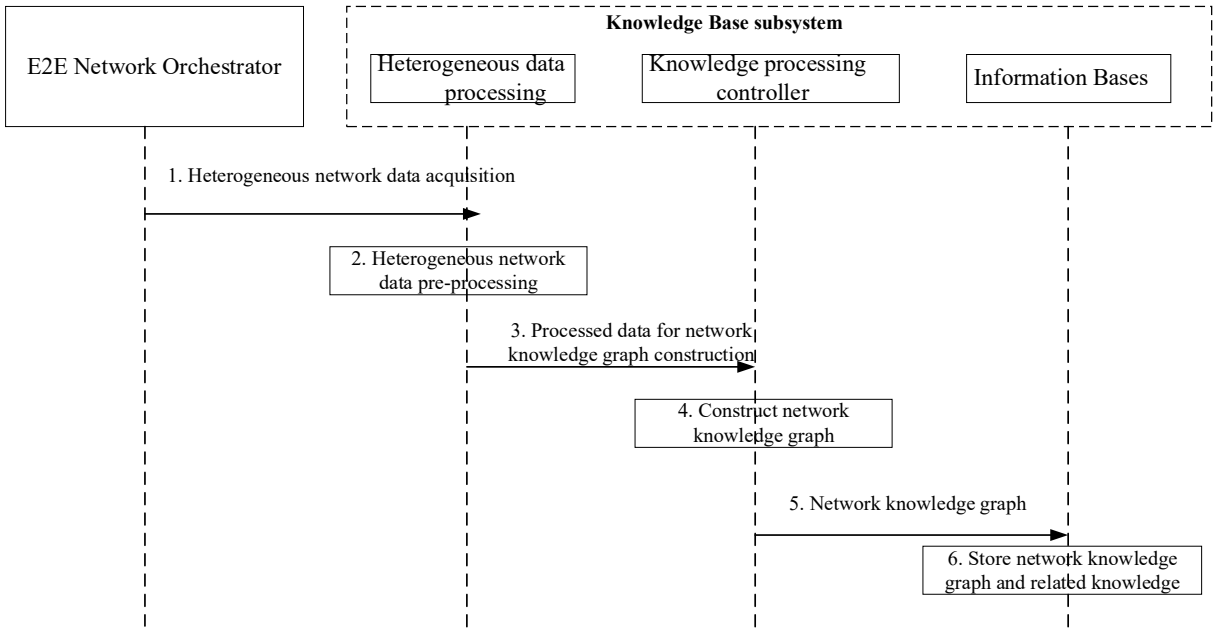

Figure 10-1 – Sequence diagram of knowledge management in network knowledge graph design time phase

The steps involved in Figure10-1 are:

1 Knowledge base subsystem acquires historical heterogeneous network data from E2E Network Orchestrator.

2 Heterogeneous data processing function of Knowledge Base subsystem pre-processes network data includes data classification, normalization, etc.

3 Heterogeneous data processing function sends the processed data to knowledge processing controller for constructing network knowledge graph.

4 Knowledge processing controller determines knowledge representation mechanism, and implement information extraction, knowledge fusion, knowledge reasoning, and knowledge verification to construct network knowledge graphs to specific network scenarios.

5 Knowledge processing controller sends the constructed network knowledge graphs to Information Bases of Knowledge Base subsystem.

6 Information Bases store the constructed network knowledge graphs and related knowledge for use in subsequent requests.

NOTE – For updating the existing knowledge graphs, it is needed to acquire updated data set in the step 1, and execute the following step 2 to step 6 for knowledge graphs retraining and updating.

10.2.2 Sequence diagram of knowledge management in controller knowledge graph design time phase

As shown in Figure 10-2, controller knowledge graph construction involves acquisition of exploratory evolution logic, experimentation logic and dynamic adaptation logic from Autonomy Engine and Dynamic Adaptation Subsystem, and then Knowledge Base subsystem constructs the controller knowledge graph based on controller specification and information.

