# ETSI GR ENI 017 V2.2.1 (2024-06)


# Experiential Networked Intelligence (ENI); Overview of Prominent Control Loop Architectures

***Disclaimer***

The present document has been produced and approved by the Experiential Networked Intelligence (ENI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
RGR/ENI-0017v221_Ctrllloop Arch

Keywords
cognitive, control

***ETSI***

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

# Contents

# Intellectual Property Rights

## Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

## Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

**DECT™**, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

# Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Experiential Networked Intelligence (ENI).

# Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

# Executive summary

The present document specifies a high-level functional abstraction of the ENI System Architecture in terms of Functional Blocks and External Reference Points. This includes describing how different classes of systems interact with ENI. Processes, models, and detailed information are beyond the scope of the present document.

# 1 Scope

The purpose of the present document is to provide further information on prominent control loop architectures that can be used in modular system design. This will be applied to the ENI reference system architecture (and any other applicable ETSI reports and standards). The present document will emphasize control loops that are adaptive and cognitive. In release 2, the present document will provide further precisions on clause 4.4.4. It will also make any other updates that are required.

# 2 References

## 2.1 Normative references

Normative references are not applicable in the present document.

## 2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GS ENI 005 (V3.1.1): "Experiential Networked Intelligence (ENI); System Architecture".

[i.2] Strassner, J., Agoulmine, N., Lehtihet, E.: "FOCALE - A Novel Autonomic Networking Architecture", ITSSA Journal 3(1), 64-79, 2007.

[i.3] Boyd, J. R.: "The Essence of Winning and Losing", June 1995.

[i.4] Strassner. J.: "Knowledge Representation, Processing, and Governance in the FOCALE Autonomic Architecture", book chapter, 2011, Elsevier.

[i.5] Strassner, J.: "Policy-Based Network Management", Morgan Kaufman, ISBN 978-1558608597, September 2003.

[i.6] MEF 78.1: "MEF Technical Specification: MEF Core Model", Strassner, J., editor, July 2020.

[i.7] IBM® Autonomic Computing White Paper: "An architectural blueprint for autonomic computing".

[i.8] Strassner, J., van der Meer, S., Won-Ki Hong, J.: "The design of an Autonomic Element for managing emerging networks and services", International Conference on Ultra Modern Telecommunications, 2009.

[i.9] Minsky, M.: "The Society of Mind", Simon and Schuster, New York, 1988.

[i.10] R. Mitchell, J. McKim: "Design by Contract, by Example", Addison-Wesley, 2001, ISBN 0201634600.

[i.11] S. van der Meer: "Architectural Artefacts for Autonomic Distributed Systems - Contract Language", in 6th IEEE Workshop on Engineering of Autonomic and Autonomous Systems (EASe), April 14-16, 2009.

[i.12] ETSI TS 103 195-2: "Autonomic network engineering for the self-managing Future Internet (AFI); Generic Autonomic Network Architecture; Part 2: An Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management".

[i.13] ETSI GR ENI 016: "Experiential Networked Intelligence (ENI); Functional Concepts for Modular System Operation".

[i.14] Clark, D.D., Partridge, C., Ramming, J.C., and Wroclawski, J.T.: "A Knowledge Plane for the Internet", August 2003.

# 3 Definition of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the terms given in ETSI GS ENI 005 [i.1] and following apply:

**abstraction:** hiding of unnecessary details to focus on data and information that is relevant for defining a particular concept or process

**administrative domain:** domain that employs a set of common administrative processes to manage the behaviour of its constituent Entities

NOTE: This is based on the definition in [i.6].

**agent:** computational process that implements the autonomous, communicating functionality of an application

**architecture:** set of rules and methods that describe the functionality, organization, and implementation of a system

**cognition:** process of understanding data and information and producing new data, information, and knowledge

**cognition model:** computer model of how cognitive processes, such as comprehension, action, and prediction, are performed and influence decisions

**context:** collection of measured and inferred knowledge that describe the environment in which an entity exists or has existed

**data model:** representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and/or protocol

NOTE: This definition is taken from [i.6].

**decision making:** set of processes that result in the selection of a set of actions to take from among several alternative possible actions

**domain:** collection of Entities that share a common purpose

NOTE: Each constituent Entity in a Domain is both uniquely addressable and uniquely identifiable within that Domain. This is based on the definition of an MCMDomain in [i.6].

**entity:** object in the environment being managed that has a set of unique characteristics and behaviour

NOTE: Objects are represented by classes in an information model.

**formal:** study of (typically linguistic) meaning of an object by constructing formal mathematical models of that object and its attributes and relationships

**information model:** representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol

NOTE: This definition is taken from [i.6].

**inferred knowledge:** knowledge that was created based on reasoning, using evidence provided

**knowledge:** 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.

**learning:** process that acquires new knowledge and/or updates existing knowledge to optimize a function using sample observations

**logic:** formal or informal language that evaluates a conclusion based on a set of premises

**management domain:** domain that uses a set of common Policies to govern its constituent Entities

NOTE: A Management Domain refines the notion of a Domain by adding three important behavioural features:

1) it defines a set of administrators that govern the set of Entities that it contains;
2) it defines a set of applications that are responsible for different governance operations, such as monitoring, configuration, and so forth;
3) it defines a common set of management mechanisms, such as policy rules, that are used to govern the behaviour of MCMManagedEntities contained in the MCMManagementDomain.

This is based on the definition of an MCMDomain in [i.6].

**model:** representation of the entities of a system, including their relationships and dependencies, using an established set of rules and concepts

**Model-Driven Engineering (MDE):** approach in which models are central to all phases of the development and implementation processes

**ontology (for ENI):** language, consisting of a vocabulary and a set of primitives, that enable the semantic characteristics of a domain to be modelled

**policy:** set of rules that is used to manage and control the changing and/or maintaining of the state of one or more managed objects

**semantics:** study of the meaning of something (e.g. a sentence or a relationship in a model)

**situation:** set of circumstances and conditions at a given time that may influence decision-making:

**situation awareness:** perception of data and behaviour that pertain to the relevant circumstances and/or conditions of a system or process, the comprehension of the meaning and significance of these data and behaviours, and how processes, actions, and new situations inferred from these data and processes are likely to evolve in the near future

## 3.2 Symbols

Void.

