ETSI TR 103 626 V1.1.1 (2020-02)

**Autonomic network engineering
for the self-managing Future Internet (AFI);
An Instantiation and Implementation of the
Generic Autonomic Network Architecture (GANA)
Model onto Heterogeneous Wireless Access Technologies
using Cognitive Algorithms**

Reference
DTR/INT-001-AFI-0027

Keywords
autonomic networking, cognition, cognitive,
control, radio, self-management

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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.
The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2020.
All rights reserved.

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.

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 IPR Policy, no investigation, 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.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Core Network and Interoperability Testing (INT).

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.

1 Scope

The present document provides a mapping of architectural components for autonomic network management & control developed/implemented in the European Commission (EC) funded WiSHFUL Project to the ETSI TC INT AFI Generic Autonomic Networking Architecture (GANA) model - an architectural reference model for autonomic networking, cognitive networking and self-management. The mapping pertains to architectural components for autonomic decision-making and associated control-loops in wireless network architectures and their associated management and control architectures.

The objective is to illustrate how the GANA can be implemented using the components developed in the WiSHFUL and ORCA Projects. To show the extent to which the WiSHFUL architecture augmented with some virtualization and hardware acceleration techniques, developed in the ORCA project, implements the GANA model, in order to guide the industry (implementers of autonomics components for autonomic networks and their associated autonomic management & control architectures) on how to leverage this work in their efforts on GANA implementations.

The mapping of the components to the GANA model concepts serves to illustrate how to implement the key abstraction levels for autonomics (self-management functionality) in the GANA model for the targeted wireless networks context, taking into consideration the work done in ETSI TR 103 495 [i.7].

The other objective is to also illustrate the value of joint autonomic management and control of heterogeneous wireless access technologies in such a GANA implementation context, with illustration on where Cognitive algorithms for autonomics (such as Machine Learning and other AI algorithms) can be applied in joint autonomic management & control of heterogeneous wireless access networks.

The present document answers the question of how to implement the ETSI GANA model using WiSHFUL architecture and ORCA concepts.

NOTE: Trademarks in the present document that are associated with the environments considered by WiSHFUL and ORCA projects in their implementation and prototyping of components are only mentioned as Citation of the environments on which components were implemented by the the two projects. The purpose of the present document is to illustrate to the industry how such WiSHFUL and ORCA components can be used to implement the ETSI GANA components in such environments considered by the projects, while making it clear that other environments not considered by the two projects can also be considered by the industry in implementing GANA components, as the present document does not serve to endorse or limit environments in which the GANA components can be implemented.

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] Joao F. Santos, Jonathan van de Belt, Wei Liu, Vincent Kotzsch, Gerhard Fettweis, Ivan Seskar, Sofie Pollin, Ingrid Moerman, Luiz A. DaSilva and Johann Marquez-Barja: "Orchestrating next-generation services through end-to-end network slicing", ORCA white paper.

NOTE: Available at https://orca-project.eu/wp-content/uploads/sites/4/2018/10/orchestrating_e2e_network_slices_Final.pdf.

[i.2] ORCA Deliverable 4.3: "Enhanced operational SDR platforms with end-to-end capabilities".

NOTE: Available at https://orca-project.eu/wp-content/uploads/sites/4/2019/02/ORCA_D4.3_final.pdf.

[i.3] WiSHFUL Project Deliverable D3.2: "First operational radio control software platform".

[i.4] WiSHFUL Project Deliverable D3.4: "Second operational radio control software platform".

[i.5] WiSHFUL Project Deliverable D4.2: "First operational network control software platform".

[i.6] WiSHFUL Project Deliverable D4.4: "Second operational network control software platform".

[i.7] ETSI TR 103 495: "Network Technologies (NTECH); Autonomic network engineering for the self-managing Future Internet (AFI); Autonomicity and Self-Management in Wireless Ad-hoc/Mesh Networks: Autonomicity-enabled Ad-hoc and Mesh Network Architectures".