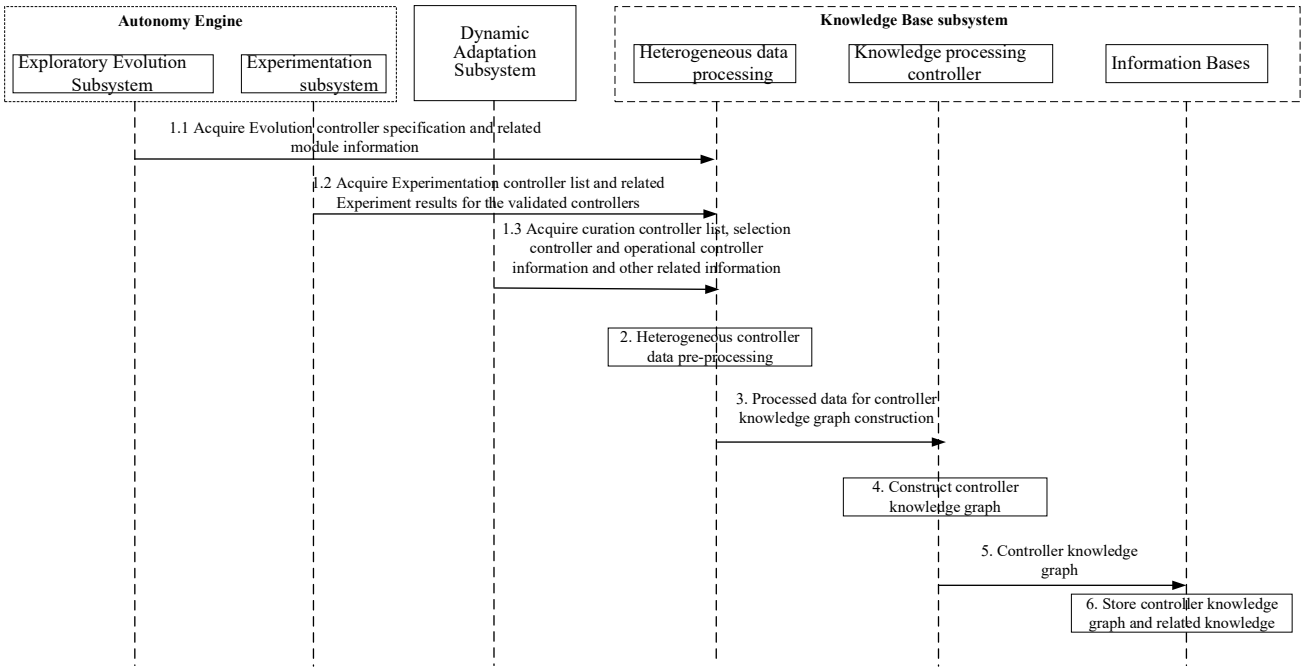

Figure 10-2 – Sequence diagram of knowledge management in controller knowledge graph design time phase

The steps involved in Figure10-2 are:

1 Knowledge base subsystem acquires controller data, includes:

1.1 Knowledge base subsystem acquires Evolution controller specification and related module information from Exploratory Evolution Subsystem;

1.2 Knowledge base subsystem acquires Experimentation controller list and related experiment results for validated controller from Experimentation Subsystem;

1.3 Knowledge base subsystem acquires curation controller list, selection controller and operation controller information, as well as other related information from Dynamic Adaptation Subsystem.

2 Heterogeneous data processing function of Knowledge Base subsystem pre-processes controller data includes data classification, normalization, etc.

3 Heterogeneous data processing function sends the processed controller data to knowledge processing controller for constructing controller knowledge graphs.

4 Knowledge processing controller determines knowledge representation mechanism, and implement controller information extraction, knowledge fusion, knowledge reasoning, and knowledge verification to construct controller knowledge graphs.

5 Knowledge processing controller sends the constructed controller knowledge graphs to Information Bases of Knowledge Base subsystem.