## 3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

| | |
|---|---|
| AFI | Autonomic Future Internet |
| AI | Artificial Intelligence |
| AMC | Autonomic Management and Control |
| API | Application Programming Interface |
| COM | Control, Orchestration and Management |
| COMPA | Control, Orchestration, Management, Policy and Analytics |
| DE | Decision Element |
| DEN | Directory Enabled Networks |
| DENON | Directory Enabled Networks ONtology |
| EMS | Element Management System |

| | |
|---|---|
| ESB | Enterprise Service Bus |
| FB | Functional Block |
| FOCALE | Foundation - Observe - Compare - Act - Learn - rEason |
| FSM | Finite State Machine |
| GANA | Generic Autonomic Networking Architecture |
| IBM® | International Business Machines |
| KP | Knowledge Plane |
| MAPE | Model-Analyse-Plan-Execute |
| MAPE-K | Model-Analyse-Plan-Execute-Knowledge |
| MBT | Model-Based Translation |
| MBTS | Model-Based Translation Services |
| MDE | Model-Driven Engineering |
| ME | Managed Element |
| NE | Network Element |
| NMS | Network Management System |
| ONIX | Overlay Network for Information eXchange |
| OODA | Observe-Orient-Decide-Act |
| OSS | Operational Support System |
| QoE | Quality of Experience |
| QoS | Quality of Service |
| XML | eXtensible Markup Language |

# 4 Prominent Control Loop Architectures

## 4.1 Introduction

Most control loop architectures for adaptive and cognitive systems use both feedback (and feedforward) mechanisms. These control loop signals play a critical role in not just stabilizing the system, but more importantly, providing mechanisms for the system to learn experientially.

EXAMPLE: A simple feedback loop consists of taking past interactions with the environment and combining them with current information to guide current and future interactions.

## 4.2 Definition

A control loop is a mechanism that senses the performance of an object or process being controlled to achieve desired behaviour. ENI is concerned with different types of closed control loops, where the controlling action is dependent on feedback from the object or process being controlled. In other words, closed loops use feedback to monitor and adjust the behaviour of a system to achieve one or more goals.

## 4.3 Types of Control Loops

### 4.3.1 Open

An open control loop is a control loop whose controlling action is independent of the output of the object or process being controlled. This type of control loop does not link the control action to the object or process being controlled (it simply continues to apply the control action). This type of control loop will likely not be used in the ENI system.

### 4.3.2 Closed

A closed control loop is a control loop whose controlling action is dependent on feedback from the object or process being controlled. This type of control loop measures the difference between the actual and desired values of a set of variables to adjust a set of parameters to change the behaviour of the system to bring the actual value closer to that of the desired value.

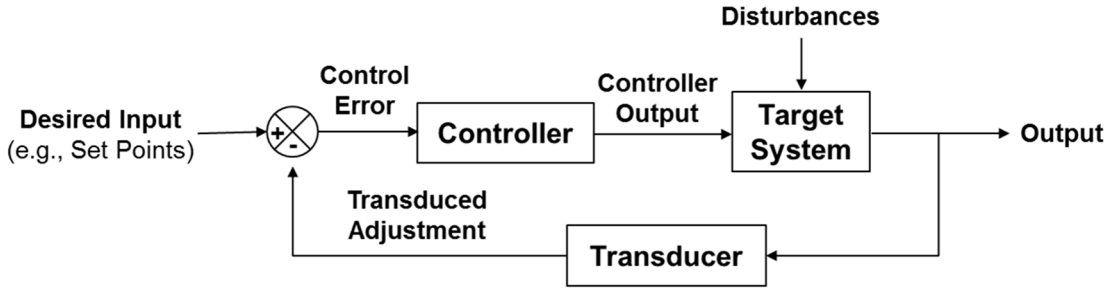


**Figure 4.3.2-1: An Exemplary Closed Control Loop**

### 4.3.3 Hierarchical Closed

A hierarchical closed control loop is a control loop that is organized in the form of a tree. This organization enables different decisions to be made by different nodes in the tree. In general, there is a set of supervisory closed control loops that allocate tasks to subordinate closed control loops. Each subordinate closed control loop performs its tasks and returns its result to its superordinate closed control loop. Advanced examples enable one of a group of designated closed control loops to take control of the hierarchy dependent on goals and the environment. This is an example of a self-organizing hierarchical closed control loop.

In general, the topmost closed control loop reasons about an abstract world model; its subordinate closed control loops reason about increasingly more specific models, or portions of models.

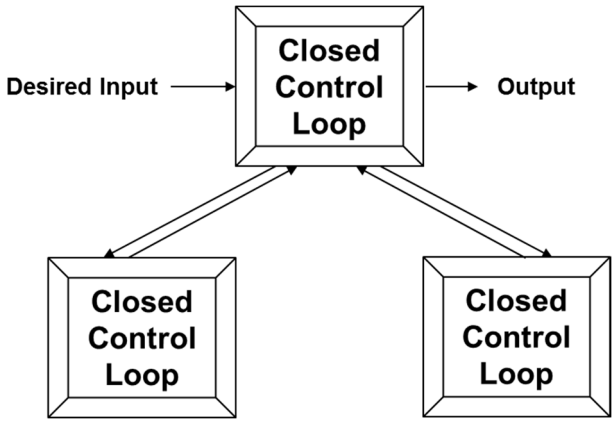


**Figure 4.3.3-1: An Exemplary Hierarchical set of Closed Control Loops**

### 4.3.4 Distributed Closed

A distributed closed control loop is a closed control loop whose components are physically distributed among different locations. Each component in a distributed closed control loop uses a message passing mechanism to communicate with one or more other components of the distributed closed control loop.

### 4.3.5 Adaptive Closed

An adaptive closed control loop is a control loop whose controlling function adapts to the object or process being controlled using parameter that are either unknown and/or vary over time. The parameters may be defined using a model that defines the desired closed loop performance, or statistical analysis to build a mathematical model from measured data.

### 4.3.6 Federated Closed

A federated closed control loop is a set of semi-autonomous closed control loops that use formal agreements to govern their interaction and behaviour. This includes rules to admit new members of the federation, as well as rules governing the visibility and types of information that can be shared with other members of the federation. Each closed control loop operates on the same goal using its own local data. Decisions from each closed control loop are then aggregated and published.

### 4.3.7 Cognitive Closed

Cognition is the process of understanding data and information and producing new data, information, and knowledge. A cognitive closed control loop selects data and behaviours to monitor that can help assess the status of achieving a set of goals, and produce new data, information, and knowledge to facilitate the attainment of those goals.

## 4.4 Prominent Control Loop Architectural Styles

### 4.4.1 OODA

Col. John Boyd's control loop [i.3], [i.4] and [i.5] consists of four phases:

- Observe, Orient, Decide and Act (OODA).