[i.8] Tayeb Ben Meriem, Ranganai Chaparadza, Benoît Radier, Said Soulhi, José-Antonio Lozano-López, Arun Prakash, ETSI White Paper No. 16: "GANA - Generic Autonomic Networking Architecture - Reference Model for Autonomic Networking, Cognitive Networking and Self-Management of Networks and Services", First edition, October 2016
ISBN No. 979-10-92620-10-8.

[i.9] ETSI TS 103 195-2 (V1.1.1) (2018-05): "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.10] ETSI TR 103 473 (V1.1.2) (2018-12): "Evolution of management towards Autonomic Future Internet (AFI); Autonomicity and Self-Management in the Broadband Forum (BBF) Architectures".

[i.11] ETSI TR 103 404: "Network Technologies (NTECH); Autonomic network engineering for the self-managing Future Internet (AFI); Autonomicity and Self-Management in the Backhaul and Core network parts of the 3GPP Architecture".

[i.12] IEEE 802.11™-2016: "IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications".

[i.13] IEEE 802.15.4™: "IEEE Standard for Low-Rate Wireless Networks".

[i.14] White Paper No.2 of the ETSI 5G: "PoC: ONAP Mappings to the ETSI GANA Model; Using ONAP Components to Implement GANA Knowledge Planes and Advancing ONAP for Implementing ETSI GANA Standard's Requirements and C-SON: ONAP Architecture".

NOTE Available at https://intwiki.etsi.org/index.php?title=Accepted_PoC_proposals.

[i.15] ETSI GS AFI 002: "Autonomic network engineering for the self-managing Future Internet (AFI); Generic Autonomic Network Architecture (An architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management)".

[i.16] ETSI INT PoC: "5G Network Slices Creation, Autonomic Management & E2E Orchestration, with Closed-Loop (Autonomic) Service Assurance for the Slices: IoT (Smart Insurance) Use Case".

NOTE Available at https://intwiki.etsi.org/index.php?title=Accepted_PoC_proposals.

[i.17] Advanced Python Scheduler.

NOTE Available at http://apscheduler.readthedocs.io/en/latest/.

[i.18] ETSI TS 103 194: "Network Technologies (NTECH); Autonomic network engineering for the self-managing Future Internet (AFI); Scenarios, Use Cases and Requirements for Autonomic/Self-Managing Future Internet".

[i.19] WiSHFUL UPI reference specification for management (M), Network (N), Radio (R) interfaces as well as network helpers.

NOTE Available at https://wishful-project.github.io/wishful_upis/index.html.

[i.20] Report on Specifications of Integration APIs for the ETSI GANA Knowledge Plane Platform with Other Types of Management & Control Systems, and with Info/Data/Event Sources in general.

NOTE Available at https://intwiki.etsi.org/index.php?title=Accepted_PoC_proposals.

[i.21] Dunkels A., Gronvall B., Voigt T.: "Contiki a Lightweight and Flexible Operating System for Tiny Networked Sensors". In Proceedings of the 9th Annual IEEE™ International Conference on Local Computer Networks, Washington, DC, USA, October 2004; pp. 455-462.

[i.22] E. Blossom. Gnu software radio.

NOTE Available at http://gnuradio.org.

[i.23] Ruckebusch P., De Poorter E., Fortuna C., and Moerman I. (2016): "GITAR: Generic extension for Internet-of-Things ARchitectures enabling dynamic updates of network and application modules". Ad Hoc Networks, Volume 36, Part 1, January 2016, Pages 127-151.

[i.24] WiSHFUL Project Deliverable D2.1: "High level requirements for testbeds and software platforms".

[i.25] WiSHFUL Project Deliverable D2.2: "Specification of first showcases".

[i.26] WiSHFUL UPI definition.

NOTE: Available at https://wishful-project.github.io/wishful_upis/wishful_upis.html.

[i.27] ZeroMQ Realtime Exchange Protocol.

NOTE Available at http://rfc.zeromq.org/spec:36.

[i.28] ORCA (Orchestration and Reconfiguration Control Architecture) project website.

NOTE Available at https://www.orca-project.eu.