6 Information Bases store the constructed controller knowledge graph and related knowledge for use in subsequent requests.

10.3 Sequence diagrams of knowledge management in knowledge graph runtime phase

10.3.1 Sequence diagram of knowledge management in network knowledge graph runtime phase

As shown in Figure 10-3, network knowledge graph utilization means that E2E Network Orchestrator acquires E2E network diagnose knowledge from Knowledge Base subsystem processes and distributes network optimization and configuration information to the Underlay network.

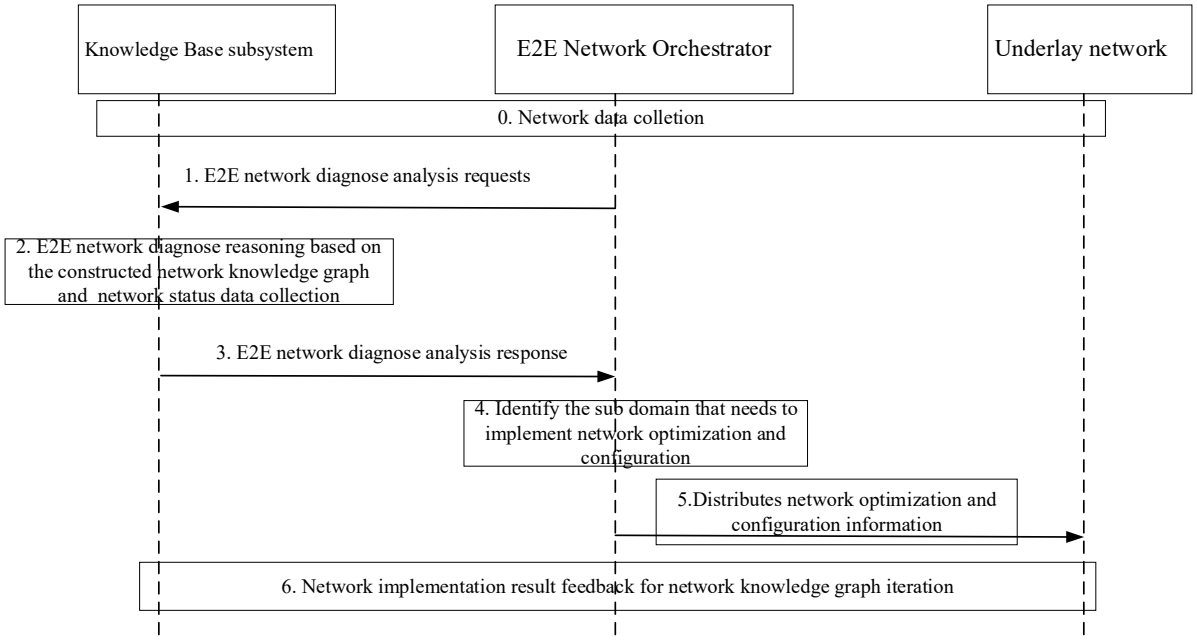

Figure 10-3 – Sequence diagram of knowledge management in network knowledge graph runtime phase

The steps involved in Figure10-3 are:

0 Network data is collected to Knowledge base subsystem from Underlay network.

1 E2E Network Orchestrator sends E2E network diagnose analysis request to Knowledge base subsystem.

2 Knowledge base subsystem executes E2E network diagnose reasoning based on the constructed network knowledge graph and real-time network status data collection. For example, for a real-time alarm record, Knowledge base subsystem can reason the current alarm, associated alarm, root cause alarm relationship and related devices.

3 Knowledge base subsystem sends the E2E network diagnose analysis response to E2E Network Orchestrator.

4 E2E Network Orchestrator identifies the sub domain that needs to implement network optimization and configuration based on E2E network diagnose analysis result.

5 E2E Network Orchestrator distributes network optimization and configuration information to Underlay network.

6 Knowledge base subsystem can iterate and update network knowledge graph based on network implementation result feedback of Underlay network.

