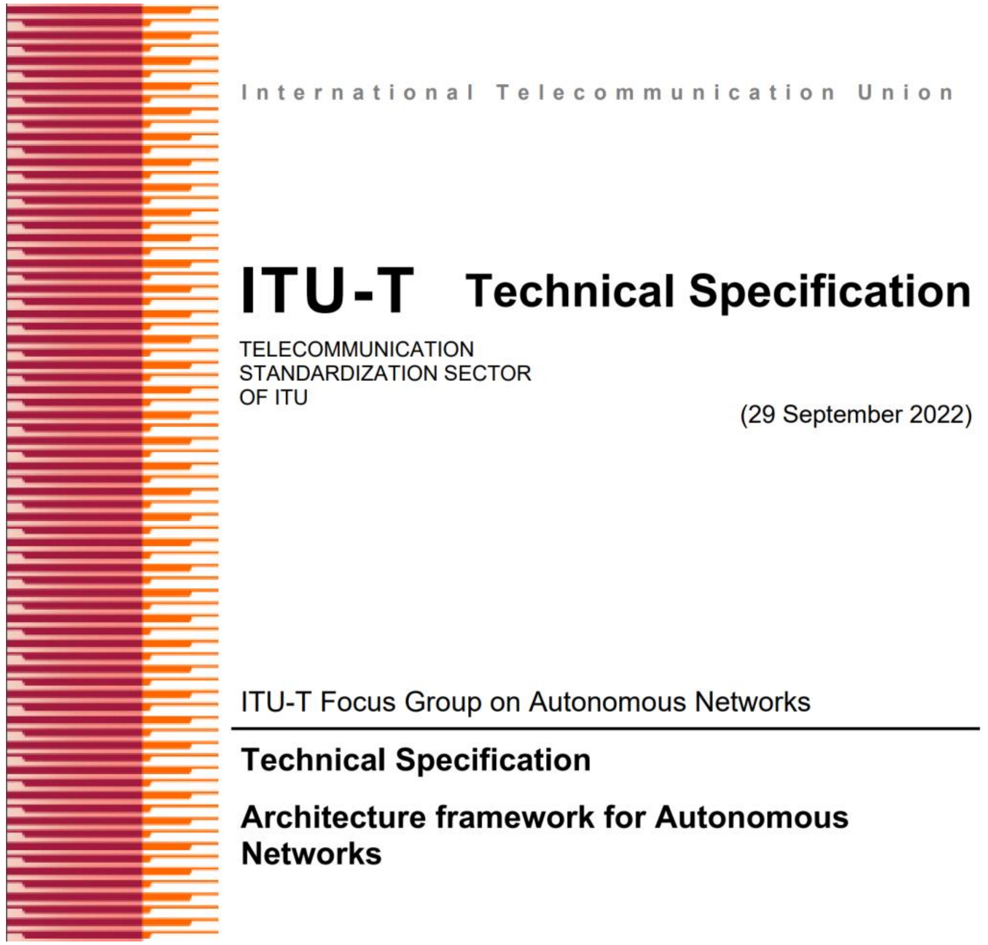
Highlights of architecture concepts and components

ITU-T Focus Group on Autonomous Networks

Paul Harvey
www.paul-harvey.org

FG-AN

Where to find



International Telecommunication Union

ITU-T Technical Specification

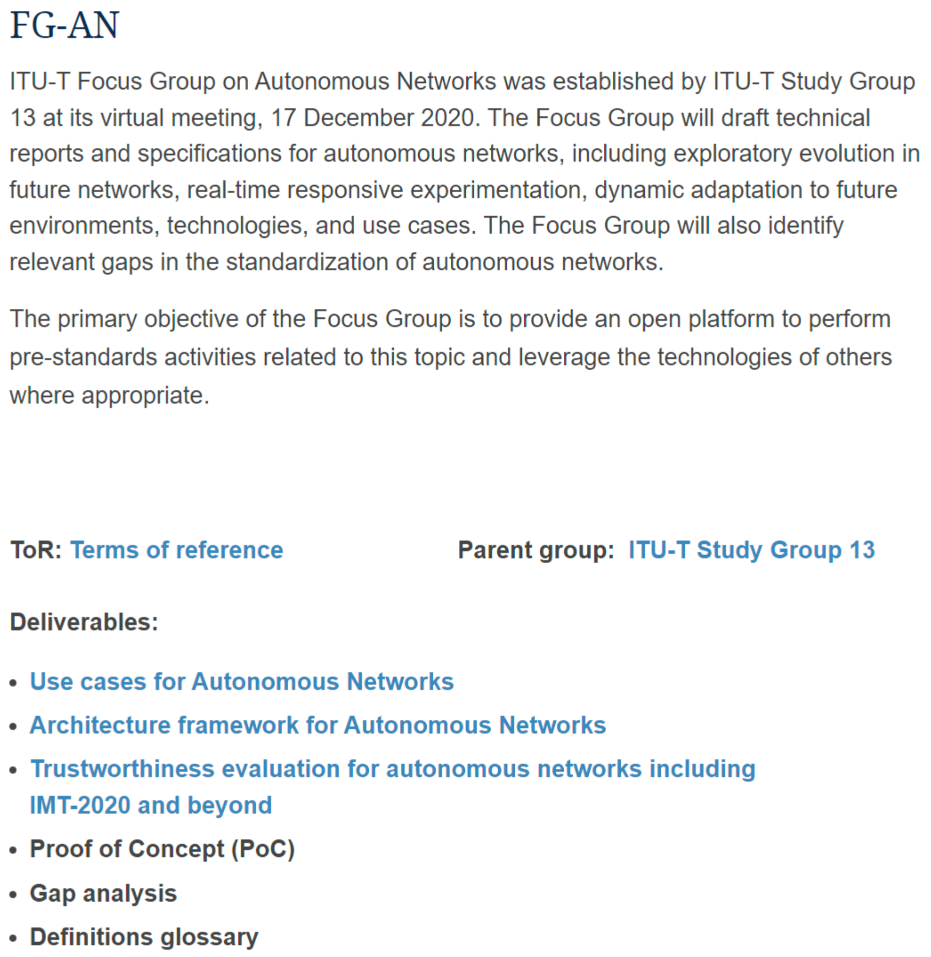
TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(29 September 2022)

ITU-T Focus Group on Autonomous Networks

Technical Specification

Architecture framework for Autonomous Networks



FG-AN

ITU-T Focus Group on Autonomous Networks was established by ITU-T Study Group 13 at its virtual meeting, 17 December 2020. The Focus Group will draft technical reports and specifications for autonomous networks, including exploratory evolution in future networks, real-time responsive experimentation, dynamic adaptation to future environments, technologies, and use cases. The Focus Group will also identify relevant gaps in the standardization of autonomous networks.

The primary objective of the Focus Group is to provide an open platform to perform pre-standards activities related to this topic and leverage the technologies of others where appropriate.

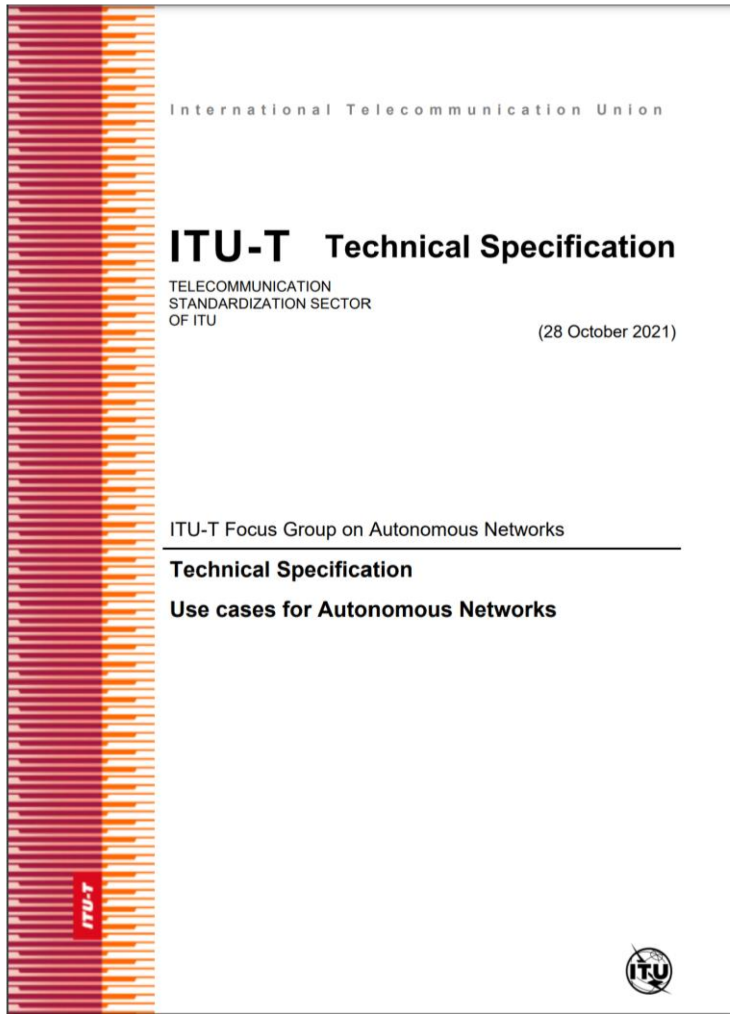
ToR: Terms of reference **Parent group:** ITU-T Study Group 13

Deliverables:

- Use cases for Autonomous Networks
- Architecture framework for Autonomous Networks
- Trustworthiness evaluation for autonomous networks including IMT-2020 and beyond
- Proof of Concept (PoC)
- Gap analysis
- Definitions glossary

https://www.itu.int/en/ITU-T/focusgroups/an/Pages/default.aspx

Process



International Telecommunication Union

ITU-T **Technical Specification**

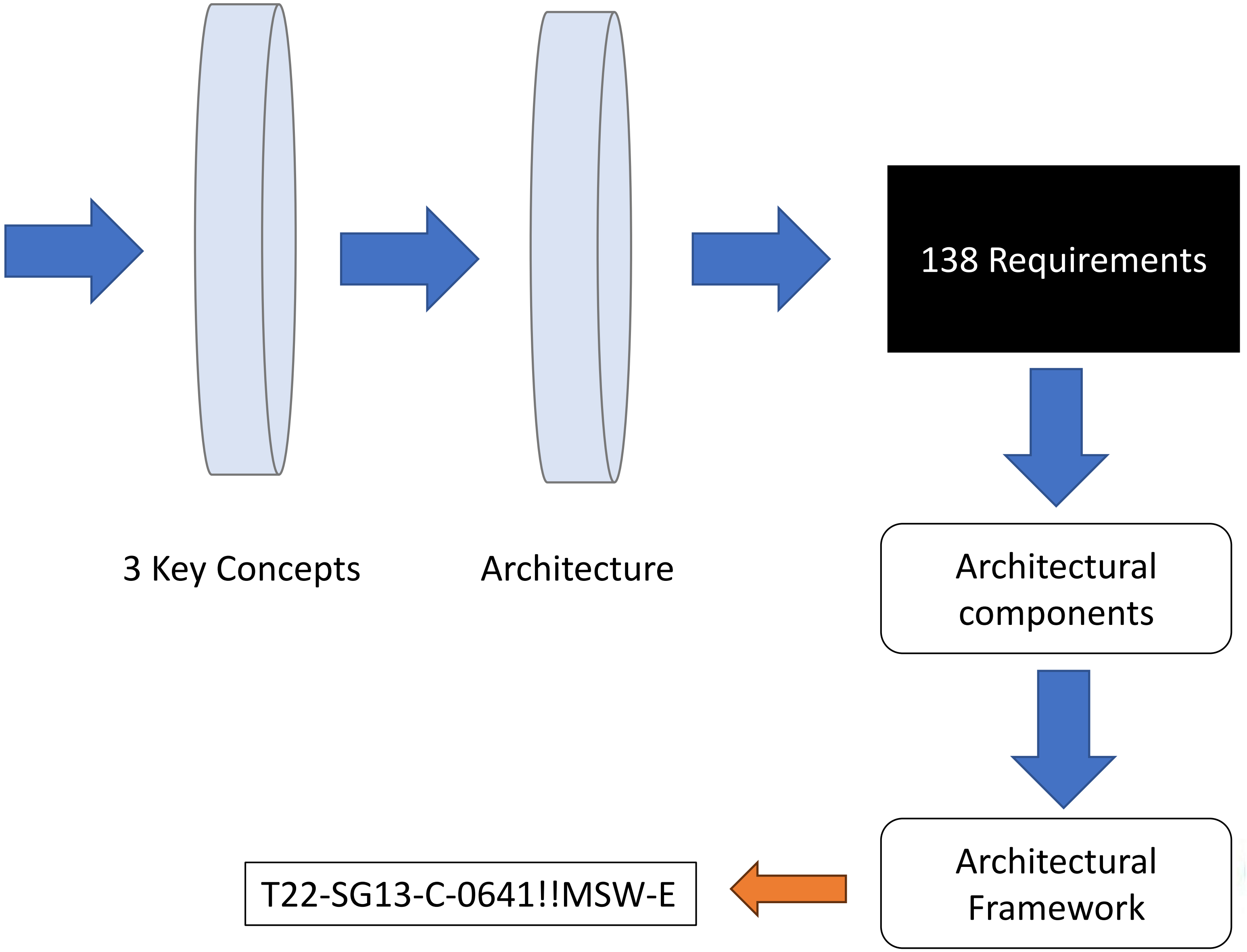
TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(28 October 2021)