It is shown in Figure 4.4.1-1 which is drawn to emphasize how orientation shapes observation, decision, and action. While the loop appears to be sequential, this is merely for convenience. The orientation step is critical, as it determines how observations, decisions, and actions are performed. Hence, observation, orientation, and action occur simultaneously and continuously. As Boyd observed, people act according to how they perceive the world, as opposed to how the world really is.

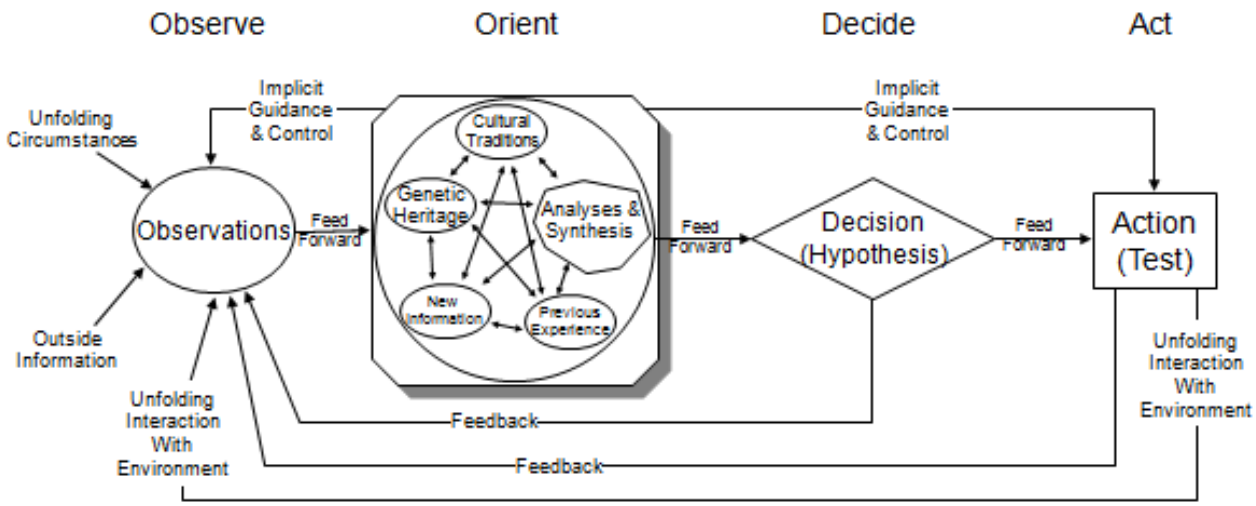


**Figure 4.4.1-1: The OODA Control Loop**

One of the strongest features of the OODA loop is to initiate or modify actions in response to observed events. If this can be transformed into a machine-understandable form, then formal logic can be applied to examine all different concurrent options to arrive at the best plan to achieve the goals of the mission. This is implemented in FOCALE, which stands for Foundation - Observe - Compare - Act - Learn - rEason; it is an adaptive and cognitive control loop (see [i.2] and [i.4]).

In stark contrast to other control loop architectures, OODA is a set of *interacting* loops, where observations in the current context are filtered (the orient phase) to make them relevant.

The OODA loop was the inspiration and foundation for FOCALE, which is an enhanced version of OODA that features the addition of cognition.

### 4.4.2 MAPE-K

In [i.7], IBM® (International Business Machines) defined the Monitor-Analyse-Plan-Execute, or MAPE, control loop. Since all 4 functions depend on the Knowledge function, it is called Model-Analyse-Plan-Execute-Knowledge (MAPE-K). It is shown in Figure 4.4.2-1.

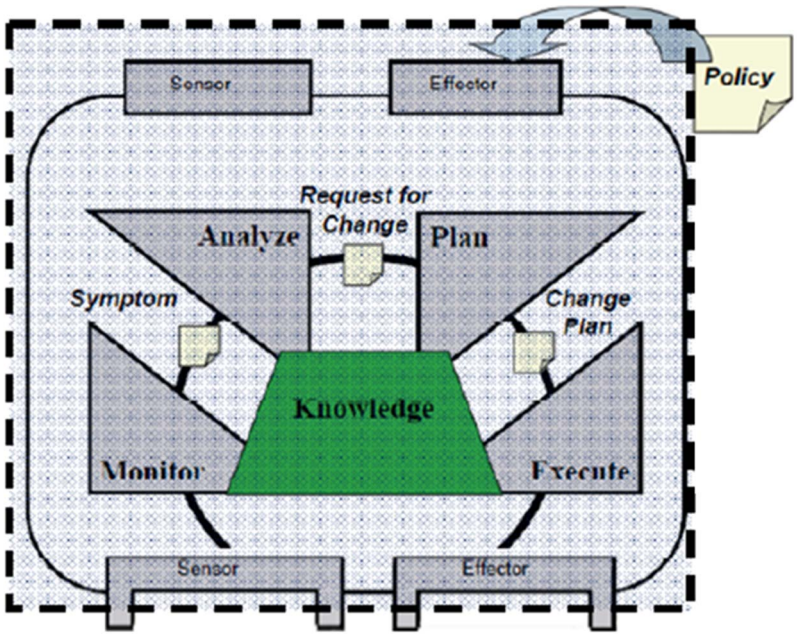


**Figure 4.4.2-1: The IBM MAPE-K Control Loop**

Sensors and Effectors get data from and provide commands to both the entity being managed and to other elements of the management system. The knowledge source implements a repository that provides access to knowledge according to the interfaces of data and information to be used by the control loop. In addition, the four control loop functions consume and generate knowledge.

### 4.4.3 FOCALE

FOCALE [i.2] and [i.4], which stands for Foundation - Observe - Compare - Act - Learn - rEason, was created to automate the complex, manually-intensive configuration tasks of network devices. The main components of FOCALE are shown in Figure 4.4.3-1. Each building block is connected using a distributed semantic Enterprise Service Bus (ESB) that supports simple as well as semantic queries. The difference between a semantic ESB and a standard ESB is that a semantic ESB can be used to orchestrate content, whereas standard ESBs are limited to orchestrating messages. The FOCALE Autonomic Manager uses the semantic ESB to *orchestrate behavior*. It can support different types of knowledge acquisition and distribution (e.g. push, pull, and scheduled) and performs common processing (e.g. semantic annotation, filtering and storage) before content is delivered to components. This enables components to register interest in knowledge in a more precise fashion, and thus reduce messaging overhead.