10.3.2 Sequence diagram of knowledge management in controller knowledge graph runtime phase

As shown in Figure 10-4, controller knowledge graph utilization indicates that Autonomy Engine and Dynamic Adaptation Subsystem apply knowledge acquired from Knowledge Base subsystem to assist in evolution process, experimentation process and dynamic adaptation process.

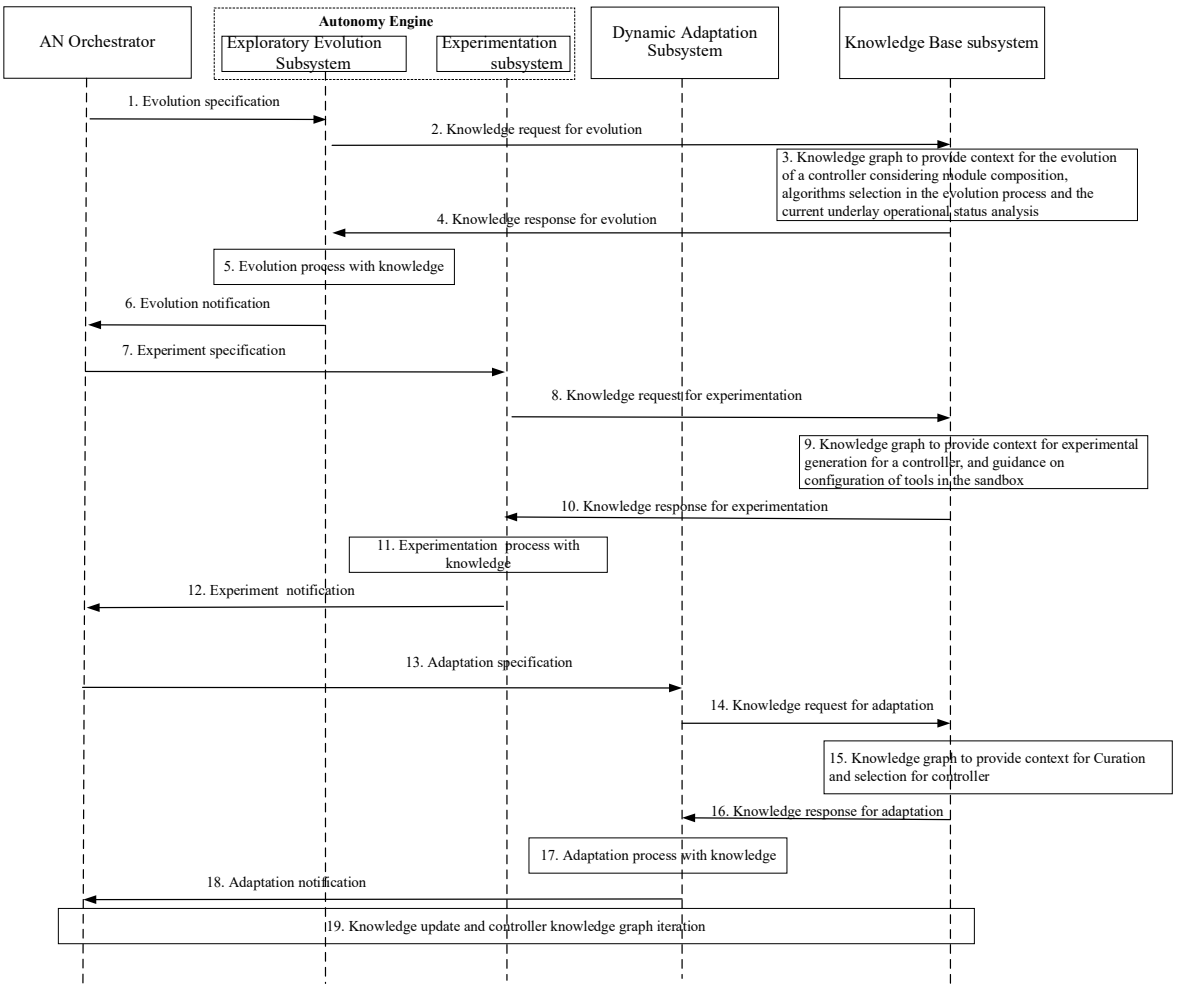

Figure 10-4 – Sequence diagram of knowledge management in controller knowledge graph runtime phase