[i.29] Tarik Kazaz, Wei Liu, Xianjun Jiao, Ingrid Moerman, Francisco Paisana, Clemens Felber, Vincent Kotzsch, Ivan Seskar, Tom Vermeulen, Sofie Pollin, Martin Danneberg and Roberto Bomfin: "Orchestration and Reconfiguration Control", EUCNC June 2017. Oulu, Finland.

[i.30] ORCA Deliverable 2.1: "Technical requirements of the ORCA test facility".

NOTE Available at https://orca-project.eu/wp-content/uploads/sites/4/2017/01/ORCA_D2.2_Final_v1.1.pdf.

[i.31] Wei Liu, Joao F. Santos, Jonathan van de Belt, Xianjun Jiao, Ingrid Moerman, Johann Marquez-Barja, Luiz DaSilva and Sofie Pollin: "Enabling Virtual Radio Functions on Software Defined Radio for Future Wireless Networks", to appear in Wireless Personal Communications.

[i.32] R. Chaparadza, et al.: "SDN Enablers in the ETSI AFI GANA Reference Model for Autonomic Management & Control (emerging standard), and Virtualisation Impact". In the proceedings of the 5th IEEE™ MENS Workshop at IEEE Globecom 2013, December, Atlanta, Georgia, USA.

[i.33] White Paper No.4 of the ETSI 5G PoC: "ETSI GANA as Multi-Layer Artificial Intelligence (AI) Framework for Implementing AI Models for Autonomic Management & Control (AMC) of Networks and Services; and Intent-Based Networking (IBN) via GANA Knowledge Planes".

NOTE Available at https://intwiki.etsi.org/index.php?title=Accepted_PoC_proposals.

[i.34] White Paper No.1: "C-SON Evolution for 5G, Hybrid SON Mappings to the ETSI GANA Model, and achieving E2E Autonomic (Closed-Loop) Service Assurance for 5G Network Slices by Cross-Domain Federated GANA Knowledge Planes".

NOTE Available at https://intwiki.etsi.org/images/ETSI_GANA_in_5G_PoC_White_Paper_No_1_v1.28.pdf.

[i.35] White Paper No.3: "Programmable Traffic Monitoring Fabrics that enable On-Demand Monitoring and Feeding of Knowledge into the ETSI GANA Knowledge Plane for Autonomic Service Assurance of 5G Network Slices; and Orchestrated Service Monitoring in NFV/Clouds".

NOTE Available at https://intwiki.etsi.org/images/ETSI_5G_PoC_White_Paper_No_3_2019_v1.19.pdf.

[i.36] White Paper No.5: "Artificial Intelligence (AI) in Test Systems, Testing AI Models and the ETSI GANA Model's Cognitive Decision Elements (DEs) via a Generic Test Framework for Testing ETSI GANA Multi-Layer Autonomics & their AI Algorithms for Closed-Loop Network Automation".

NOTE Available at https://intwiki.etsi.org/index.php?title=Accepted_PoC_proposals.

[i.37] White Paper No.6: "Generic Framework for Multi-Domain Federated ETSI GANA Knowledge Planes (KPs) for End-to-End Autonomic (Closed-Loop) Security Management & Control for 5G Slices, Networks/Services".

NOTE Available at https://intwiki.etsi.org/index.php?title=Accepted_PoC_proposals.

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

Autonomic Behaviour (AB): process which understands how desired Managed Entity (ME) behaviours are learned, influenced or changed, and how, in turn, these affect other elements, groups and networks [i.18]

NOTE: In the GANA model, an autonomic behaviour is any behaviour of a DE that is observable on its interfaces. A GANA DE is also called an Autonomic Function (AF).

autonomic networking: networking paradigm that enables network devices or elements (physical or virtual) and the overall network architecture and its management and control architecture to exhibit the so-called self-managing properties, namely:

- Auto-discovery of information and entities
- Self-configuration (auto-configuration), Self-diagnosing, Self-repair (Self-healing)
- Self-optimization, and other self-* properties

NOTE 1: Autonomic Networking can also be interpreted as a discipline involving the design of systems (e.g. network nodes) that are self-managing at the individual system levels and together as a larger system that forms a communication network of systems.