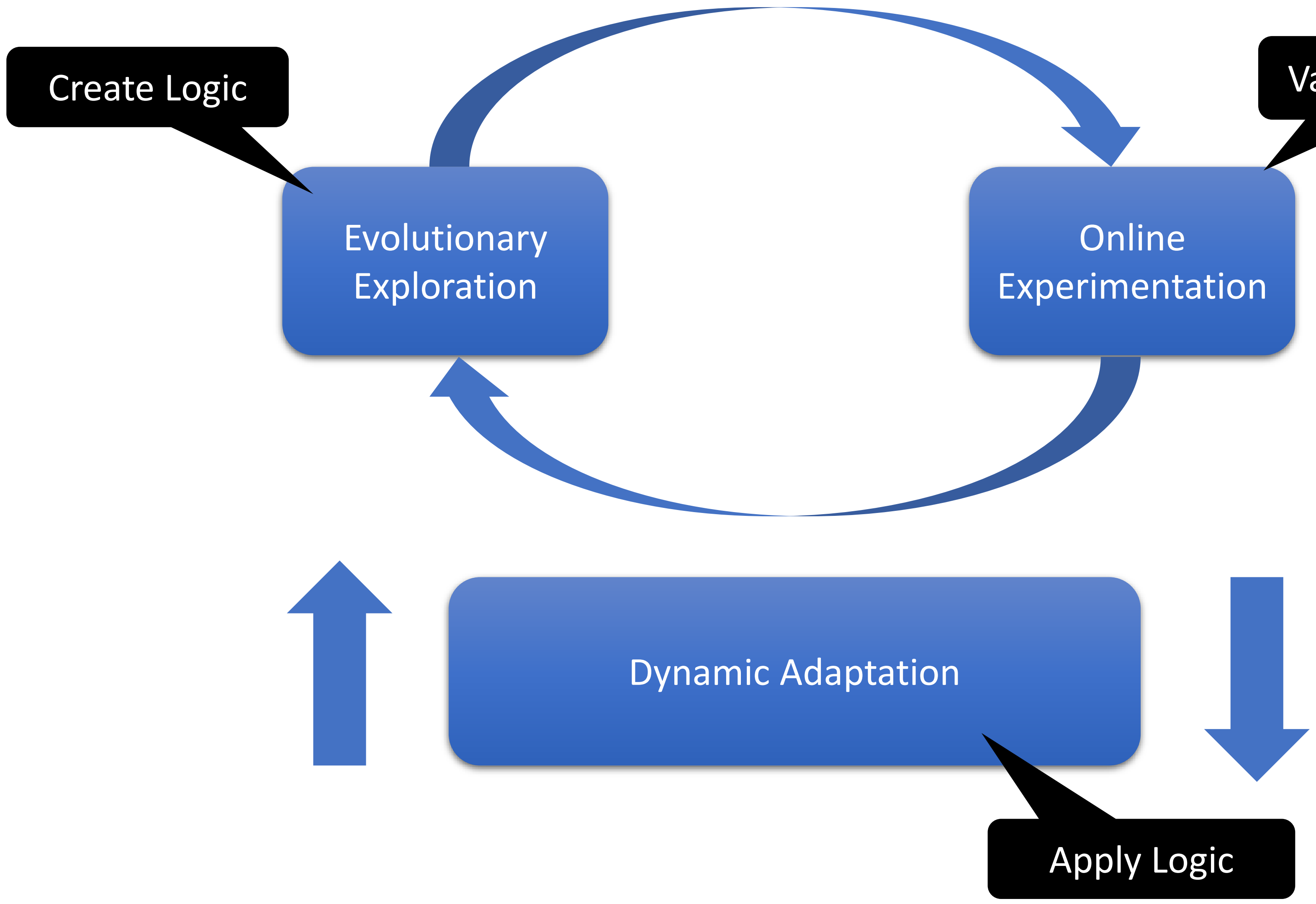
ITU-T Focus Group on Autonomous Networks

Technical Specification

Use cases for Autonomous Networks



Three Key Concepts

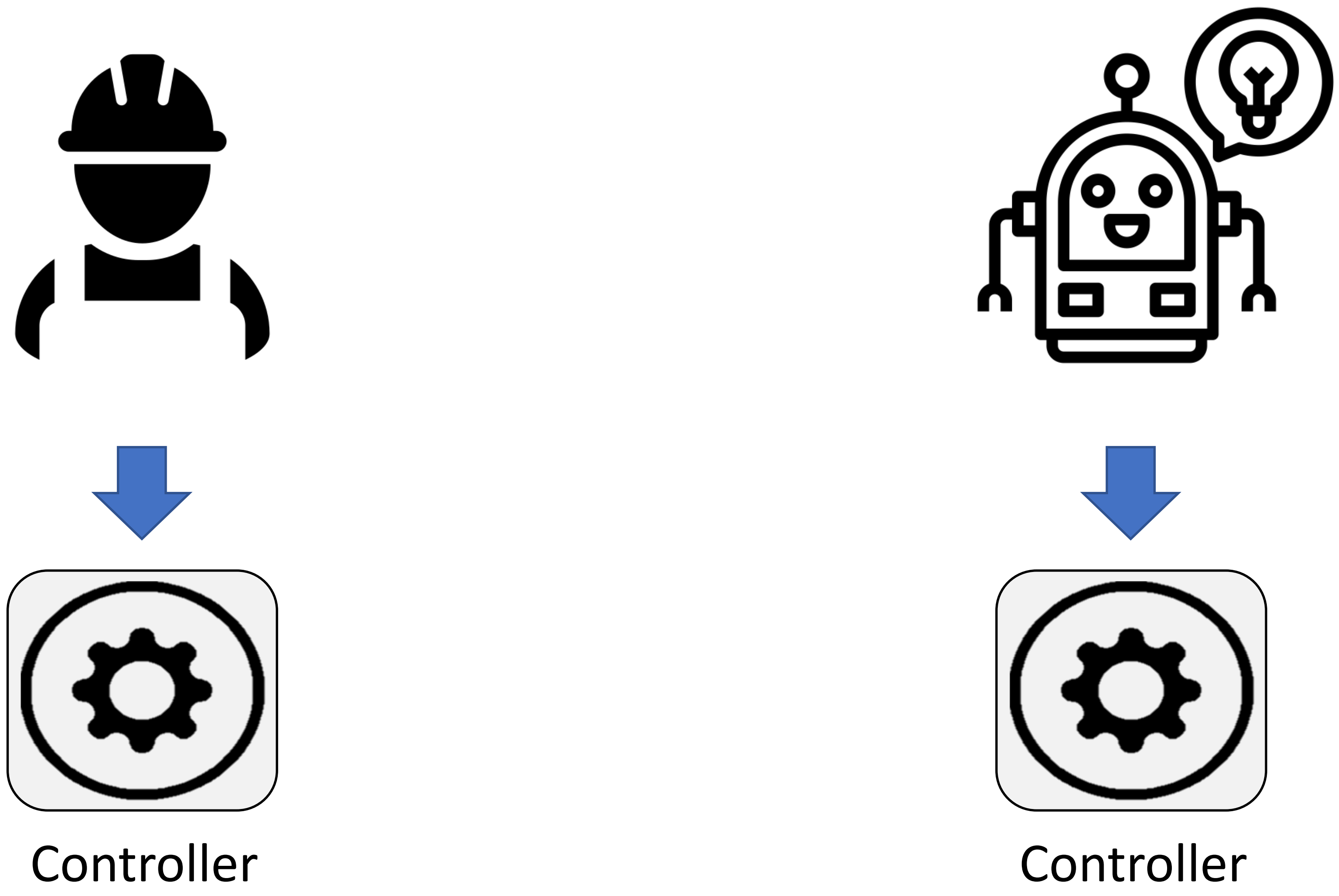


Requirements

- Requirements for Exploratory Evolution
- Requirements for Online Experimentation
- Requirements for Dynamic Adaptation
- Requirements for Knowledge
- Requirements for Autonomous Network Orchestration

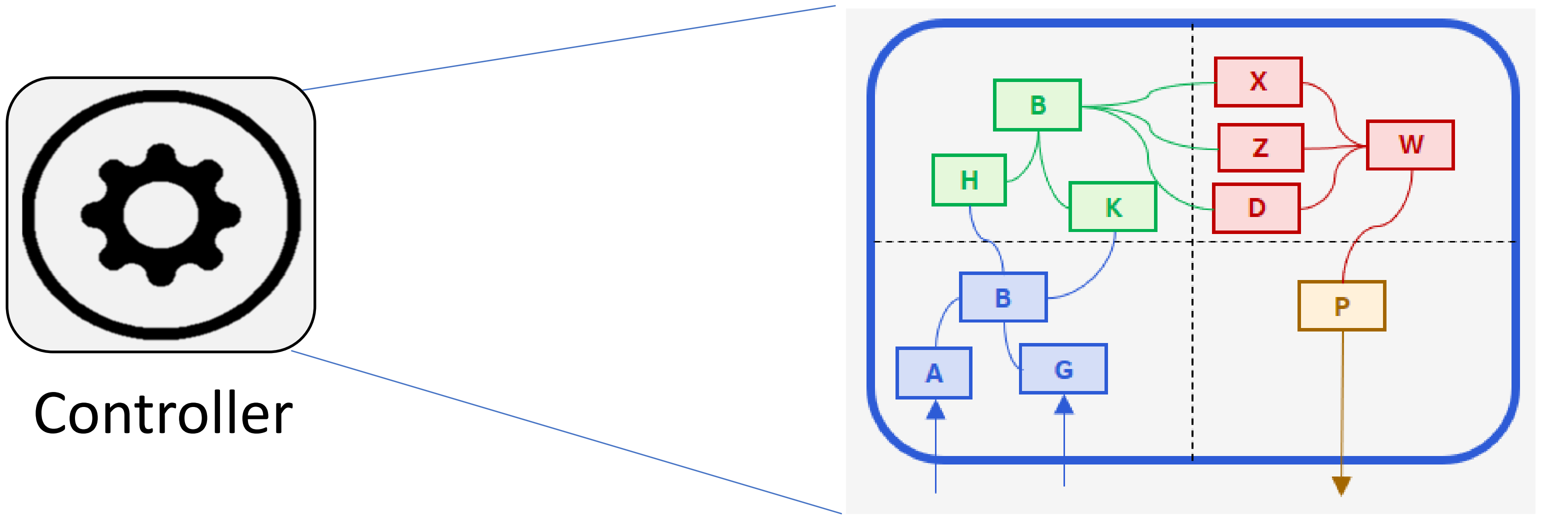
AN-arch-exp-req-003	The AN architecture is required to have the ability of executing experimentations, collating and validating the results of the experimentations, considering the metadata and constraints and corresponding controller descriptions. NOTE 1- Experimentations may have several phases, for example, simulation driven, testbed driven or canary test driven. The phases of an experimentation may be configurable and automated, for example, as per a workflow. NOTE 2- The specific success and failure criteria for the experiments are out of scope of this Recommendation. Acceptable formats for representing metadata and constraints related to potential success and failure as related to a use case are for further study.

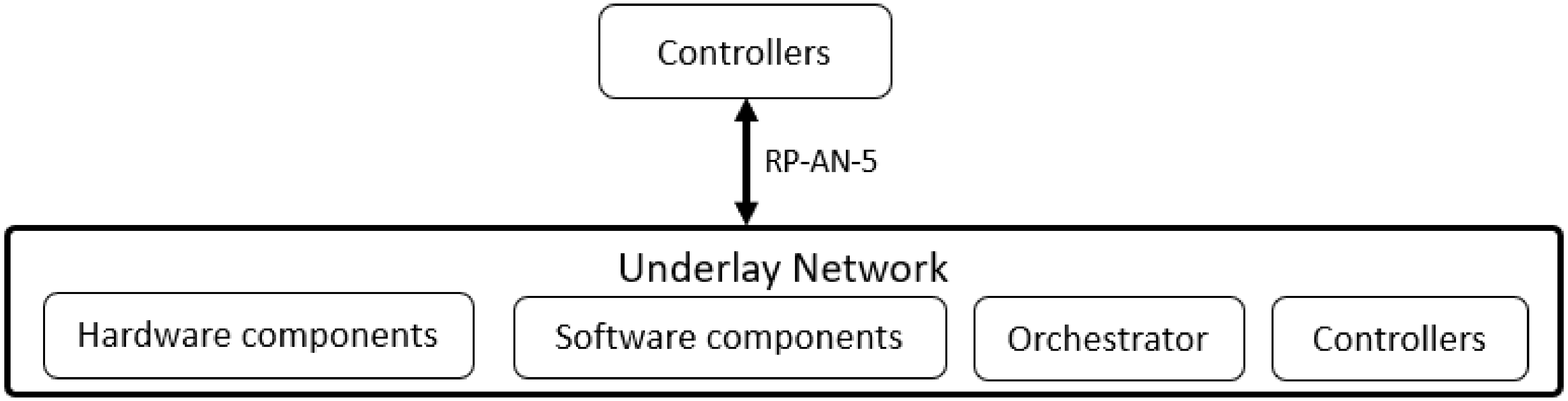
AN-arch-evo-req-008	The AN architecture is required to support the ability of discovering the characteristics of controllers which are relevant for enabling evolution. NOTE - Examples of characteristics of controllers which are relevant for evolution are: capabilities exposed by the controllers, and requirements to be satisfied for the controllers.