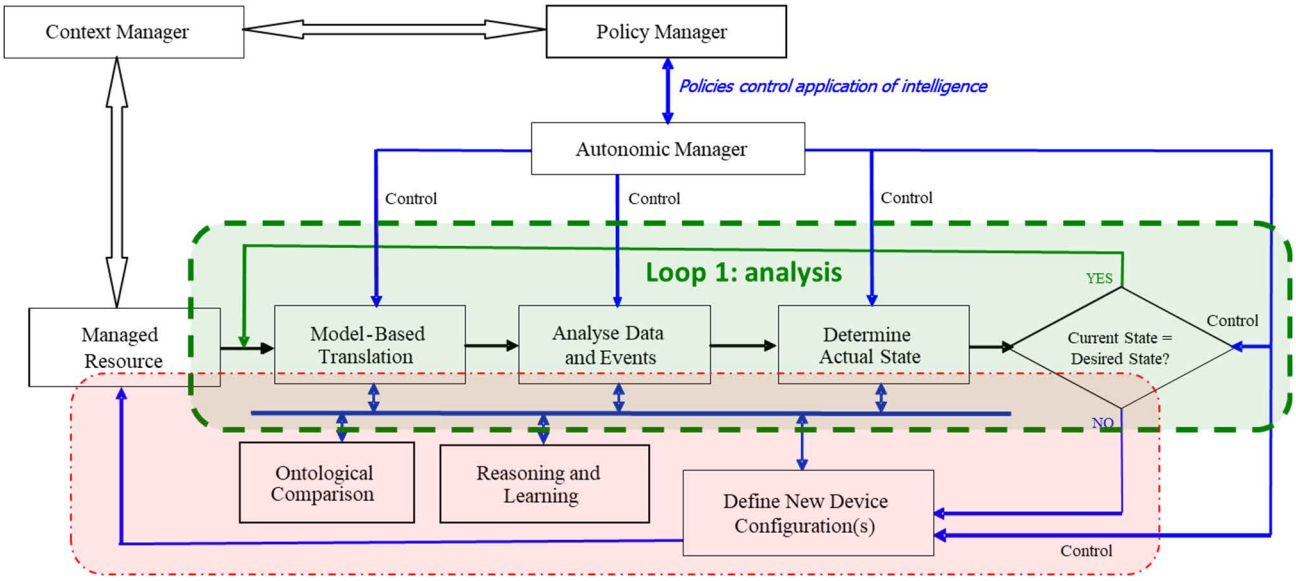


**Figure 4.4.3-1: A Simplified Version of the FOCALE Control Loop Architecture**

FOCALE uses the DEN-ng information model and the DENON-ng ontologies to translate disparate sensed data into a common networking lingua franca. DEN-ng is used to represent the static characteristics and behavior of entities; DENON-ng is then used to augment this model with consensual meaning and definitions so that domain- and vendor-specific concepts can be mapped into a common terminology. This enables facts extracted from sensor input data to be reasoned about using ontology-based inferencing. This is the foundation for Cognitive Control Loops (see clause 4.3.7) and the basis for model-driven decision-making (see [i.1] and [i.4]).

In FOCALE, sensor data is retrieved and translated from vendor- and device-specific data into a normalized form in eXtensible Markup Language (XML) using model-based mapping and ontology-based reasoning. This is then analysed to determine the current state of the managed entity. The current state is compared to the desired state from the appropriate Finite State Machines (FSMs) [i.13]. If no problems are detected, the system uses the maintenance loop; otherwise, the reconfiguration loop is used so that the services and resources provided can adapt to these new needs.

Nodes in a FOCALE FSM represent a configuration state; edges represent state transitions, and connote permission to change the configuration of a managed resource [i.13]. Static behavior is thus "programmed" into FOCALE by designing a set of FSMs; dynamic behavior is defined by altering one or more FSMs. Context-aware policy rules govern both autonomic control loops - the green analysis closed control loop and the red reconfiguration closed control loop [i.2] and [i.4]. This enables context to select the set of policies that are applicable for any situation; policies are used to then define the functionality allowed. The autonomic manager uses the current set of context-aware policies to govern each of the architectural components of the control loop, enabling each of the different control loop components to change how it operates as a function of context. As context changes, policies change, and system functionality is adjusted accordingly.

## 4.4.4 GANA

The Generic Autonomic Networking Architecture (GANA) [i.12] reference model is designed for autonomic communication, autonomic networking, and autonomic and cognitive management and control. [i.12] says that "GANA fuses [sic] a number of leading autonomics efforts/models, including FOCALE, IBM-MAPE-K, 4D architecture, Knowledge Plane for the Internet and other models, as a unified holistic reference model for Autonomic Management & Control (AMC)".

NOTE: FOCALE and ENI both use data fusion. "Fuse" means "join or blend to form a single entity" (Oxford). Just as data fusion considers complete data from multiple sources to produce a result, "fusing" elements from multiple projects should also use complete concepts.

Hence, the above reference to "fusing" FOCALE into GANA is incorrect, as FOCALE and IBM MAPE-K are fundamentally incompatible and cannot be fused for at least the following six reasons:

1) First, IBM MAPE-K (see clause 4.4.2) has no ability to either limit the amount of data to be ingested or to filter said data based on relevance or context. In contrast, FOCALE uses a semantic service bus that can do both of these tasks.

2) Second, the underlying control loops are fundamentally different:

   a) The MAPE-K is a single control loop that primarily reacts to changes in monitored data. FOCALE is a set of hierarchical control loops.

   b) MAPE-K has limited internal feedback. FOCALE uses an enhanced OODA control loop, which is much richer and contains multiple feedback sources within the control loop (see clause 4.4.1).

3) Third, FOCALE is based on AI and semantic advances to Boyd's OODA control loop (see clause 4.4.3 and [i.1]), neither of which is used by MAPE-K.

4) Fourth, FOCALE uses model-driven engineering to construct code to manage behaviour at runtime, whereas MAPE-K does not.

5) Fifth, FOCALE uses a novel cognition model that is implemented using the interaction of three Functional Blocks (see [i.1], whereas MAPE-K does not have a cognition model.

6) The above six points are then applied to compare FOCALE and GANA in clause 4.4.7. Again, it is shown that it is impossible to "fuse" FOCALE into GANA (for a set of different reasons).

Figure 4.4.4-1 shows the GANA abstraction levels for self-management functionality; this consists of interworking hierarchical and/or nested control loops. Figure 4.4.4-1 shows the key GANA Functional Blocks for implementing autonomics in target architectures. Similar to an ENI System, communication and APIs use a set of Reference Points.