The steps involved in Figure10-4 are:

1 AN orchestrator sends an evolution specification to Exploratory Evolution subsystem, and the evolution specification is described in clause 9.1of [ITU-T Y.3061].

2 Exploratory Evolution subsystem sends knowledge request for evolution to Knowledge base subsystem.

3 Knowledge base subsystem can analyse module composition of a controller for the evolution, provide algorithms selection in the evolution process and the current underlay operational status analysis with the constructed controller knowledge graph.

4 Knowledge base subsystem responds the knowledge for evolution in step 3 to Exploratory Evolution subsystem

5 Exploratory Evolution subsystem implements Evolution process with knowledge.

6 Exploratory Evolution subsystem notifies Evolution result to AN orchestrator.

7 AN orchestrator sends Experiment specification to Experimentation subsystem.

8 Experimentation subsystem sends knowledge request for experimentation to Knowledge base subsystem.

9 Knowledge base subsystem can provide context for experimental generation for a controller, and guidance on configuration of tools in the sandbox with the constructed controller knowledge graph.

10 Knowledge base subsystem responds the knowledge for experimentation in step 9 to Experimentation subsystem.

11 Experimentation subsystem implements Experimentation process with knowledge.

12 Experimentation subsystem notifies Experimentation result to AN orchestrator.

13 AN orchestrator sends Adaptation specification to Dynamic Adaptation Subsystem.

14 Dynamic Adaptation Subsystem sends knowledge request for adaptation to Knowledge base subsystem.

15 Knowledge base subsystem can provide knowledge of the effectiveness of current set of curated controllers, semantic descriptions of operational controllers, current operational environment context as well as past performance analysis of operational controllers in the underlay or during experimentation with the constructed controller knowledge graph.

16 Knowledge base subsystem responds the knowledge for adaptation in step 15 to Dynamic Adaptation Subsystem

17 Dynamic Adaptation Subsystem implements Adaptation process with knowledge.

18 Dynamic Adaptation Subsystem notifies the Adaptation result to AN orchestrator.

19 Exploratory Evolution Subsystem, Experimentation subsystem, Dynamic Adaptation Subsystem and AN orchestrator will give the feedback for knowledge update and controller knowledge graph iteration if needed.

11 Security considerations

This Technical Specification describes the knowledge management for autonomous networks which is expected to be meet general network security requirements and mechanisms in IP-based networks [ITU-T Y.2701] [ITU-T Y.3101] and specific security considerations concern autonomous networks [ITU-T Y.3061]. Specifically, the major security requirements for knowledge management in autonomous network are listed in the following:

[REQ-SM-1] It is required to support encrypting the transmission and storage of sensitive data (such as personal information, sensitive service data, etc.) in the knowledge graph, and support to control data access via encryption and decryption methods;

[REQ-SM-2] It is required to support restricting the transfer of data in the knowledge graph to specific authorized entities;

[REQ-SM-3] It is required to ensure the confidentiality, integrity and availability of data in knowledge graph;

[REQ-SM-4] It is required to ensure the knowledge cannot be illegally obtained by unauthorized users.

Appendix I

Requirements for knowledge in AN architecture

The following are requirements with respect to knowledge in autonomous networks [ITU-T Y.3061].