NOTE 2: The term "autonomic" comes from the autonomic nervous system (a closed control loop structure), which controls many organs and muscles in the human body. Usually, humans are unaware of its workings because it functions in an involuntary, reflexive manner - for example, humans do not notice when their heart beats faster or their blood vessels change size in response to temperature, posture, food intake, stressful experiences and other changes to which human are exposed. And their autonomic nervous system is always working [i.18].

Decision Making Element (DME): functional entity designed and assigned to autonomically manage and control its assigned Managed Entities (MEs) by dynamically (re)-configuring the MEs and their configurable and controllable parameters in a closed-control loop fashion

NOTE 1: Decision Making Elements (DMEs) [i.19] referred in short as Decision Elements (DEs) fulfil the role of Autonomic Manager Elements.

NOTE 2: In GANA a DE is assigned (by design) to very specific MEs that it is designed to autonomically manage and control (ETSI GS AFI 002 [i.15] provides more details on the notion of ownership of MEs by specific DEs required in a network element architecture and the overall network architecture).

Managed Entities (MEs): physical or logical resource that can be managed by an Autonomic Manager Element (i.e. a Decision Element) in terms of its orchestration, configuration and re-configuration through parameter settings [i.18]

NOTE: MEs and their associated configurable parameters are assigned to be managed and controlled by a concrete DE such that an ME parameter is mapped to one DE. MEs can be protocols, whole protocol stacks, and mechanisms, meaning that they can be fundamental functional and manageable entities at the bottom of the management hierarchy (at the fundamental resources layer in a network element or node) such as individual protocols or stacks, OSI layer 7 or TCP/IP application layer applications and other types of resources or managed mechanisms hosted in a network element (NE) or in the network in general, whereby an ME exposes a management interface through which it can be managed. MEs can also be composite MEs such as whole NEs themselves (i.e. MEs that embed sub-MEs).

OpenWRT: According to https://openwrt.org/ OpenWRT is a Linux™ operating system for people who want to install high-performance, easily-configured, reliable and robust firmware on a home router or embed the Linux-based software in other equipment.

overlay: logical network that runs on top of another network

EXAMPLE: Peer-to-peer networks are overlay networks on the Internet. They use their own addressing system for determining how files are distributed and accessed, which provides a layer on top of the Internet's IP addressing.

self-advertising: capability of a component or system to advertise its self-model, capability description model, or some information signalling message (such as an IPv6 router advertisement message) to the network in order to enable other entities to discover it and be able to communicate with it, or to enable other entities to know whatever is being advertised

self-awareness: capability of a component or system to "know itself" and be aware of its state and its behaviours

NOTE: Knowledge about "self" is described by a "self-model".

self-configuration: capability of a component or system to configure and reconfigure itself under varying and unpredictable conditions

self-healing: capability of a component or system to detect and recover from problems (manifestations of faults, errors, failures, and other forms of degradation) and continue to function smoothly

self-monitoring: capability of a component or system to observe its internal state, for example by monitoring quality-of-service metrics such as reliability, precision, rapidity, or throughput

self-optimization: capability of a component or system to detect suboptimal behaviours and optimize itself to improve its execution

self-organizing function: function that includes processes which require minimum manual intervention

self-regulation: capability of a component or system to regulate its internal parameters so as to assure a quality-of-service metric such as reliability, precision, rapidity, or throughput

3.2 Symbols

Void.

3.3 Abbreviations

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

3GPP	3rd Generation Partnership Project
AB	Autonomic Behaviour
AF	Autonomic Function
AFI	Autonomic network engineering for the self-managing Future Internet
AI	Artificial Intelligence
AMC	Autonomic Management & Control
AN	Access Node
ANS	Autonomic Nervous System