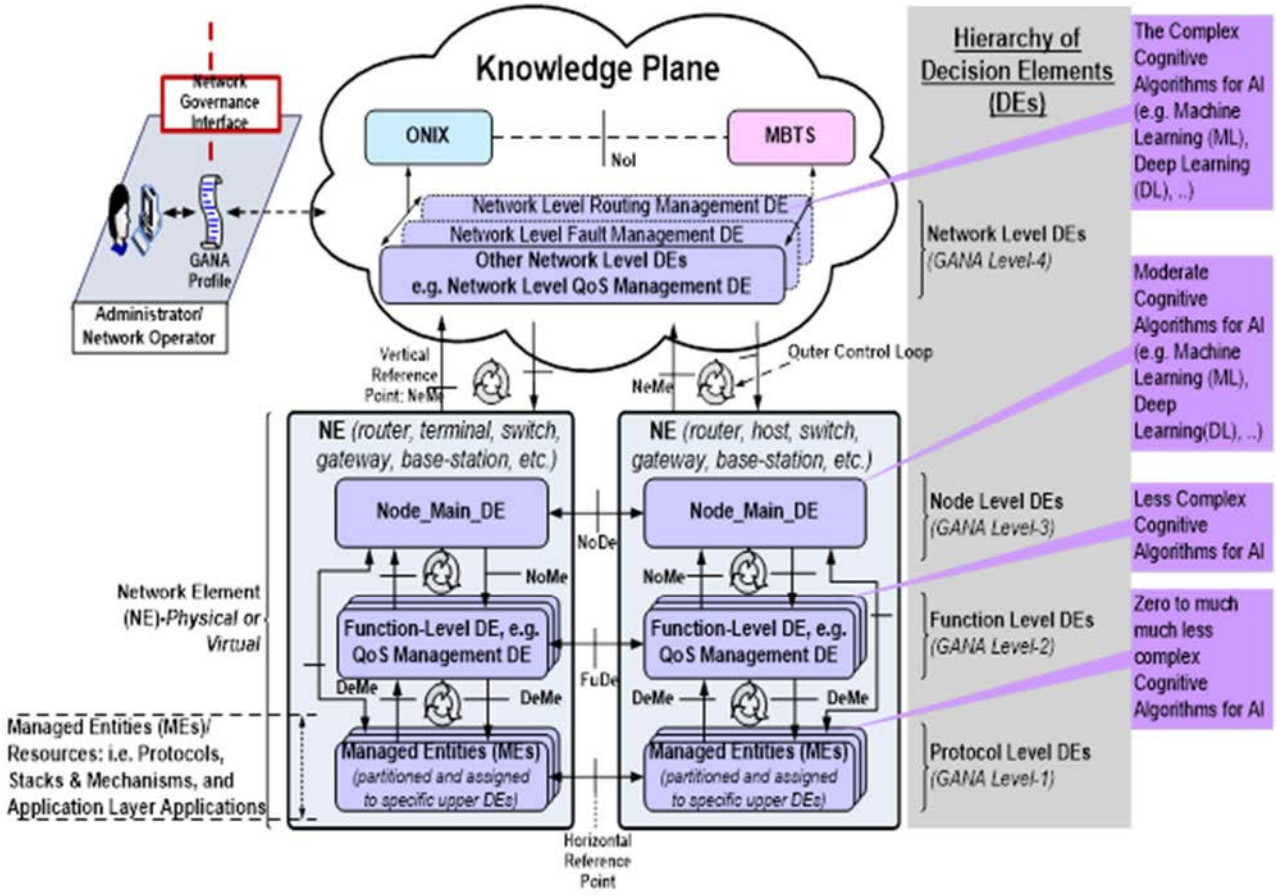


**Figure 4.4.4-1: The GANA Architecture**

A GANA Managed Element (ME) is a physical or virtual resource that can be managed by a DE (Decision Element). An ME can be composed of multiple MEs (e.g. a sub-network).

A GANA DE monitors MEs assigned to it to analyse and compare the state of the MEs against the desired state that is adaptively computed from certain objectives meant to be enforced by the DE (similar to FOCALE), and then creates a plan of actions or strategies to dynamically change the state and operations of the MEs; a selected plan is executed to change the behaviour of MEs. DEs exist as a hierarchy of decision-making functions. Level 1 DEs need to implement fast control loops (possibly with little or no cognition); going up the hierarchy, cognition in each control loop increases, so that at Level 4, DEs are cognition-based, and consequently execute relatively slowly.

Each GANA node is governed by the GANA Knowledge Plane (KP). The KP interacts with EMSs, NMSs, and OSSs through Reference Points, and also enables any of these to be "enhanced" by using replaceable and (re)-loadable DEs that can be used for specific management and control operations. DEs drive self-* operations by programmatically (re)-configuring MEs. Network-level DEs in the KP have a network-wide scope, and operate an outer control loop. Node-level DEs govern the behaviour of a node; Function-level DEs govern routing, forwarding, QoS, QoE, and similar services; Protocol-level DEs manage protocols.

The Overlay Network for Information eXchange (ONIX) is used for auto-discovery of information and entities, and implements a real-time inventory. The Model-Based Translation Service is similar to that of FOCALE. It translates technology- and vendor-specific raw data onto a common data model for use by network level DEs, based on an accepted and shared information model.

### 4.4.5 COMPA

The Control, Orchestration, Management, Policy, and Analytics (COMPA) adaptive control loop realizes an automation pattern that can operate recursively at different layers in a network. A higher-level COMPA autonomic control loop can orchestrate functions that are implemented as lower-level COMPA autonomic loops. The COMPA automation pattern can therefore recurse down to the resource level in a network. The recursive nature of the automation pattern is ideal for automating monitoring, including root cause analysis of problems.