Requirement	Description
AN-arch-knw-req-001	The AN architecture is required to provide capabilities for the management of knowledge related to autonomous networks. NOTE – Managing knowledge includes storing, querying, exporting, importing, and optimizing knowledge.
AN-arch-knw-req-002	The AN architecture is required to enable the update of knowledge based on the various processes involved in the AN.
AN-arch-knw-req-003	The AN architecture is required to support the utilization of components like stored controllers and knowledge to deploy and manage controllers in underlay networks. NOTE – Examples of components include stored controllers and knowledge.
AN-arch-knw-req-004	The AN architecture is required to enable the storage and management of supporting artifacts for the lifecycle management of controllers. NOTE 1 – Examples of supporting artifacts are knowledge, AI/ML or other types of models, workflow representations, policies which need to be applied while managing the lifecycle of controllers, etc. NOTE 2 – Examples of management of supporting artifacts are storage of knowledge in knowledge base, creation, modification, deletion, and storage of AI/ML models in ML model repository [b-ITU-T Y.3176] and of policies, query and discovery of various artifacts.
AN-arch-knw-req-005	The AN architecture is required to support the production of human and machine-readable reports of periodic or aperiodic nature.
AN-arch-knw-req-006	The AN architecture is recommended to enable the analysis and correlation of domain specific, unstructured data in natural languages from the underlay networks. NOTE 1 – Examples of domain specific unstructured data in natural languages are logs from NFs. NOTE 2 – Advances in analysis of natural language text may be exploited from third party models and repositories.
AN-arch-knw-req-007	The AN architecture is recommended to support capabilities for deriving knowledge from the analysis and correlation of domain specific unstructured data in natural languages from the underlay networks.
AN-arch-knw-req-008	The AN architecture is required to support capabilities for importing and exporting of controller specifications at various stages of their management. NOTE – Examples of various stages of management of controller specifications are before and after exploratory evolution.
AN-arch-knw-req-009	The AN architecture is required to support the integration of third party provided derivation mechanisms for metrics from collected parameters and measurements. NOTE – An example of derived metrics is QoE while an example of collected measurements is QoS parameters.
AN-arch-knw-req-010	The AN architecture is required to support the capturing of both service KPI requirements as well as deployment preferences and considerations in the intent.

Requirement	Description
AN-arch-knw-req-011	The AN architecture is required to provide capabilities for the AN to capture domain specificities. NOTE 1 – Examples of domain specificities are latency criteria, location information, data privacy requirements, etc. NOTE 2 – Examples of capturing domain specificities are TOSCA service definitions. The design of controllers may also be represented using TOSCA declarative definitions.
AN-arch-knw-req-012	The AN architecture is required to provide the ability for the AN to capture service specificities. NOTE – Examples of service specificities include service level requirements for QoS.
AN-arch-knw-req-013	The AN architecture is required to support the AN capability of using inputs from external environment and user specific models to design as well as apply controller outputs to underlay networks. NOTE – Example of inputs from external environments is mobility prediction models for users with assistive needs or groups of users.
AN-arch-knw-req-014	The AN architecture is required to support the AN capability of using user preferences while designing as well as applying controller outputs to underlay networks. NOTE – Standard representations of user profiles or preferences or user models with assistive needs are examples of user preferences.
AN-arch-knw-req-015	The AN architecture is recommended to enable the transfer by the AN of user specific models between different domains in the underlay network. NOTE – This may help in update of models in other domains, update of simulators.
AN-arch-knw-req-016	The AN architecture is required to provide means for the AN to integrate data collection mechanisms. NOTE – Data collection mechanisms may include AR/VR glasses or other types of sensors. Data collection mechanisms may be provided by third-party providers.
AN-arch-knw-req-017	The AN architecture is recommended to provide means for the AN to use virtual models along with the real-world inputs to analyse and optimize the underlay network and to provide feedback to operators. NOTE – Analysis may use AI/ML models. Optimization may involve configurations in the underlay network. Feedback to operators may be generated in AR/VR formats.
AN-arch-knw-req-018	The AN architecture can optionally support the ability to integrate in the AN third-party modules or applications for collection, analysis, or feedback. NOTE – For example a software development kit (SDK) may be exposed to third party developers who may develop new applications to analyse the AR-collected data.
AN-arch-knw-req-019	The AN architecture is required to enable the AN discovery of capabilities available at the various domains of underlay networks. NOTE – Capabilities of the underlay networks may differ based on their resource availability, e.g., compute, memory, already deployed controllers.
AN-arch-knw-req-020	The AN architecture can optionally enable the AN creation of use case descriptions which can then be decomposed to controllers which can be deployed at various domains of the underlay network, based on the capabilities at those levels of the underlay network. NOTE – use case descriptions may be in the form of intents.