Controller
Controller

Definition 3.2.4: Controller

A workflow, open loop or closed loop of a system under control in an autonomous network, composed of modules, integrated in a specific sequence, using interfaces exposed by the modules, to solve a specific problem or satisfy a given requirement.





The interactions are:

- Controller interacting with hardware components.
- Controller interacting with software components.
- Controller interacting with an orchestrator or other software control mechanisms.
- Controller interacting with other controllers.

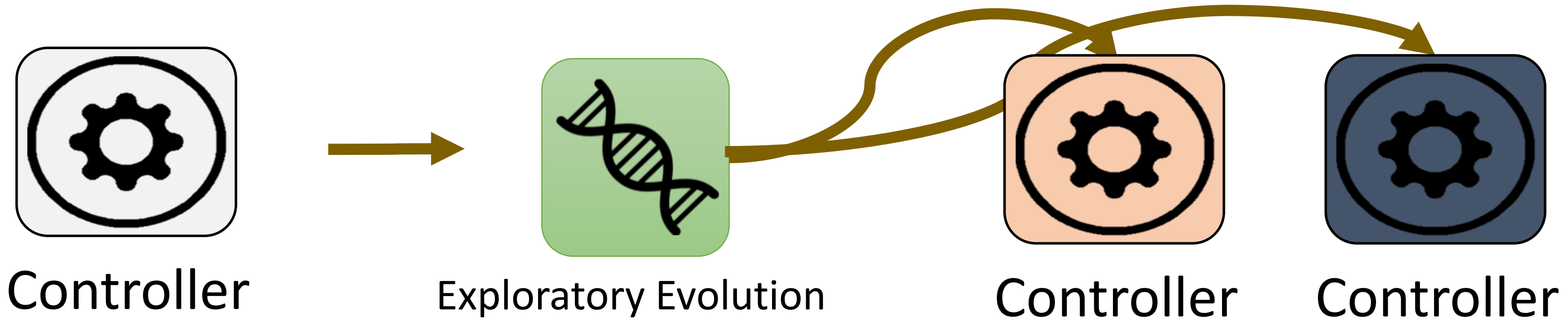
NOTE 4 - Building upon this simple representation, hierarchies of controllers may be formed.

Exploratory Evolution

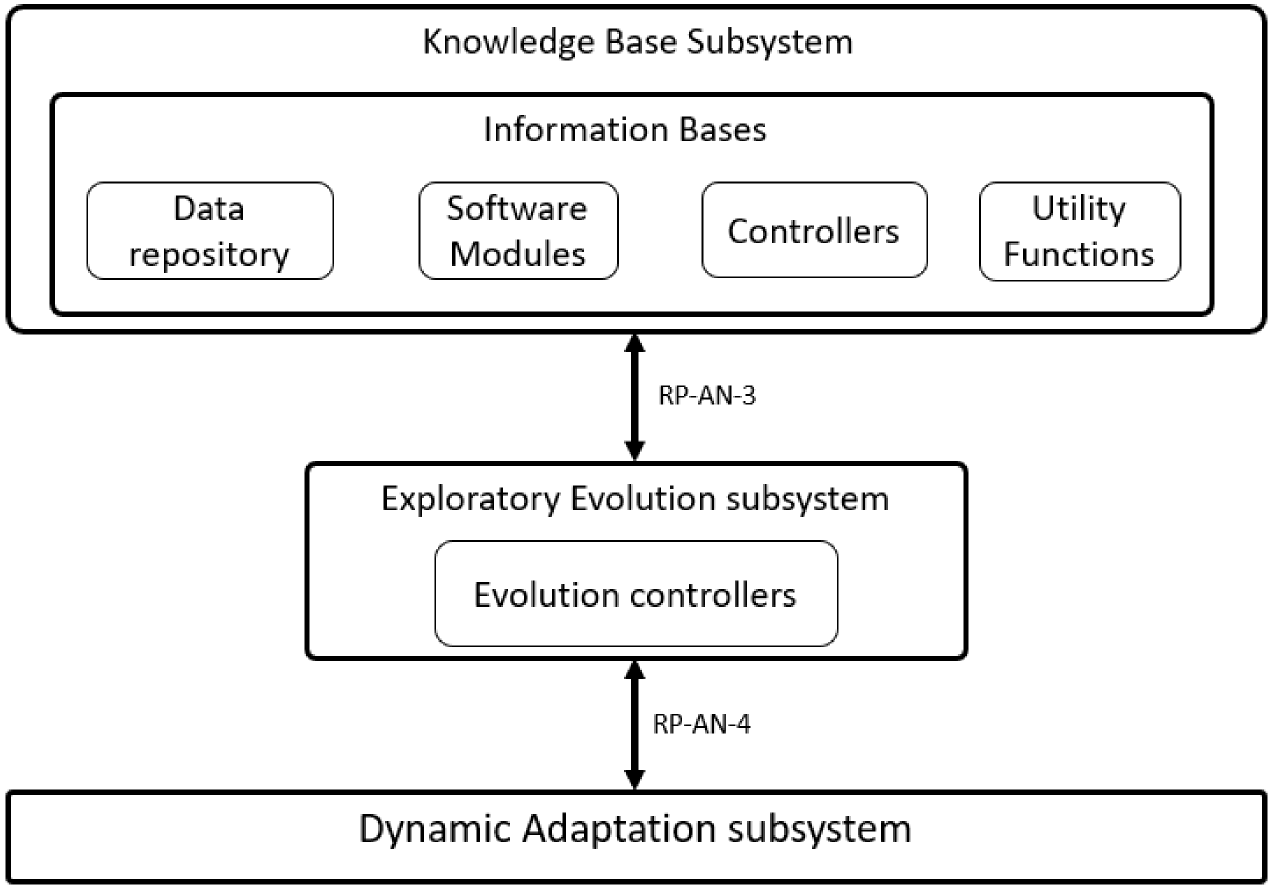
Exploratory evolution exploratory evolution introduces the mechanisms and processes of exploration and evolution to adapt controllers in response to changes in the underlay network.

NOTE 1 - An example of a process that creates a controller is the composition of controllers from modules or other closed loops. This may involve the selection of modules which are used for composition.

NOTE 2 - An example of a process that modifies an existing controller is the dynamic change in the controller's structure by adding new modules, deleting existing modules, replacing existing modules, or rearranging the structure of a controller's modules, in accordance with the real time changes in the system under control.



Exploratory Evolution



- **3.2.7 Evolution controller**: A controller responsible for the evolution of controllers by manipulating the module instance used within a controller, the structure or topology of connections between modules in a controller and/or the values chosen for the module(s) parameters

- Examples of processes to drive the modification of a controller are:
  - biologically inspired artificial evolution(e.g evolutionary computing or genetic programming [b-large-evolution, b-evolution])
  - Bayesian optimisation [b-bayesian-radio]
  - game theoretic approaches [b-game-theory].

Examples of Controller Evolution

1) A “RAN channel scheduling controller” is an example of a controller used to allocate radio resources to users in a multi-user environment. Exploratory evolution is applied to a RAN channel scheduling controller in response to the change of radio channel feedback from the UE. This may include selecting the most appropriate algorithm from a set of alternatives.

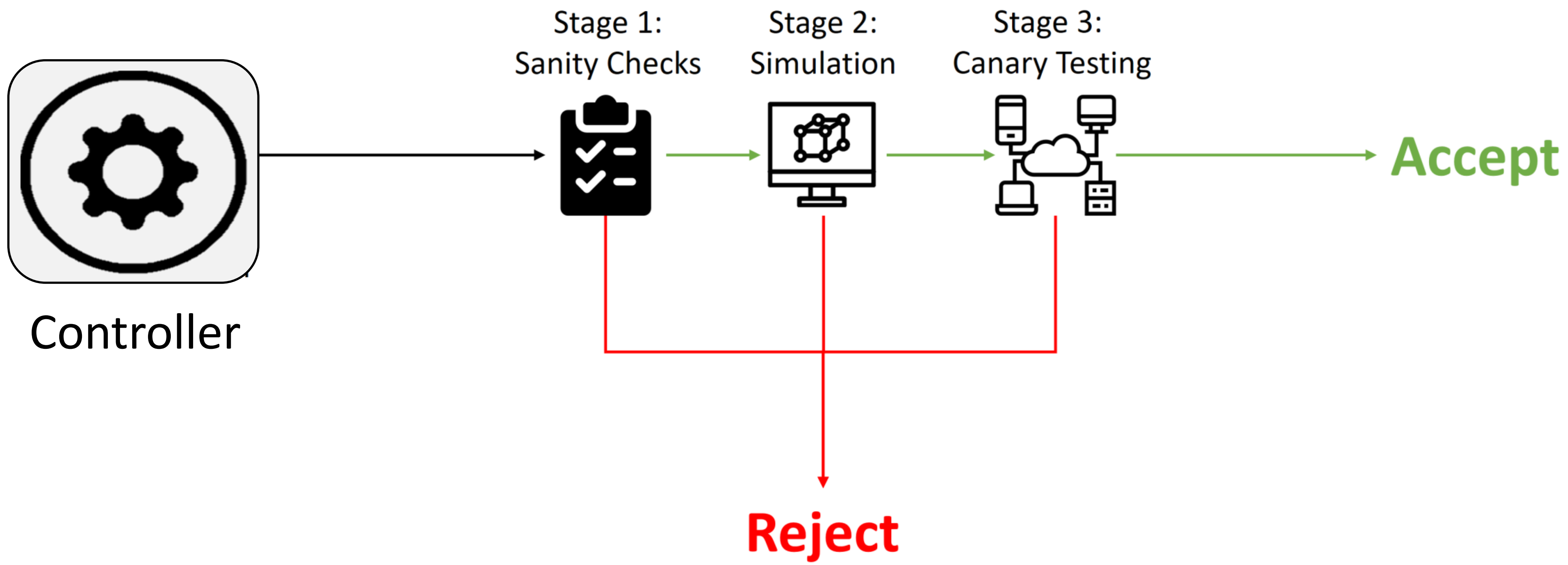
2) An “anomaly detection controller” is an example of a controller used to detect abnormal states in the operation of a network service, such as security attacks or peaks in resource usage for network function. In this context, the new approaches of data fusion algorithms [b-data-fusion] may be applied. Exploratory evolution is applied to “anomaly detection controller” by optionally using and configuring newly provided data fusion algorithms as the input of an “anomaly detection controller”,

3) A “time-to-live controller” is an example of a controller used to configure the time duration for which a certain content is cached in a CDN server. In a time-to-live controller in a caching system at the edge, optimisation of the timeout parameter(s) is an example of application of exploratory evolution.