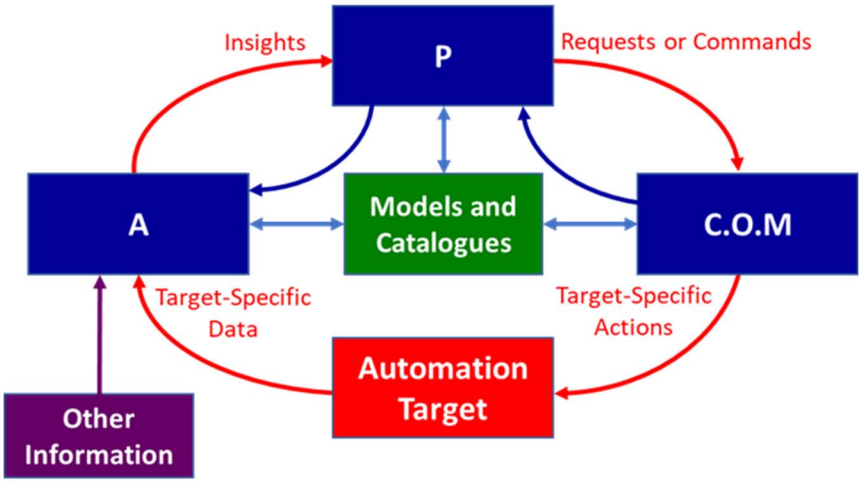


**Figure 4.4.5-1: The COMPA Architecture**

The red loop gathers data from the automation target, normalizes it (similar to FOCALE), and then uses machine learning and statistical analysis to discover and understand patterns and trends. This is them sent to the Policy function, which can recommend or decide on actions (similar to ENI). The Control-Orchestration-Management (COM) function performs non-functional, functional, and semantic validation on recommendations and actions. For example, non-functional validation checks security/access control rights and resource availability; functional validation ensures that the recommendation or action is functionally correct; semantic validation would test the recommendation or action against a semantic model to infer any potential deviations. Once validated, the recommendation or action can then be applied by translating the technology-neutral recommendation or action specification into a vendor-specific representation required to execute the recommendation or action.

The blue internal feedback flow enables the loop to self-stabilize. Feedback from validations can therefore be used to direct the decision making. Feedback from the decision making about input patterns and predictions and user feedback can also be used to tune analytics and complex event processing.

### 4.4.6 Cognitive Control Loops (FOCALE v3)

The original FOCALE control loops are loosely based on the OODA loops, as shown in Figure 4.4.1-1. The Model-Based Translation function corresponds to the Orient function of OODA - it transforms raw sensor data into a form that can be correlated with the current context. The Analyse, Determine State, and Compare functions of FOCALE correspond to the Decide function of OODA, except that OODA is not focused on state, whereas FOCALE is; this is because FOCALE uses state to orchestrate behavior. This is reflected in the Foundation function of FOCALE as well. FOCALE v3 [i.8] is based on research about how cognition can be more effectively used to govern behaviour. A cognitive system can learn from experience, and examine its own capabilities to optimize its operation. A truly cognitive system requires a formal cognitive model, which was the focus of FOCALE v3.

The foundation of the FOCALE v3 cognitive model is based on work done by Minsky, who developed a model of human intelligence that is built using agents, which interact according to three layers, called reactive (or subconscious), deliberative, and reflective [i.9]. Reactive processes respond immediately when they receive an appropriate external stimulus. FOCALE v3 extends this reaction to protect the goals of the system. Reactive processes have no understanding of what external events "mean"; rather, the process simply responds with some combination of pre-defined and learned reactions.

Deliberative processes receive data from, and can send "commands" to, reactive processes; however, they do not interact directly with the external world. This part of the brain is responsible for our ability to achieve more complex goals by applying short- and long-term memory in order to create and carry out more elaborate plans. This knowledge is accumulated and generalized from experience and what is learned.

Reflective processes supervise the interaction between the deliberative and reactive processes. These processes enable the brain to reformulate and reframe its interpretation of the situation in a way that may lead to more creative and effective strategies. It considers what predictions turned out wrong, along with what obstacles and constraints were encountered, in order to prevent sub-optimal performance from occurring again. It also includes self-reflection - the analysis of its own performance, as well as how well the actions that were taken solved the problem at hand.

The new FOCALE v3 cognition model replaces the original two alternative reflective control loops with a hierarchical approach: a set of outer control loops are used for "large-scale" adjustment of functionality by reacting to context changes, and a set of inner control loops that provide more granular adjustment of functionality based on a particular situation for each associated outer control loop. In addition, both the outer and the inner control loops use reactive, deliberative, and reflective reasoning, as appropriate.

The inner loops have two important changes. First, the Decide function has been explicitly unbundled from the Act function. This enables additional machine-based learning and reasoning processes to participate in determining which actions should be taken. Second, the new cognition model has changed the old FOCALE control loops, which were both reflective in nature, to a set of three control loops that are reactive, deliberative, and reflective. This streamlines analysis and processing, freeing the resources of an individual FOCALE system to be used for other purposes (similar to the recursive COMPA approach).

### 4.4.7 Comparison

Of the six control loops described, MAPE-K is an outlier, as all other control loops are based on OODA.

There are several problems with the MAPE-K architecture. First, the control loop is gated by the monitoring function. Hence, if too much data floods the system, the performance of the rest of the system suffers, even if the monitored data is not relevant. Second, the Autonomic Manager interacts with the entity that it is managing, and not with the environment. Hence, it is very difficult for the environment to influence the operation of the control loop. Third, there is no guarantee that an Autonomic Manager has the ability to perform a desired function, as communication depends on the loose concept of an interface. This makes it difficult to capture semantics, such as dependencies on other components, side effects, and pre- and post-conditions required for a function to be successfully invoked. In contrast, FOCALE uses software contracts [i.10] and [i.11] to provide interoperable specifications of the functional, operational, and management aspects of a feature. Fourth, the previous point implies that this represents an exclusive control mechanism, and is not intended to be combined with other approaches, like FOCALE, GANA, COMPA, and FOCALE v3 provide. Fifth, it is also unclear how distribution is supported in this design. Most distributed systems support a set of core services, such as naming, messaging, and transaction management, upon which higher level services, such as policy management and security, are built; no such services are discussed in the MAPE-K architecture. Finally, this design uses the Autonomic Manager as the component to be distributed. As can be seen, the Autonomic Manager is a relatively complex entity designed to realize a given set of pre-defined functions. This makes it very difficult for the Autonomic Manager to support emergent behavior.