Requirement	Description
AN-arch-knw-req-021	The AN architecture is required to support means for the AN to use interoperable format for storing controllers. NOTE – Various components in AN may read and write from the stored controllers, e.g., an evolution controller may read existing controllers (even from third parties) and utilize them for composing new controllers, which are in turn written in the storage.
AN-arch-knw-req-022	The AN architecture can optionally support the ability for the AN to discover the characteristics of underlay networks at runtime.
AN-arch-knw-req-023	The AN architecture is required to support description of use cases in a declarative format.
AN-arch-knw-req-024	The AN architecture is required to support derivation of domain specific intents from the use case description. NOTE – ML intent [b-ITU-T Y.3172] is an example of domain specific intent.

Appendix II

Requirements for knowledge in AN use cases

The following are requirements with respect to knowledge in autonomous networks use cases [ITU-T Y Suppl. 71].

Critical requirements

- AN-UC01-REQ-001: It is critical that the AN enable the exchange of knowledge between the different involved AN components.
- AN-UC01-REQ-002: It is critical that the AN enable the optimization of knowledge bases.

NOTE 1 – Examples of optimizations applied to the knowledge bases are access policies, granularity of storage, interconnection between various knowledge bases and relation between problems and solutions, addition of new knowledge.

- AN-UC01-REQ-003: It is critical that the AN enable the creation of reports on the use of knowledge bases for consumption by humans and machines.

NOTE 2 – Example of contents of reports are statistics on access of knowledge bases by various AN components as well as network services in the same or different administrative domains.

- AN-UC01-REQ-004: It is critical that the AN enable the exchange of knowledge between the involved AN components and other entities in the same administrative domain.

NOTE 3 – Other entities include AN components and network services.

- AN-UC01-REQ-005: It is critical that the AN use one or more knowledge bases for mapping one or more high level use case descriptions to the controller specification.

NOTE 4 – Controller specification may use languages such as TOSCA [b-TOSCA], whereas use case descriptions may be unstructured. The high-level use case description is to be converted to a structured controller specification. This process of "conversion" may utilize the help of humans (using GUIs) who can better understand unstructured information, and/or automated generation techniques.

Expected requirements

- AN-UC01-REQ-006: It is expected that the AN enable the exchange of knowledge between AN components and other entities in different administrative domains.

NOTE 5 – The entities in other administrative domains include AN components and network services.

Added value requirements

- AN-UC01-REQ-007: It is of added value that automated generation techniques be used by the AN to produce controller specifications, using the stored controller descriptions and the knowledge base.

NOTE 6 – For example, a graph neural network (GNN) based recommendation engine may be used to help automatic generation techniques.

Bibliography

[b-ITU-T Y.3172] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in future networks including IMT-2020.*

[b-ITU-T Y.3176] Recommendation ITU-T Y.3176 (2020), *Machine learning marketplace integration in future networks including IMT-2020.*

[b-ITU-T F.750] Recommendation ITU-T F.750 (2005), *Metadata framework.*

[b-ITU-T Y.4563] Recommendation ITU-T Y.4563 (2021), *Requirements and functional model to support data interoperability in Internet of things environments.*

[b-ETSI GS ENI 005] ETSI GS ENI 005, *Experiential Networked Intelligence (ENI).*

[b-GB/T 42131-2022] GB/T 42131-2022, *Artificial Intelligence, knowledge graph technology framework.*