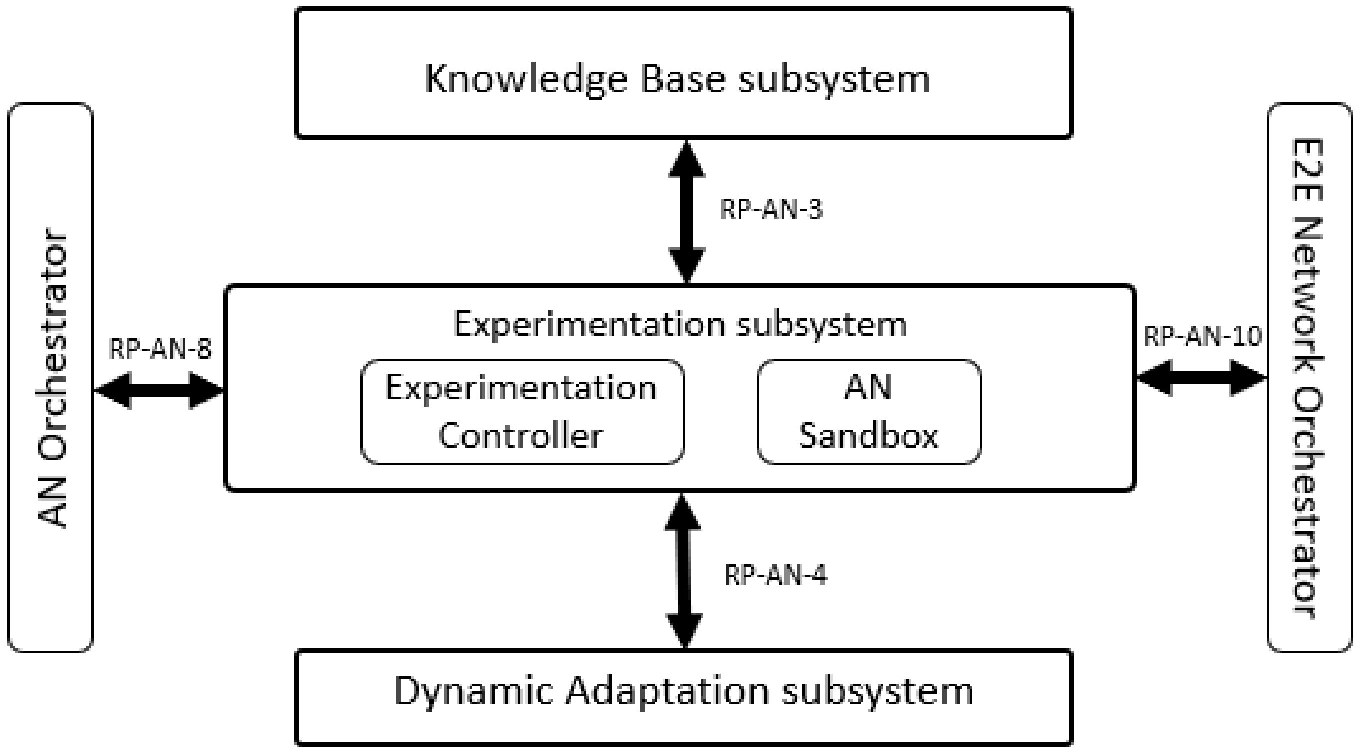
4) A “scaling controller” is an example of a controller used to increase or decrease the resource allocation for a network function. In this context, exploratory evolution may be applied by controlling the configuration of the scaling method of deployed controllers in a specific network domain.

Realtime Online Experimentation

3.2.8 Experimentation: The process of executing the generated potential scenarios and trials upon the controllers, within the parameters of the scenarios and trials and then collecting the results.



Realtime Online Experimentation



3.2.9 experimentation controller: A controller which generates potential scenarios of experimentations based on controller specifications and additional information as provided by the knowledge base, executes the scenarios in the AN Sandbox, collates and validates the results of the experimentation.

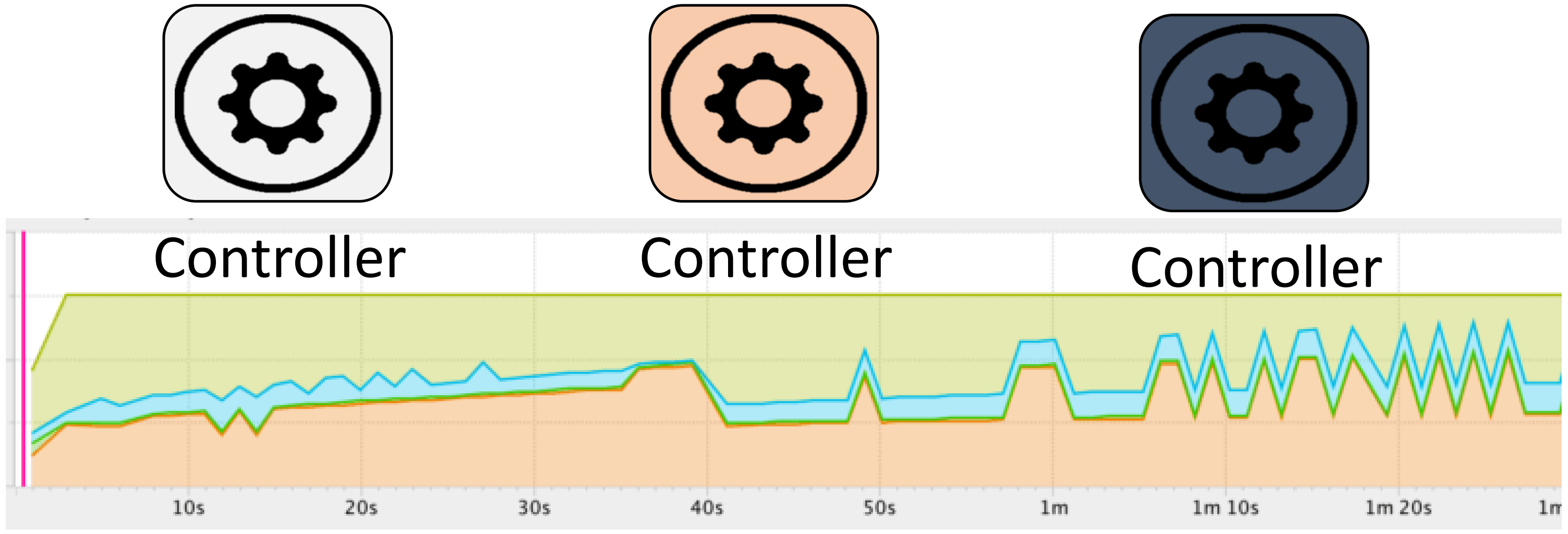
Examples of Controller Experimentation

Examples of experimentation in various application contexts are given below:

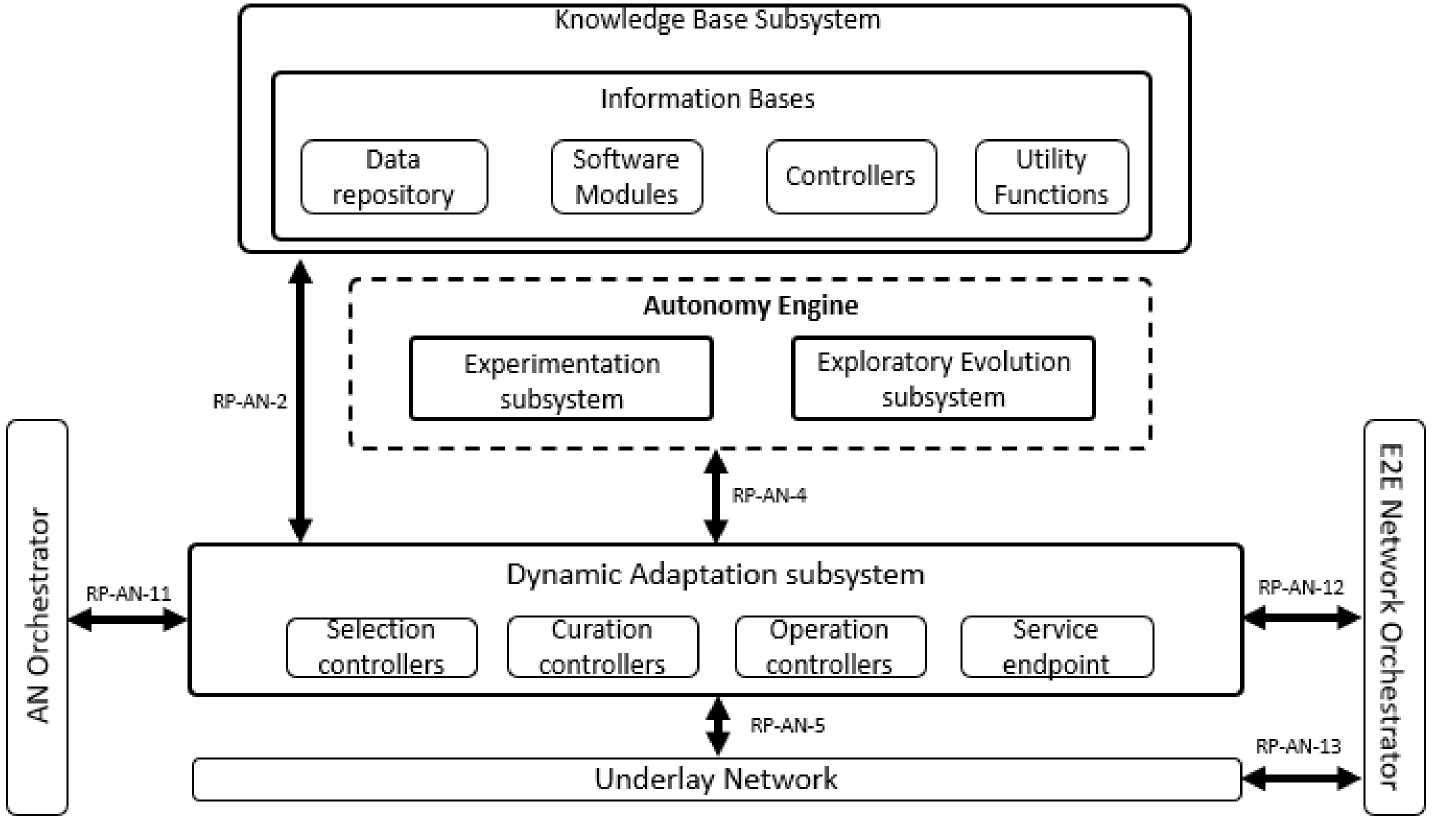
- The use of static “sanity checking” such as formal methods [ITU-T Y.3320] or model checking to ensure that provided management and orchestration solutions are well-formed against pre-defined rules
- The use of simulators or digital twins in offline validation of controllers. These simulators or digital twins can support the same interfaces as underlays.
- The use of digital twins [b-Digital-twin] in online validation of controllers before deployment
    - NOTE 5 - online validation involves use of timescales comparable to real underlays e.g. validation of controllers (xApps) [b-ORAN] using digital twins.
- Combinations of the above to achieve broader coverage of validation, from the offline validation to online validations during the operation of the underlay.

Dynamic Adaptation

- Dynamic adaptation is the **process of continuous integration of controllers to an underlay**, as the underlay undergoes changes at run-time. Integration of controllers may involve multiple domains of the underlay.



Dynamic Adaptation



3.2.1 adaptation controller: A controller responsible for selecting candidate controllers ready for integration and for executing their integration in the underlay network.

Adaptation controller has two parts:

- **Curation controller** (responsible for selection and maintenance of the controllers within the curated controller lists from the evolvable controllers) and

- **Selection Controller** (responsible for the selection of a services' operational controller from the curated controller lists).

Examples of Dynamic Adaptation

Examples of adaptation in various application contexts are given below:

- The need to use different traffic shaping algorithms for various geographical contexts, such as urban vs rural
- Business priorities may change over a period of time, e.g. prioritization of performance KPIs over energy efficiency or prioritisation of internal applications over third party applications. These changes in business priorities may necessitate the use of different virtual machine or container scheduling controllers.
- There could be a need to deploy new technology in order to improve or optimise operation, including adding new capabilities that previously did not exist. E.g. new AI/ML algorithms or new data fusion approaches to blend the increasing number of data sources.
- There could be a need to deploy new technology in order to address errors or faults. E.g. data acquisition or actuation software for new hardware devices or adaptation software to account for incompatibilities in deployed technology.

Architecture Components

Evolution Controller

Creates and modifies a controller in accordance with the system under control and the real-time changes therein.

Knowledge Base

Manages knowledge derived from and used in autonomous networks. It is updated and accessed by various components in the autonomous network.

AN Orchestrator

managing workflows and processes in the AN and steps in the lifecycle of controllers

Experimentation Controller

Validates controllers using inputs from a combination of underlay network, simulators and/or testbeds.

An Sandbox

environment in which controllers can be deployed, experimentally validated with the help of (domain specific) models of underlays

Adaptation Controller

Continuous integration of controllers to an underlay, as the underlay undergoes changes at run-time.

Architecture Components

FGAN-I-345-R2

Evolution Controller

Creates and modifies a controller in accordance with the system under control and the real-time changes therein.

Knowledge Base

Manages knowledge derived from and used in autonomous networks. It is updated and accessed by various components in the autonomous network.