API	Application Programming Interface
ARM	Advanced RISC Machine
BBF	BroadBand Forum
BSS	Business Support System
CF	Control Framework
CP	Control Program
CPU	Central Processing Unit
C-SON	Centralized Self Organizing Network
DE	Decision making Element
DeMe	Rfp_GANA-Level2&3-AccessToProtocolsAndMechanisms
DME	Decision making Element
E2E	End to End
EC	European Commission
EMS	Element Management System
FlowDesc	Flow Description
F-MBTS	Federation MBTS
FMM	Rfp_FederationMBTS- to-FederationMBTS
FOO	Rfp_ONIX-to-ONIX
FPGA	Field Programmable Gate Array
FuDe	Rfp_FunctionLevelDE-to-FunctionLevelDE
GANA	Generic Autonomic Network Architecture
GCP	Global Control Program
GITAR	Generic extension for Internet-of-Things Architectures
G-MBTS	Gouvernance MBTS
GNU radio	GNU's Not Unix™ radio
GoS	Rfp_OSS_to_Governance-MBTS
GPS	Global Positioning System
IBN	Intent Based network
INT	Core Network and Interoperability Testing
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information eXport
IPv6	Internet Protocol version 6
IRIS	Implementing Radio In Software
KP	Knowledge Plane
KP DE	Knowledge Plane Decision-making Element
LAN	Local Area Network
LQI	Link Quality Indicator
LTE	Long Term Evolution
MAC	Medium Access Control
MBTS	Model Based Translation Service
MCE	Monitor and Configuration Engine
ME	Managed Entity
ME-to-DE	reference point ME to DE
MIB	Management Information Base
MIPS	Microprocessor without Interlocked Pipelined Stages
MO	Managed Object
NBI	NorthBound Interface

NOTE: See Figures 11 and 12.

NDPI	Native Device Programming Interface
NE	Network Element
NeDe	Rfp_NetworkLevelDE-to-NetworkLevelDE
NeI	Rfp_NetworkLevelDE-to-ONIX-System
NeM	Rfp_EMS_OR_NMS-to-NodeMainDE
NeMe	Rfp_NetworkLevelDE-to-NodeMainDE
NF	Network Function
NFV	Network Function Virtualisation
NIC	Network Interface Card
NMS	Network Management System
NoDe	Rfp_NodeMainDE-to-NodeMainDE
NoI	Rfp_NodeMainDE-to-ONIX-System

NTP	Network Time Protocol
ONAP	Open Networking Automation Platform
ONIX	Overlay Network for Information eXchange
OODA	Observe, Orient, Decide and Act Loop
OR	logical OR symbol
ORCA	Orchestration and Reconfiguration Control Architecture
OS	Operating System
OsDE	reference point OSS knowledge plane Decision-making Element
OsDe	Rfp_OSS-to-Network-Level-Des
OSI	Open Systems Interconnection
OsI	Rfp_OSS-to-ONIX-System (Network Governance Reference Point: OSS/BSS to ONIX) (Knowledge Plane)
OSi	Rfp_OSS-to-ONIX-System (Network Governance Reference Point: OSS/BSS to ONIX) (Knowledge Plane)
OSS	Operation Support System
PC	Personal Computer
PER	Packet Error Rate
PHY	Physical
PoC	Proof of Concept
PON	Passive Optical Network
PRR	Packet Received Rate
PTP	Precision Time Protocol
QoS	Quality of Service
RAN	Radio Access Network
RAS	Reconfigurable Antenna Systems
RAT	Radio Access Technology
RF	Radio Frequency
RISC	Reduced Instruction Set Computing
RRH	Remote Radio Head
RSSI	Received Signal Strength (power) Indication
SDN	Software Defined Networks
SDR	Software Defined Radio
SNMP	Simple Network Management Protocol
SON	Self Organizing Networks
TAISC	Time Annotated Instruction Set Computer
TB	Technical Body
TC	Technical Committee
TCP/IP	Transfer Control Protocol/Internet Protocol
TDMA	Time Division Multiple Access
UPI	Unified Programming Interface
UPI_G	Unified Programming Interface Global
UPI_HC	Unified Programming Interface Hierarchical Control
UPI_M	Unified Programming Interface Management
UPI_N	Unified Programming Interface Network
UPI_R	Unified Programming Interface Radio
USRP	Universal Software Radio Peripheral
VoIP	Voice over IP
WARP	Wireless open-Access Research Platform
WG	Working Group
WiFi™	IEEE 802.11™ family of standards
WiSHFUL	Wireless Software and Hardware platforms for Flexible and Unified radio and network controL
WLAN	Wireless Local Area Network
WMP	Wireless MAC Processor
WSN	Wireless Sensor Network
xDSL	any Digital Subscriber Line
XFSM	eXtended Finite State Machines
xPON	any PON
ZRE	ZeroMQ Realtime Exchange protocol