The OODA loop does have some deficiencies. For example, goals are not explicitly shown, and the model needs some modifications for collaborative decision-making (e.g. shared situation awareness, task re-allocation, negotiation of goals, confirmation and authorization of decisions). Most importantly, cognition is not present; this was one of the primary motivations for FOCALE, which is an enhanced version of OODA (see clause 4.4.1). Finally, attention and memory, as well as a cognitive representation of the world, need to be added. However, in stark contrast to the MAPE-K control loop (see clause 4.4.2), OODA is a set of *interacting* loops, where observations in the current context are filtered (the orient phase) to make them relevant. Note how orientation shapes observation, shapes decision, shapes action, and in turn is shaped by the feedback and other phenomena coming into our sensing or observing window. The need to orient observations is the inspiration for the FOCALE model-based translation layer, which orients observed data to the current context. Finally, unlike the MAPE-K loop, this is not a sequential loop. First, a balance needs to be maintained between delaying decisions (which means delaying actions) and performing more accurate analysis that eliminates Autonomic Manager the need to revisit previously made decisions. Second, both the speed of re-orientation as well as being able to apply suitable actions via the implicit guidance and control link to Action are critical for supporting decision-making. This enables a simpler observe-orient-act control loop to be employed in situations that can benefit from this. As will be seen, these principles have inspired the design of FOCALE.

There are a number of similarities between FOCALE and OODA. For example, the need to orient observations is the inspiration for the FOCALE model-based translation layer, which orients observed data to the current context and translates different data sources into a single neutral form to facilitate their correlation and integration. Like OODA, FOCALE is not a sequential loop. This enables simpler control loops to be employed in situations that warrant it. The most prominent differences between FOCALE and OODA are:

1) FOCALE uses a semantic ESB, which enables feedback between any FB (Functional Block).

2) It also enables the control loop to be interrupted at any point. Second, it adds machine learning and reasoning, supported by formal logic in the form of ontologies.

3) Finally, it adds Model-Driven Engineering (MDE) functionality, which enables a single normalized technology-neutral representation to generate technology- and vendor-specific commands.

There are a number of differences between GANA and FOCALE; note that ENI is similar to FOCALE but contains some differences to how it interacts with external entities (e.g. it uses an API Broker):

1) First, there are significant differences between how the terms "information model", "data model", "cognition", "knowledge", and "ontology" are defined between GANA and FOCALE. This undermines the claim that FOCALE or ENI can be "fused" into GANA [i.12] (see also clause 4.4.4).

2) Model-Based Translation (MBT). The MBT is at different architectural abstraction levels in GANA vs. FOCALE (and ENI) and performs different functions:

    a) In GANA, the MBT operates above the managed and network elements ("*The GANA Knowledge Plane consists of the following Functional Blocks: GANA Network Level Decision Elements (DEs), Model-Based Translation Services (MBTS) Functional Block …* [i.12]" More importantly, [i.12] says "*MBTS… which is an intermediation layer between the GANA KP DEs and the NEs (physical or virtual) for translating technology specific and/or vendors' specific raw data onto a common data model for use by network level DEs, based on an accepted and shared information/data model*". Finally, there are different types of MBTSs: one for DE-NE interaction, one for translating a proprietary operator network profile, and one for translating operations and primitives between domains, and one for translating information retrieved from NEs to the language understood by the KP and vice-versa.

    b) In FOCALE, the MBT is a set of agents that operate immediately on ingested data and convert those data to a single normalized internal form. This enables all succeeding Functional Blocks to only use a single representation of data, information, and knowledge, and helps synchronize the operation of all Functional Blocks. When FOCALE is done processing, it uses the MBT to perform the reverse operation.

    c) There are three profound differences in how an MBT is used in GANA vs FOCALE:

        i) In GANA, the MBT is used for translating network element information. In FOCALE, the MBT is used for translating any type of input data or information to its normalized form, regardless of whether it came from a network element or not.

        ii) In GANA, there are three different types of translation services. In FOCALE, there is one.

iii) GANA abstracts the concept of model-based translation to translate between different elements that have associated models. In fact, [i.12] says "*MBTS 'maps' Ontologies or Information Models used to describe node-local behaviour which needs to be shared with the Knowledge Plane*". It also says "*Any suitable frameworks for policy based management and associated information models for policy management and orchestration can be applied for the GANA DEs*". This is an unsolvable problem for models in general and ontologies in particular, as there is no one format (standard or otherwise) that enables interoperability for information models or ontologies. In contrast, FOCALE uses its MBT strictly to translate incoming data and information to a single normalized form used throughout FOCALE, and again from that normalized form to a vendor-specific form. This is a much simpler task, as the MBT is only taking data and information (and specifically NOT models and ontologies) and translating to and from a single internal format.

3) Knowledge Plane. The Knowledge Plane [i.14] had a noble goal: *"to build a fundamentally different sort of network that can assemble itself given high level instructions, reassemble itself as requirements change, automatically discover when something goes wrong, and automatically fix a detected problem or explain why it cannot do so*".

NOTE: This topic is just about the use of the Knowledge Plane concept. It does not discuss different types of knowledge or how they are used.

a) There is no concept of the Knowledge Plane in FOCALE or ENI. In particular, there is no concept of "self-assembly". Rather, each function is defined as a modular Functional Block, and there is a cognition layer that realizes a novel cognition model that manages behaviour.

b) [i.14] defines the Knowledge Plane as a "*construct that exhibits cognitive capabilities and behaviours in performing the management and control of networks and services, and operates on network-wide views (including knowledge continuously gathered about the state and behaviours of network elements) to dynamically and autonomically program (configure) network resources, services and their configurable parameters- as governed by the business and technical objectives required of the network… This definition of the Knowledge Plane is tailored to the GANA Knowledge Plane and is adopted from the definition…by Clark, Partridge, Ramming and Wroclawski, and refined accordingly for the GANA autonomic network*".

4) Control Loops. There is a significant difference in how the GANA and FOCALE architectures use control loops:

a) [i.12] defines four basic hierarchical control loops. These are used as levels of abstractions in which associated Decision Elements (DEs) that provide autonomic functions may be introduced within the network, namely:

- Protocol-Level;
- Function-Level;
- Node-Level; and
- Network-Level (in the realm called the GANA Knowledge Plane)", though it recommends only concentrating on the latter three.