AN Orchestrator

managing workflows and processes in the AN and steps in the lifecycle of controllers

Experimentation Controller

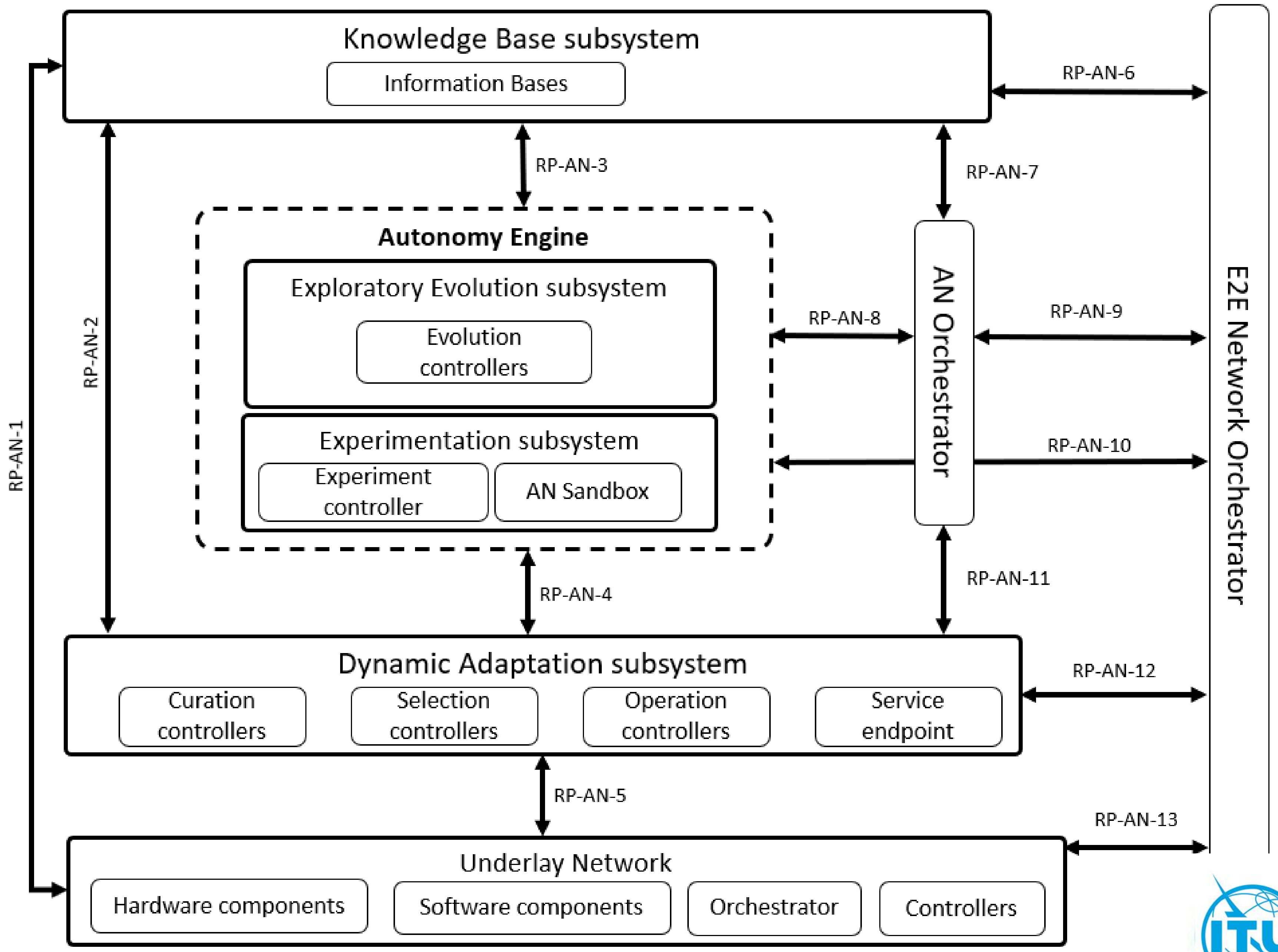
Validates controllers using inputs from a combination of underlay network, simulators and/or testbeds.

An Sandbox

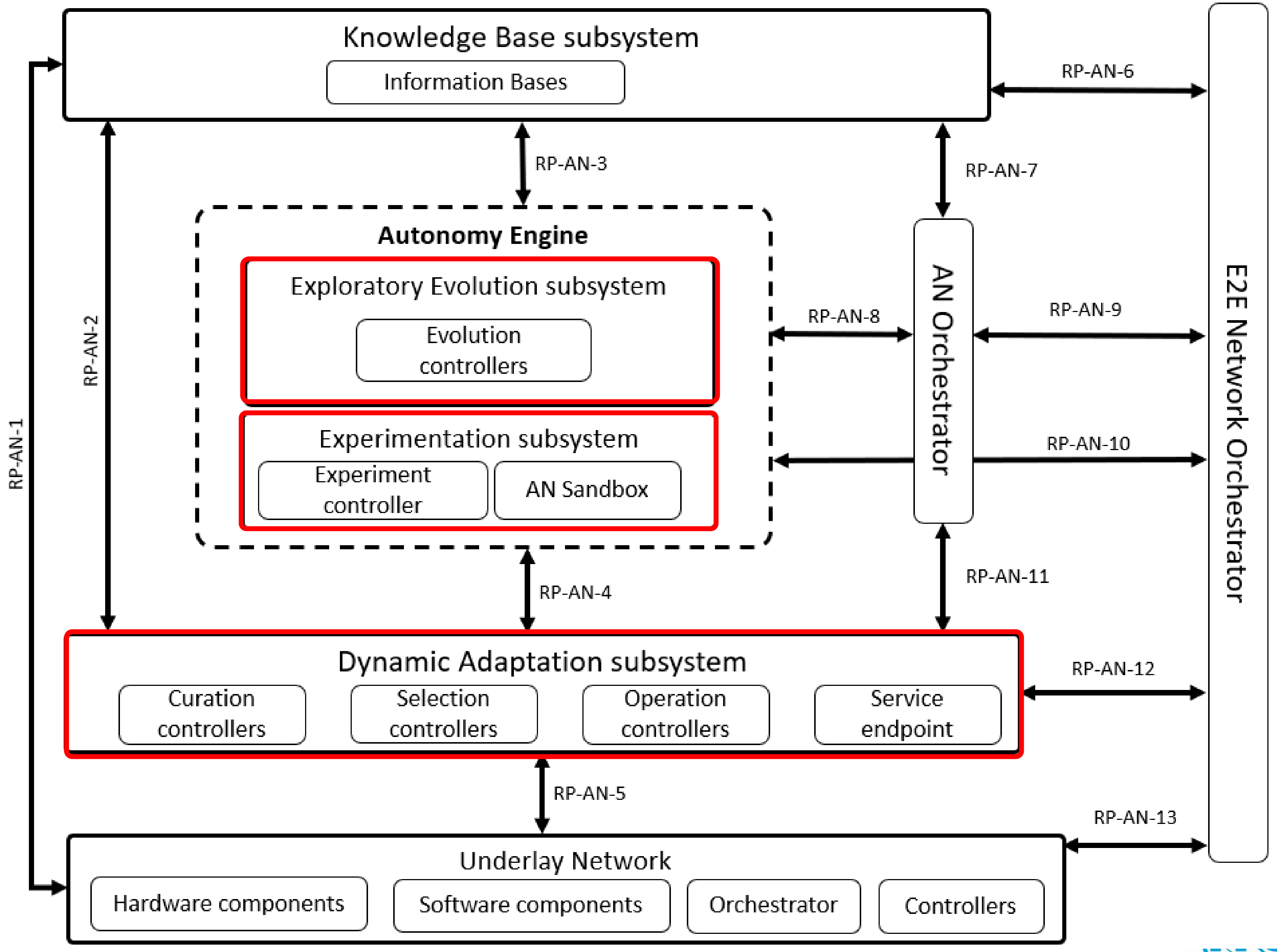
environment in which controllers can be deployed, experimentally validated with the help of (domain specific) models of underlays

Adaptation Controller

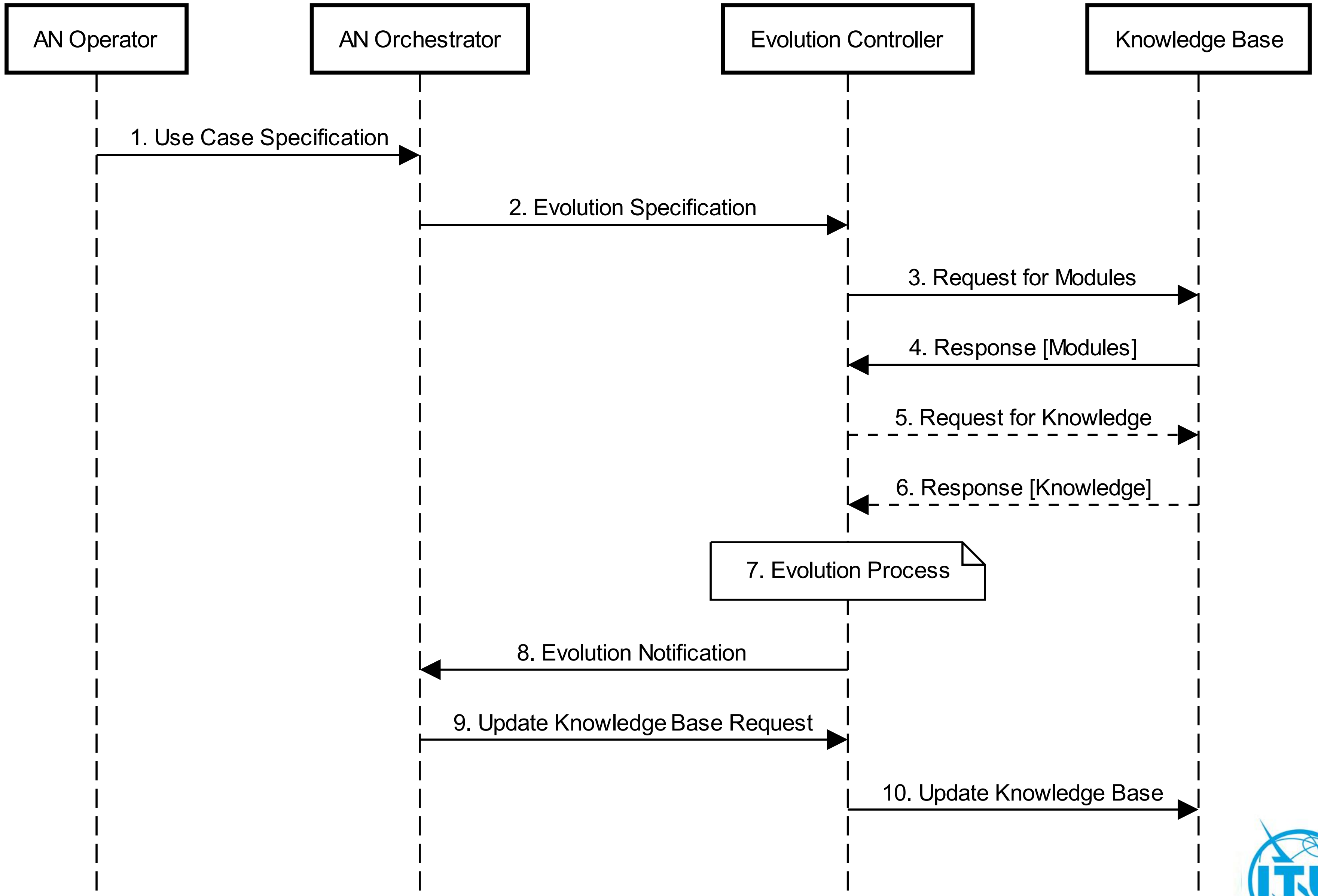
Continuous integration of controllers to an underlay, as the underlay undergoes changes at run-time.


Knowledge Base subsystem
Information Bases
RP-AN-1
RP-AN-2
RP-AN-3
RP-AN-4
RP-AN-5
RP-AN-6
RP-AN-7
RP-AN-8
RP-AN-9
RP-AN-10
RP-AN-11
RP-AN-12
RP-AN-13
Autonomy Engine
Exploratory Evolution subsystem
Evolution controllers
Experimentation subsystem
Experiment controller
AN Sandbox
AN Orchestrator
E2E Network Orchestrator
Dynamic Adaptation subsystem
Curation controllers
Selection controllers
Operation controllers
Service endpoint
Underlay Network
Hardware components
Software components
Orchestrator
Controllers