4 Principles for Autonomic Networking and Enablers

4.1 Overview on Autonomics Principles and Enablers, and introduction to the emerging concept of "Network compartmentation"

Autonomic networking paradigm aims at creating self-managing networks to overcome the rapidly growing complexity of current networks and future networks. The complexity aspect of particular concern is management and control of the networks and services they are required to deliver to various service consumers. Management complexity can be characterized by factors such as huge number of devices, services to be provisioned and assured, and configuration parameters of network resources that need to be configured and dynamically optimized to cope with various workloads and challenges the networks encounter daily during their operations, e.g. manifestations of faults/errors/failures/security threats and performance degradations on various network resources. The autonomic networking paradigm is the enabler for self-driving and self-aware networks and services.

Autonomic networking mimics biological autonomic systems, especially those complex life forms that are provided with an Autonomic Nervous System (ANS) that is not consciously controlled. Analogously to biological systems, current networks require a conscious control that is mimicked by a centralized network control where a central entity (the brain), receives information from the peripheral elements, knows the status of the whole system, takes decisions and finally applies actions by sending commands to peripheral actuators (muscles). Biological and networking components share such general principles. However, in many applications in wireless networks, the timing for decisions is not compatible with latencies due to the loop from the peripheral sensor to the central intelligence and back to peripheral actuators. In such a case, as discussed in clause 5.5, forms of control by delegation has to be taken into account. The ETSI GANA standard takes into consideration this issue.

Autonomic systems require specific capabilities that appear to be in common with current trends in networks, especially with wireless networks. These capabilities (functions) include:

- Autognostic capabilities (self-discovery, awareness, and analysis).
- Control capabilities on network elements and interfaces.
- Capabilities to define and verify performance and constraints.
- Capabilities to identify attacks and run defending actions.

The WiSHFUL project does not deliberately aim at creating autonomic networks because it is focused on radio and network control for experimentation in wireless networks. However, it appears that WiSHFUL naturally fulfils most of the principles indicated above and provides key enablers for autonomic networks. The ETSI GANA architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management of Networks and Services (ETSI TS 103 195-2 [i.9]) is purposely designed for autonomic networks and is fully specified in the ETSI standard ETSI TS 103 195-2 [i.9] (while a brief introduction to the GANA model is also provided later in the present document); it defines high-level requirements and architectural components for self-management networks. Conversely, WiSHFUL architecture provides the low-level requirements for wireless autonomic networks, which also map well to the GANA framework; in fact, the project defines programmability models and control models for radio and network components.

GANA Autonomic Management and Control (AMC) software modules called Decision-making-Elements (DEs) are meant to be designed in such a way that they employ such models in driving autonomics in a Network Element/Function (NE/NF) and in the outer realm (the management and control systems realm) overlay. For more details on this subject clause 4.6 of the present document discusses the GANA abstractions levels for autonomicity (autonomic/self-management functionality) and how they complement each other.

NOTE 1: The present document aims to illustrate to autonomics implementers how to use WiSHFUL components and components from the ORCA project as well, to implement the GANA framework's multi-layer autonomics in order to realize autonomic management and control of heterogeneous wireless access technologies using cognitive algorithms.

NOTE 2: WiSHFUL provides platforms, tools and architectural elements for wireless experimentation. WiSHFUL does not take into account security, and the experimental environment is considered trusted. No detection of misbehaving nodes is implemented and autonomic capabilities of self-defence are out of scope for the WiSHFUL project.

NOTE 3: WiSHFUL supports most of the key enablers for autonomic networks: network compartmentation, function atomization and composition, closed control loop, context recognition and adaptation, which are discussed in the following clauses.