It also talks about "fast" and "slow" control loops.

b) FOCALE is less worried about "fast vs. slow" execution of a control loop. Rather, FOCALE concentrates on the current set of tasks that a control loop is performing. FOCALE has the ability to create new control loops as needed based on the task being executed, including nested control loops.

c) Hence, the differences are:

i) GANA starts with control loops at three different architectural abstractions (function, node, and network), whereas FOCALE starts with two OODA-inspired control loops that are at the same architectural level.

ii) GANA explicitly associates different cognition functionality and speed with its three control loops, whereas FOCALE associates control loops to a set of tasks, where each control loop uses common cognition services.

iii) FOCALE can dynamically create and delete control loops based on accomplishing a set of tasks. In contrast, each GANA DE's control loop "realizes some specific control-loop(s) with its own algorithms and logic." The difference is that GANA creates control loops at a particular architectural point, whereas FOCALE creates control loops to accomplish a specific set of tasks.

iv) [i.12] implies that the terms "nested control loop" and "hierarchical control loop" are the same, whereas in FOCALE, they are different. Specifically, a hierarchical control loop in FOCALE is associated with an abstraction level, whereas a nested control loop is not (nested means that a parent control loop has paused execution because it spawned a child control loop; when the child control loop is finished, the parent control loop will resume execution).

v) One of the main additional differences is that the GANA control loops are focused on supporting network configuration and management. In contrast, the OODA control loop central to ENI and FOCALE, in combination with its cognition model, do more than this - it associates different business rules, regulatory policies, environmental conditions, and other factors to achieve a set of goals that may affect the behaviour of the system (the "goal-policies" mentioned in [i.12] are a subset of FOCALE's more generic concept of goals). Hence, FOCALE's control loops are used to understand the environment and other external factors and then perform other tasks, such as network configuration and management.

5) Cognition. Cognition is also fundamentally different between GANA and FOCALE.

a) Cognition in GANA is defined as "learning, analysing and reasoning capability used to effect advanced adaptation of behaviour or state of an entity for which the capability is being employed." GANA does not define a cognition model.

b) Cognition is defined in ENI (and FOCALE) as the "process of acquiring and understanding data and information and producing new data, information, and knowledge." FOCALE also defines a novel and explicit cognition model.

c) The difference between GANA and FOCALE is threefold:

i) GANA uses cognition to manage the behaviour of an entity. In contrast, FOCALE uses cognition to understand how the environment is changing and what to do to support the goals that it is currently trying to achieve.

ii) FOCALE in addition uses cognition to produce new data, information, and knowledge, which GANA does not do.

iii) Most importantly, FOCALE defines a cognition model, which defines which cognitive processes are performed. GANA does not have a cognition model.

6) Semantic Service Bus. This is an Enterprise Service Bus that can also route messages based on meaning. This is not present in GANA, but is a primary mechanism of sharing information in FOCALE.

COMPA is also similar to FOCALE. It uses the concepts of MDE and normalization, and has functionality similar to the software contracts used by FOCALE v3. If COMPA is compared to OODA, then the A (analytics) corresponds to the observe, orient, and decide functions, while the P (policies) and COM (Control, Orchestration, and Management) map to the action function. It is similarly difficult to compare COMPA to FOCALE (any version) or GANA. This is not to say that COMPA cannot be used; rather, it means that COMPA emphasizes governing automation targets, while FOCALE and GANA emphasize the decision-making that leads to governance.

FOCALE v3 is recursive, much like COMPA. FOCALE v3 is the only architecture that uses context and situation awareness as part of its decision-making process (inside the control loop). It is also the only architecture that uses the principles of cognition, and emphasizes reactive, deliberative, and reflective processes. Thus, it is the main inspiration for the ENI architecture.

## 4.5 Domains and Control Loops

### 4.5.1 Introduction

A Domain is a mechanism used to define a collection of different Entities that share a common purpose. Control loops use this property to manage the behaviour of the Entities contained in a Domain in a common way (e.g. using the same type of policies).

Control loops exist entirely within a single Administrative Domain. The ENI System architecture enables coordination between multiple control loops within the same and/or across different Administrative Domains; see clause 6.2.4 in ETSI GS ENI 005 [i.1] for an explanation of how conflicts between two or more control loops are resolved.

### 4.5.2 Administrative Domains and Control Loops

An Administrative Domain is a Domain that employs a set of common administrative processes to manage the behaviour of its constituent Entities.

When two or more control loops exist in the same Administrative Domain, then the same set of policies is used to govern the behaviour of all Entities in that Administrative Domain (see ETSI GS ENI 005 [i.1], clauses 5.2 and 6.2.4 for elaboration of this principle). In particular, an Administrative Domain may consist of one or more nested Administrative Domains, forming a hierarchy. In this case, all child Administrative Domains of a given parent Administrative Domain implement the same behaviour, and do not implement behaviour that conflicts with the behaviour defined in a parent Management Domain.

When two or more control loops exist in different Administrative Domains and need to be coordinated, then the coordination may be between peer and/or hierarchical control loops. Peer control loops are two or more independent control loops that each reside in a different Administrative Domain that interact with each other, and hence, require coordination of their actions. In this case, the policies for each of the peer control loops are examined for conflicts before the interaction takes place, as described in ETSI GS ENI 005 [i.1], clause 6.2.4.

Hierarchical control loops in one Administrative Domain interact with hierarchical control loops in a different Administrative Domain in a similar way as peer control loops. Typically, the parent control loop for each Administrative Domain is responsible for coordinating the actions of its control loop with the actions of the hierarchical control loops of the different Administrative Domains that it is interacting with. In particular, the complete set of policies of the parent control loop and all of its child control loops are compared with the complete set of policies of the parent control loop and all of its child control loops for each different Administrative Domain.

### 4.5.3 Management Domains and Control Loops

A Management Domain is an Administrative Domain that uses a set of common Policies to govern its constituent Entities. This includes the definition of a set of administrators that govern the Entities that it contains, along with a set of applications that are responsible for different governance operations that use a common set of management mechanisms.

The interaction of peer and hierarchical control loops in a Management Domain is similar to the interaction of peer and hierarchical control loops in an Administrative Domain, as described in clause 4.5.2. The main difference is that in addition to comparing the Policies of each Management Domain, the applications and/or administrators that implement the Policies also need to be coordinated.

### 4.5.4 Collaborating Control Loops in the Same System

NOTE: This is for further study.

### 4.5.5 Collaborating Control Loops in Different Systems

NOTE: This is for further study.

# 5 Summary and Recommendations

The present document has described several principles for constructing different types of closed control loops. Six important control loop architectural styles were described and compared. Of these, the progression of the original FOCALE closed control loops to FOCALE v3 is notable, as the latter changed the control loop from an adaptive closed control loop to an adaptive and cognitive closed control loop. It also added a robust cognition model.

These principles are all used in the design of the ETSI GS ENI 005 [i.1], and are applicable to other ETSI reports and standards.

Thus, the present document recommends that the contents of the present document are applicable as primary concepts that sustain the ETSI GS ENI 005 [i.1] and other related documents and specifications.

# History

| Document history | | |
|---|---|---|
| V2.1.1 | August 2021 | Publication |
| V2.2.1 | June 2024 | Publication |
| | | |
| | | |
| | | |