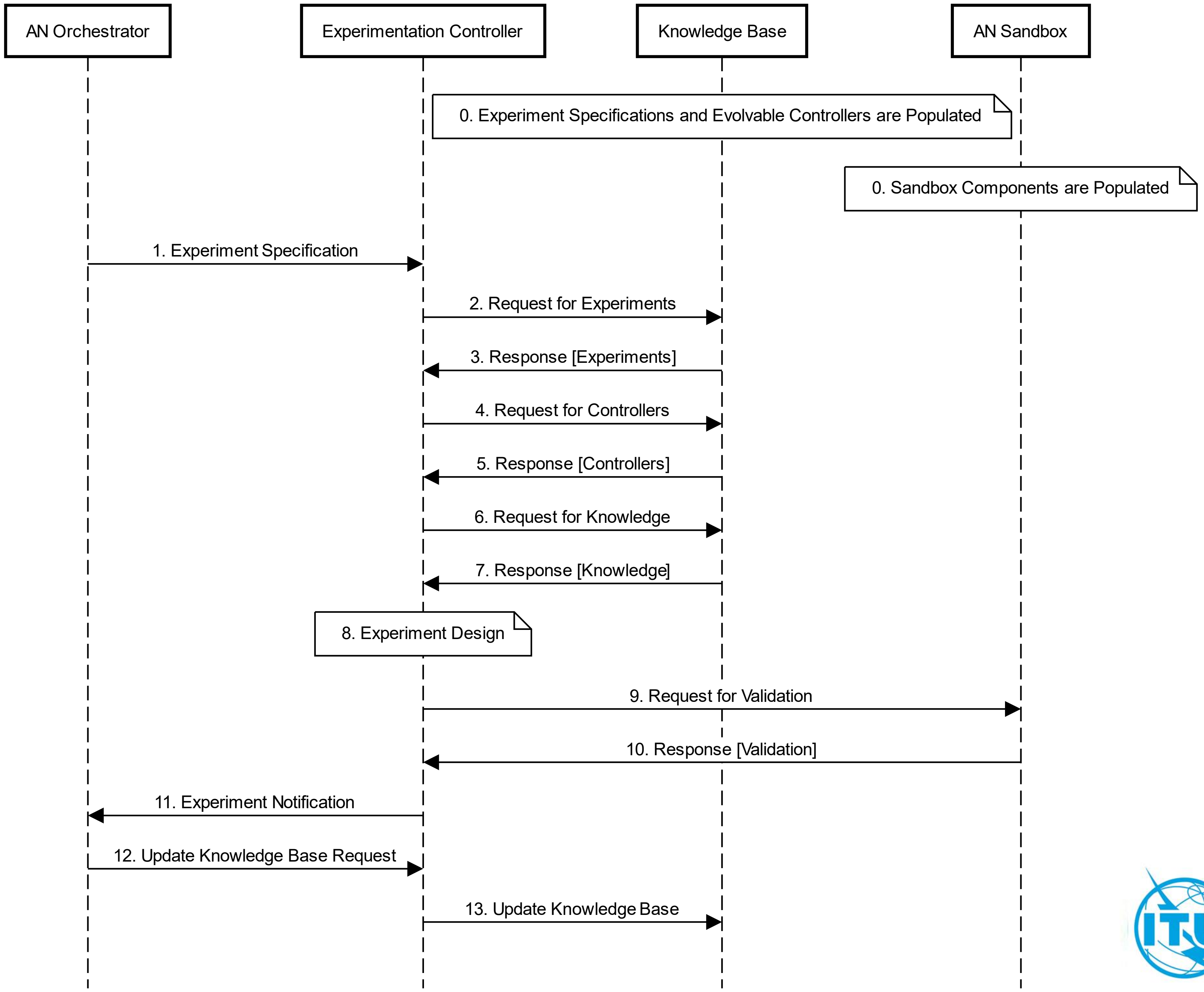
3 Key Concepts



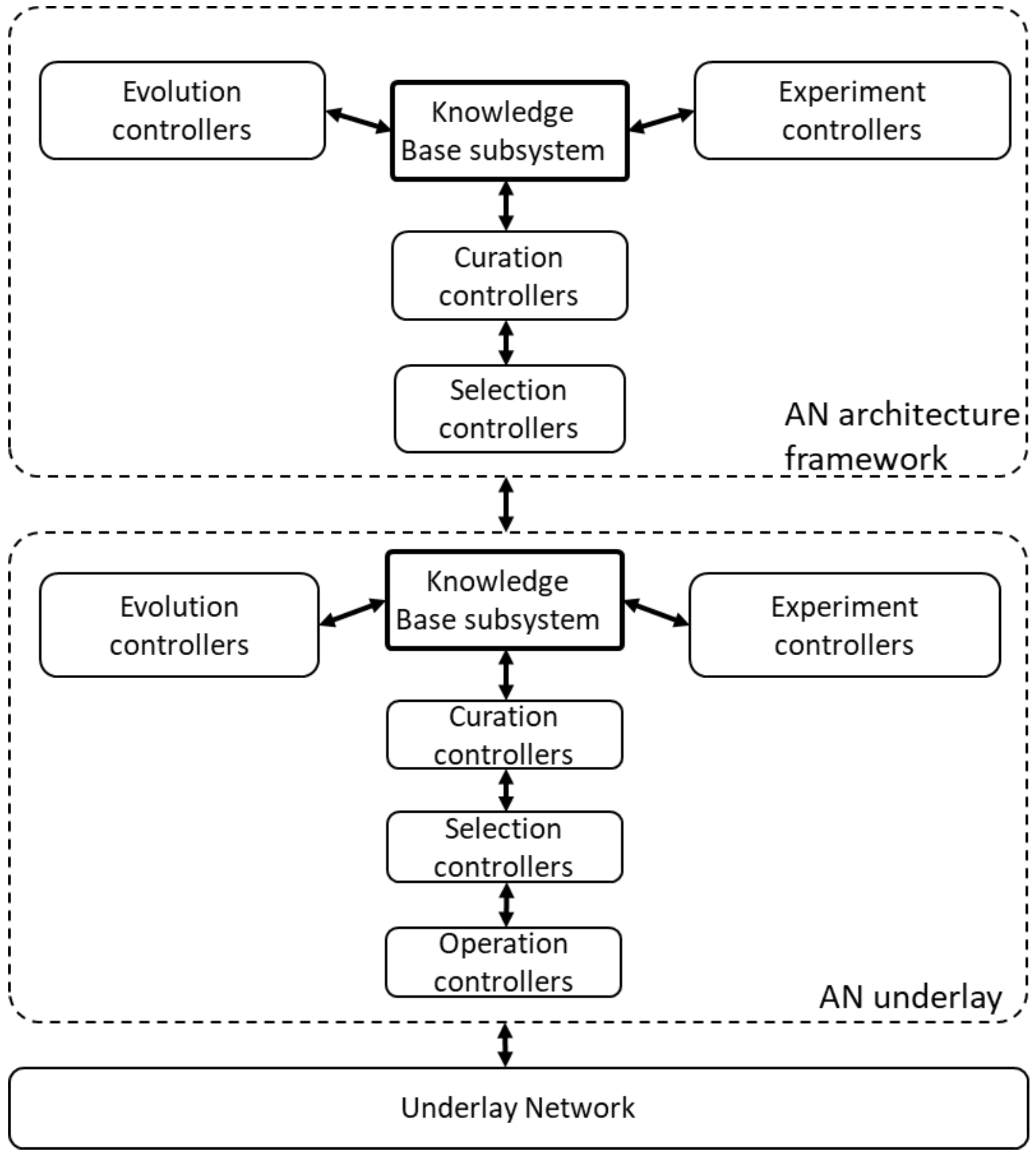
Exploratory Evolution



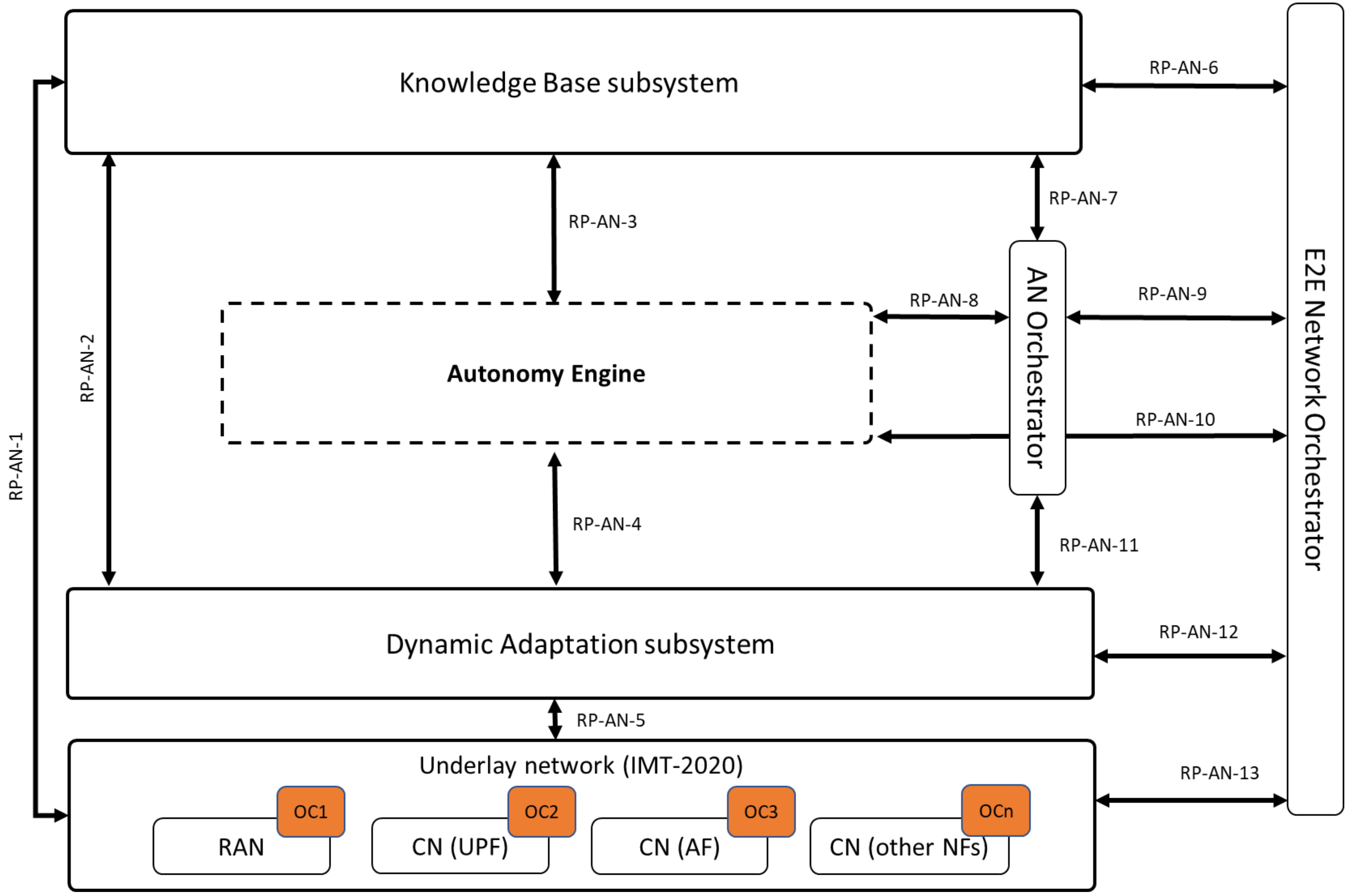
Online Experimentation


AN Orchestrator
Experimentation Controller
Knowledge Base
AN Sandbox
0. Experiment Specifications and Evolvable Controllers are Populated
0. Sandbox Components are Populated
1. Experiment Specification
2. Request for Experiments
3. Response [Experiments]
4. Request for Controllers
5. Response [Controllers]
6. Request for Knowledge
7. Response [Knowledge]
8. Experiment Design
9. Request for Validation
10. Response [Validation]
11. Experiment Notification
12. Update Knowledge Base Request
13. Update Knowledge Base

Self-Reflective Design



AN Architecture Mapping



AN Architecture Meets O-RAN

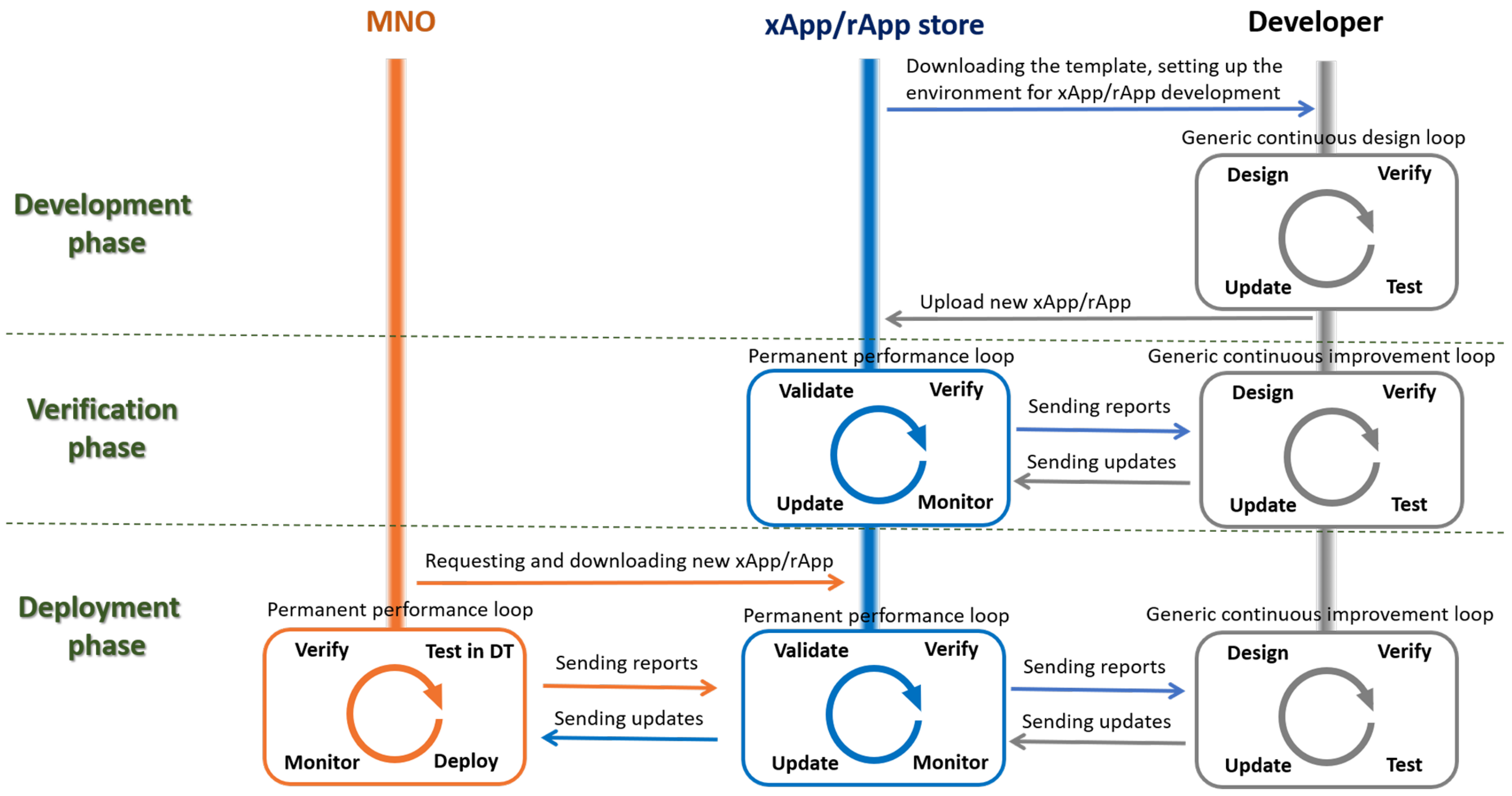


Fig. 4 – Key phases in automated xApp/rApp development, upload and deployment

A. Kliks, M. Dryjanski, V. OV Ram, L. Wong, and P. Harvey, “Towards Autonomous Open Radio Access Networks,” *ITU J. Futur. Evol. Technol.*, vol. 4, 2023.

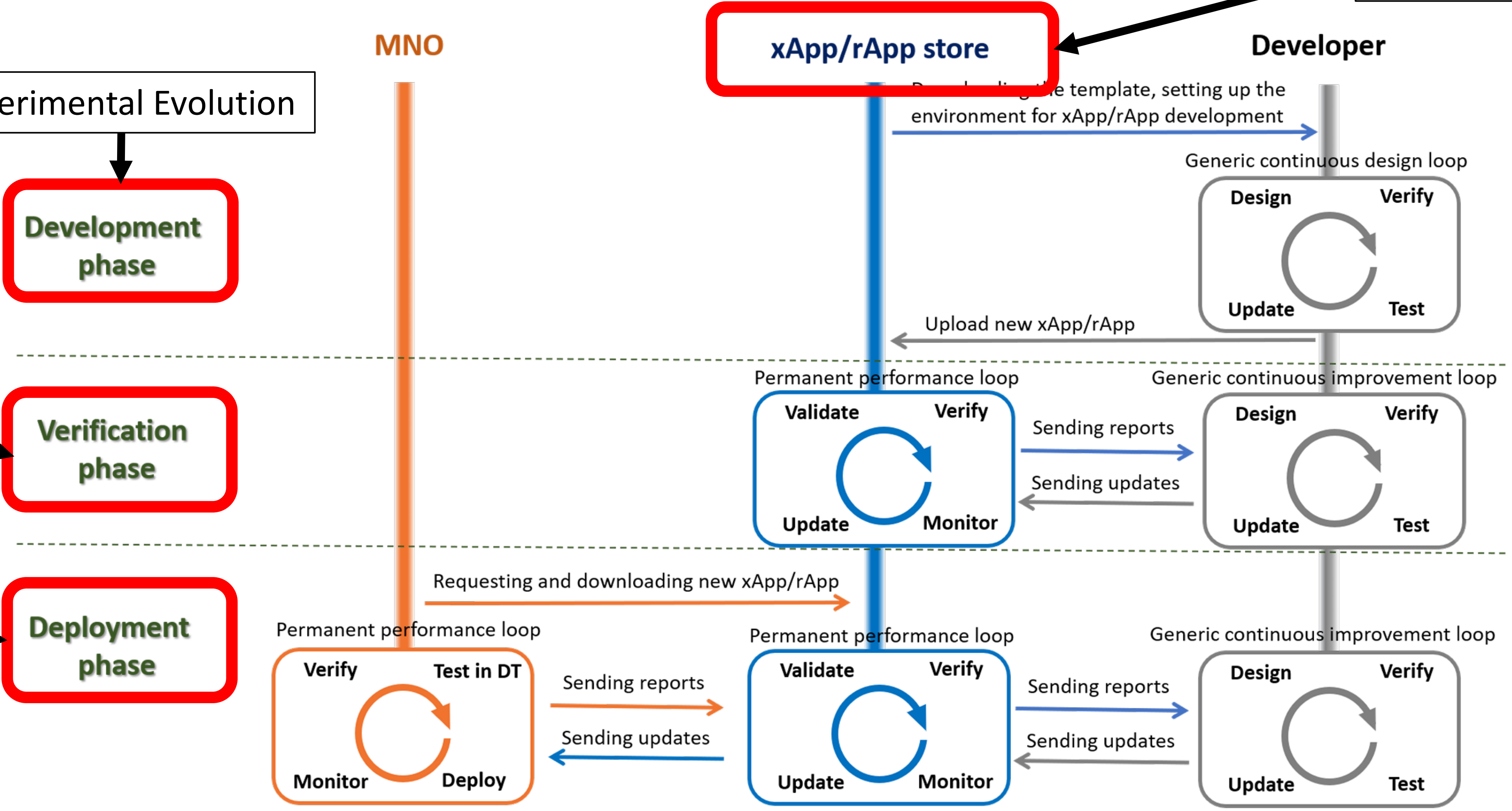
AN Architecture Meets O-RAN

Knowledge Base

Experimental Evolution

Online Experimentation

Dynamic Adaptation



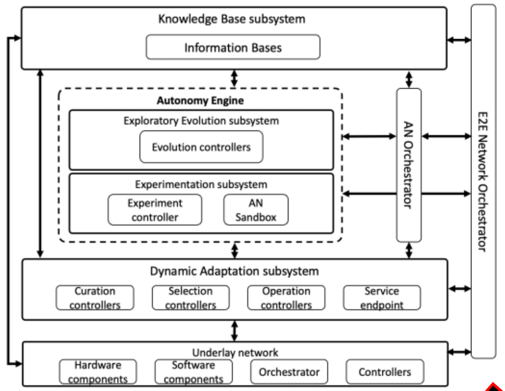
MNO
xApp/rApp store
Developer
Development phase
Downloading the template, setting up the environment for xApp/rApp development
Generic continuous design loop
Design
Verify
Update
Test
Upload new xApp/rApp
Verification phase
Permanent performance loop
Validate
Verify
Update
Monitor
Sending reports
Sending updates
Generic continuous improvement loop
Deployment phase
Requesting and downloading new xApp/rApp
Verify
Test in DT
Monitor
Deploy

Fig. 4 – Key phases in automated xApp/rApp development, upload and deployment

xApp == Operational Controller

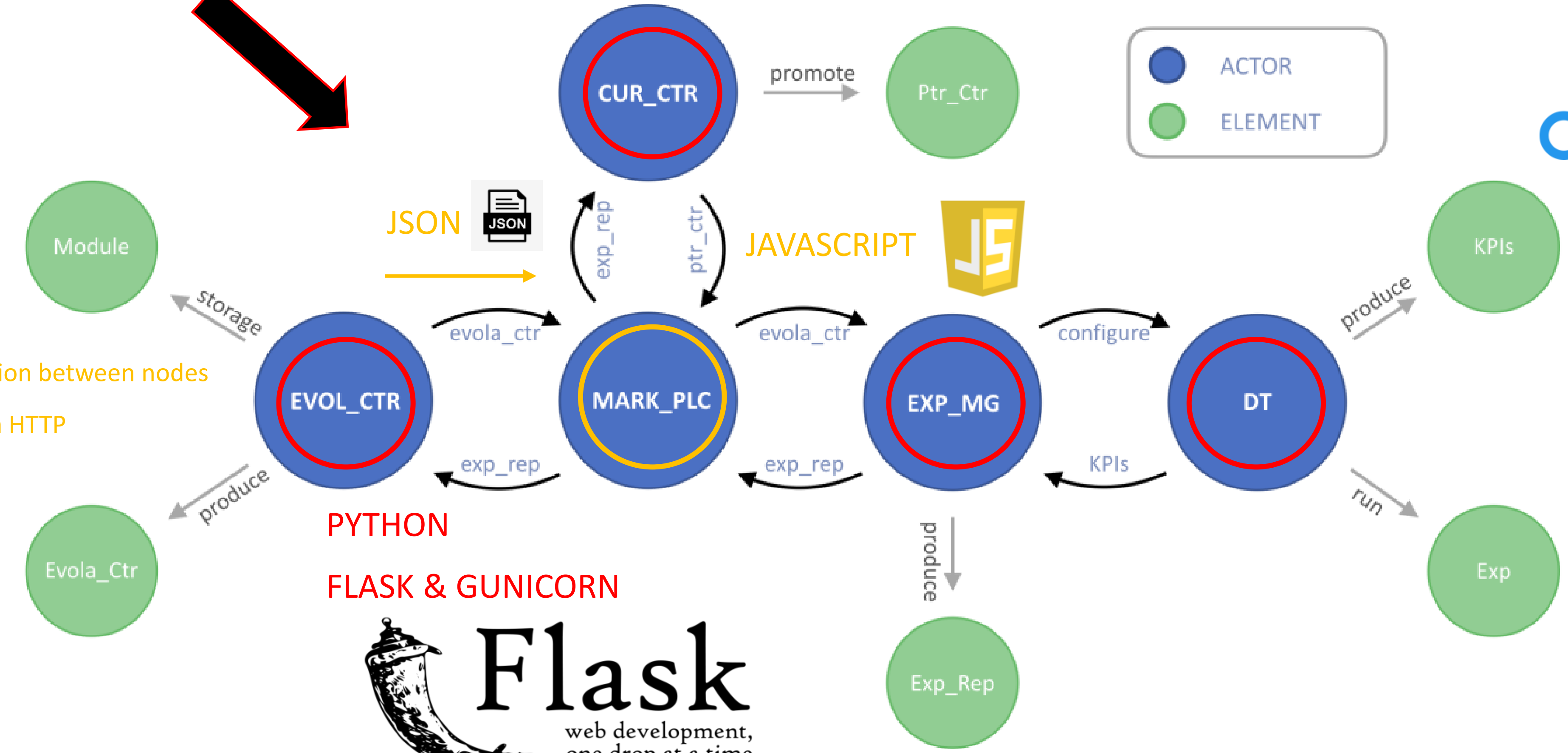
A. Kliks, M. Dryjanski, V. OV Ram, L. Wong, and P. Harvey, "Towards Autonomous Open Radio Access Networks," *ITU J. Futur. Evol. Technol.*, vol. 4, 2023.

AN Architecture Meets ***Reality***



JSON

JAVASCRIPT



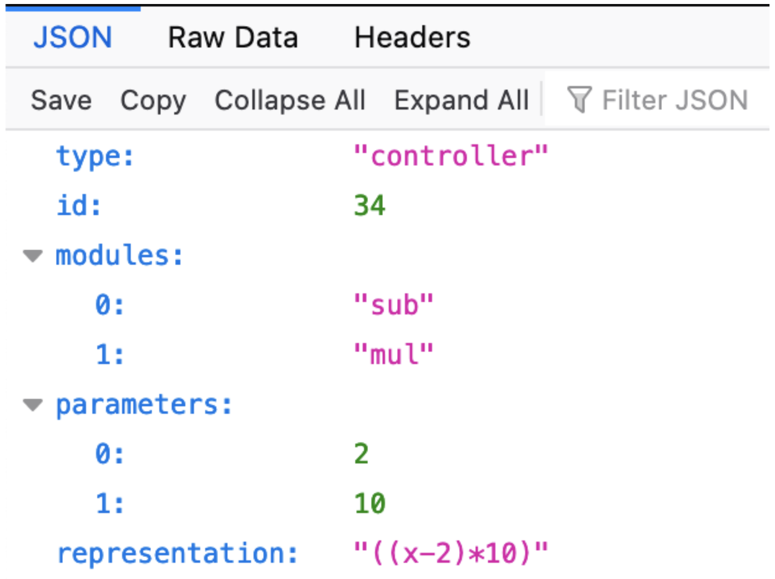
Communication between nodes

JSON files via HTTP

PYTHON

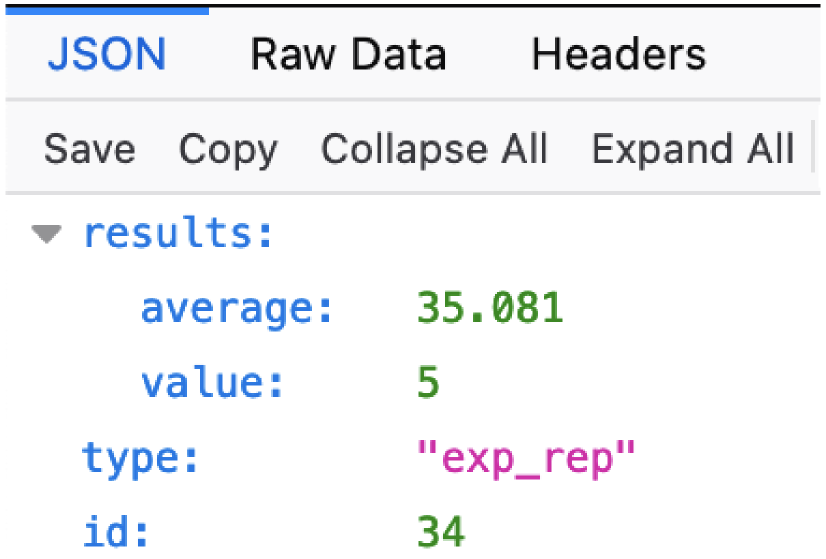
FLASK & GUNICORN

AN Architecture Meets ***Reality***



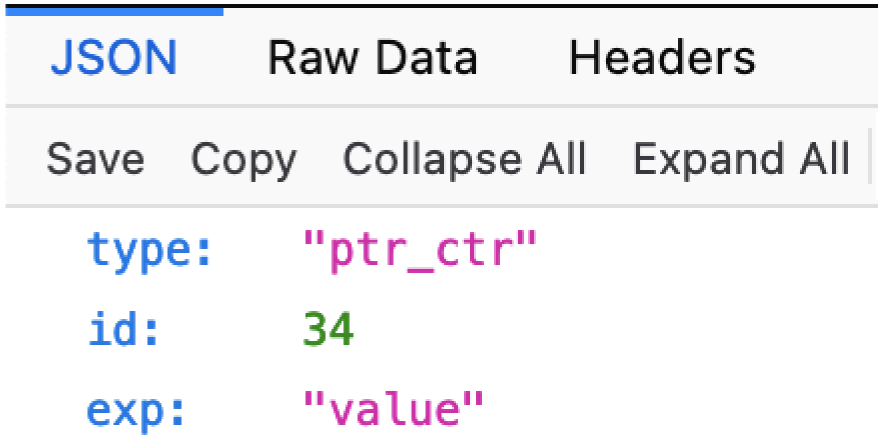
```
type:            "controller"
id:              34
▼ modules:
   0:            "sub"
   1:            "mul"
▼ parameters:
   0:            2
   1:            10
representation:  "((x-2)*10)"
```

Controller



```
▼ results:
    average:   35.081
    value:     5
  type:        "exp_rep"
  id:          34
```

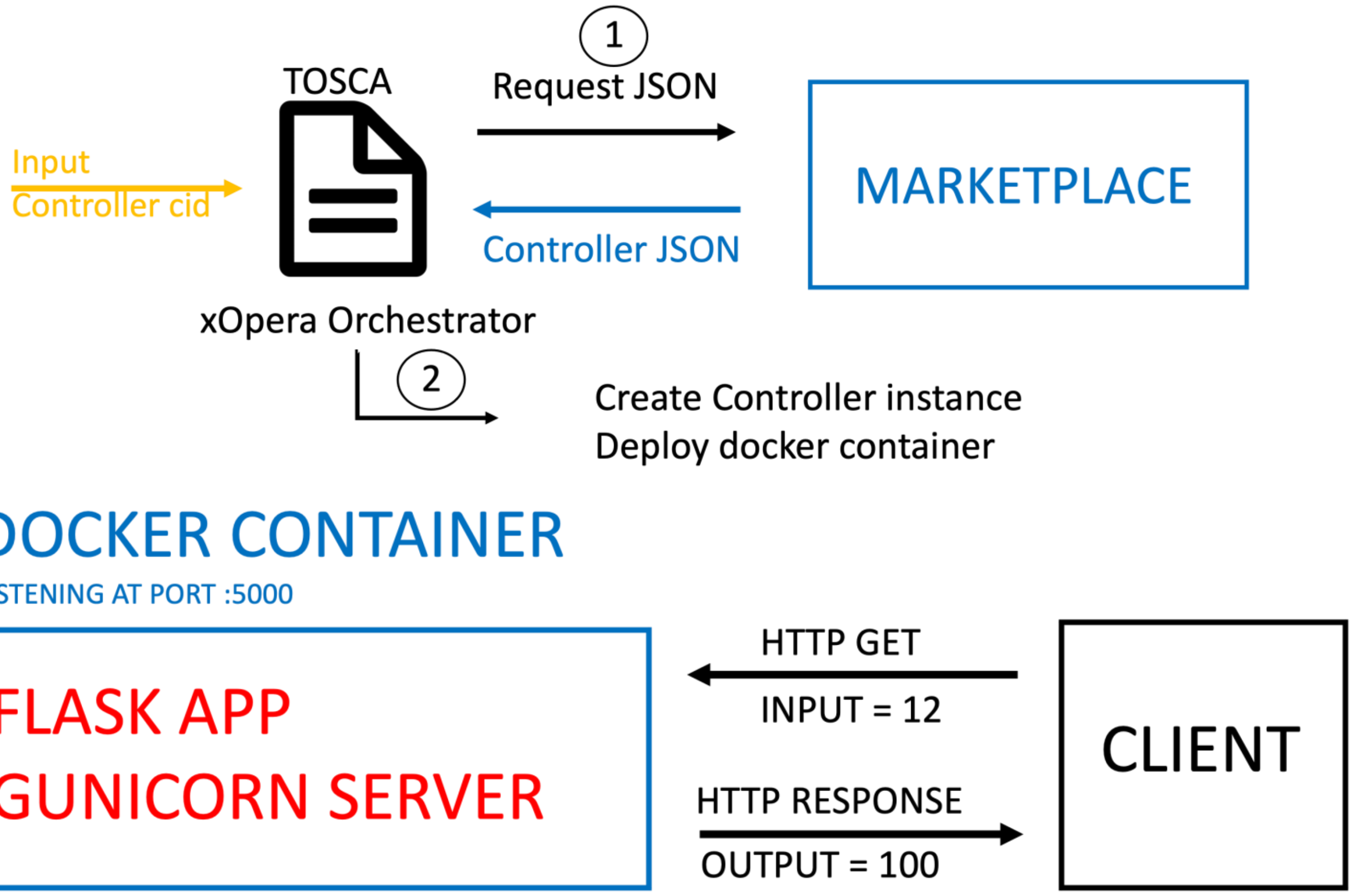
Experiment Report



```
type:   "ptr_ctr"
id:     34
exp:    "value"
```

Protected Controller

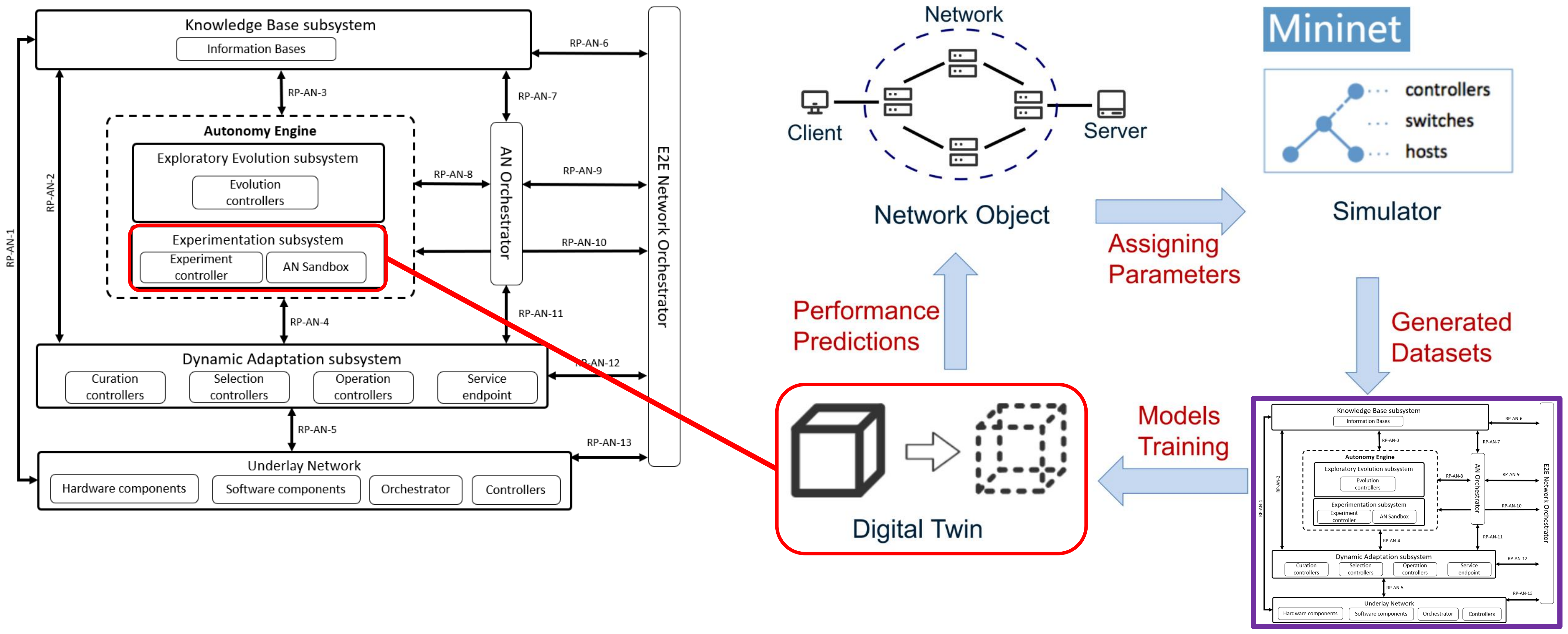
AN Architecture Meets *Reality*



Topology and Orchestration Specification for Cloud Applications

GITHUB: https://github.com/FGAN-Digital-Twins/BC-AN#evolution-blockchain-and-ipfs-an-autonomous-network-architecture-poc

AN Architecture Meets *Itself*

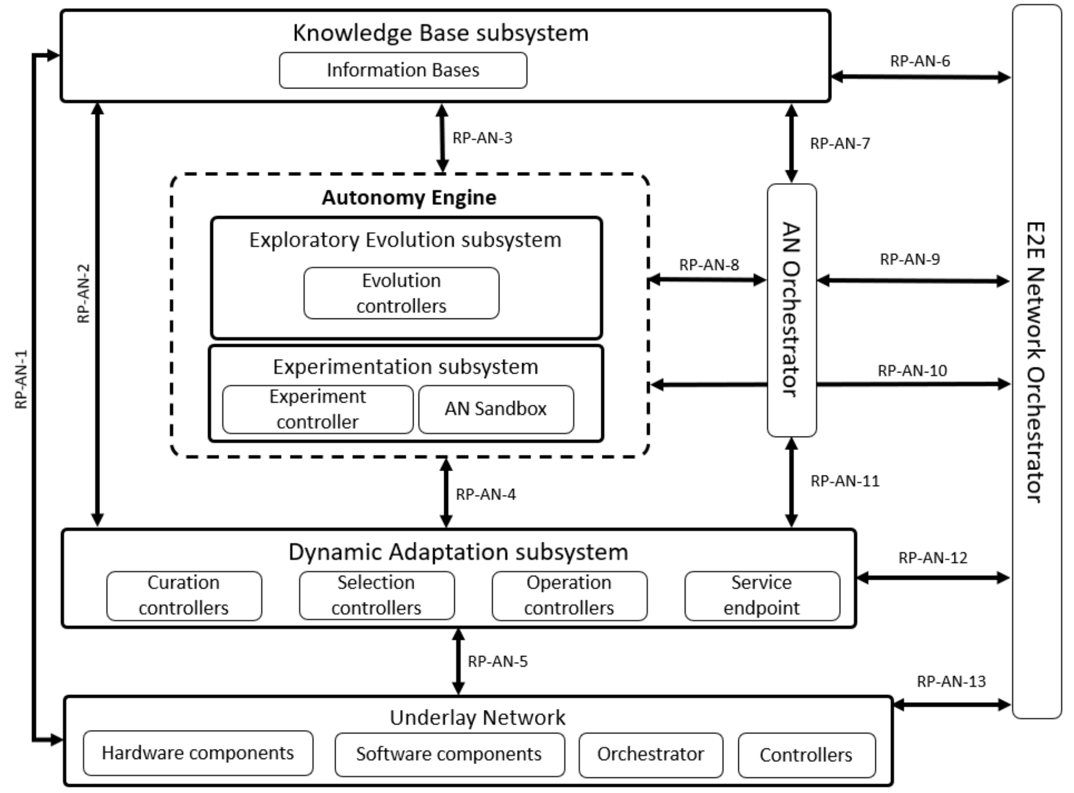


Questions – (FG)AN Architecture Highlights

- https://www.itu.int/en/ITU-T/focusgroups/an/Pages/default.aspx

#autonomousnetworks

@jhebus


Knowledge Base subsystem
Information Bases
RP-AN-6
RP-AN-3
RP-AN-7
RP-AN-2
RP-AN-1
Autonomy Engine
Exploratory Evolution subsystem
Evolution controllers
Experimentation subsystem
Experiment controller
AN Sandbox
RP-AN-8
AN Orchestrator
RP-AN-9
RP-AN-10
E2E Network Orchestrator
RP-AN-4
RP-AN-11
Dynamic Adaptation subsystem
Curation controllers
Selection controllers
Operation controllers
Service endpoint
RP-AN-12
RP-AN-5
RP-AN-13
Underlay Network
Hardware components
Software components
Orchestrator
Controllers