Network compartmentation: Network compartmentation is an emerging concept to consider in networking, e.g. in relation to another concept called network slicing. However, note that Network compartmentation is not taken into account in WiSHFUL.

WiSHFUL addresses several communication contexts with heterogeneity in devices, technologies, programmability logic.

4.2 Function atomization

The emerging quest for wireless access flexibility and adaptability requires programmable services, devised to customize the wireless access operations according to specific network and application scenarios instead of implementing a specific MAC protocol stack.

These services are composed by simpler primitives: elementary non-programmable functionalities that are natively provided by the system. Primitives deal with low-level atomic actions such as the physical transmission and reception of frames. The hierarchical decomposition of traditional MAC/PHY resource control functionalities is preliminary to introduce programmability at MAC/PHY levels. A well-defined functional decomposition permits to recompose functions programmatically. Such programs permit to shift from *configurability* obtained by tuning parameters to *programmability* by defining new node behaviours through composed procedures. Atomic decomposition of functions allows maximal re-composition freedom but, it conversely introduces complexity in composing smaller functional blocks.

As discussed later in clause 5, WiSHFUL supports several programmable platforms. Most of these platforms pre-existed WiSHFUL, using the functional decomposition described above. However, WiSHFUL platforms are based on different technologies; the decomposition in atomic primitives and the functional composition in services has not a unique form but it depends on the underlying technology, being customized for WSN, WLAN or LTE.

4.3 Function composition

Function composition permits to build flexible, dynamic and autonomic networks. Functional composition requires a language for linking the atomic functional elements as building blocks to link together. Function composition permits to define the behavioural logic for wireless nodes both singularly and in groups. Function composition requires also well-defined application program interfaces for calling the composed services by the radio and network programs.

4.4 Closed control loop (s)

Autonomic networks require a closed control loop (s) to maintain the properties of the controlled network/node within desired bounds by constantly monitoring target parameters. This general concept appears in WiSHFUL both in direct and indirect forms, respectively with control by commands and control by delegation, depending on the status of communication capabilities and the desired reaction time.

The two control principles are also defined and enabled by the ETSI GANA Model's definition of abstractions levels for implementing the concepts of "fast control-loops" in Network Elements/Functions (NEs/NFs) and "slow control-loops" in the realm of management and control systems of a network, as well as the need for "slow control-loops" to exercise policy control of the lower level "fast control-loops" in NEs/NFs.

In the first case, the central controller directly controls the network elements by instructing them the actual *actions* to be performed. In the second case, the central controller provides the *control logic* to controllers that run locally on the nodes. The most appropriate approach depends on the scenario, as both have pros and cons.

A single global control takes decisions on the basis of global information, however the latency gained in the path between the controller and the controlled element may capture an obsoleted view of the network. Conversely, having local controllers provides more timely control with a partial and fragmented vision. Clause 5 illustrates central controllers and local controllers and their interworking and mappings to the GANA model concepts.

NOTE: Clause 4.6 describes the concepts of "slow control-loops" and "fast control-loops" for implementing autonomics and how they complement each other.

4.5 Context recognition and adaptation

The Autonomic Behaviour (AB) required by complex and dense networks requires first context recognition, then context adaptation. Advanced sensing capabilities of wireless nodes, both singularly and in group are required, then a kind of 'network intelligence' recognizes the context (using legacy classification techniques or more advanced machine learning algorithms), then applies the optimal configuration settings or, even better, the optimal protocol logic.

The aim to design autonomic networks demonstrates, by itself, that the one-size-fits-all monolithic solution cannot adapt to any possible operational context, because not all possible scenarios can be foreseen. Tuneable configuration parameters do not provide the required level of flexibility, because instead of spanning the whole fan of behavioural logic, they only adjust operating conditions of a pre-defined protocol logic.

WiSHFUL provides context recognition and adaptation by the means of its intelligent control architecture. WiSHFUL architecture coordinates and control wireless nodes at the radio and network levels (mappings of these levels to the ETSI GANA levels for autonomics are defined later in the present document). It contains a dedicated framework for providing intelligence to the wireless network that includes a Data Collection Component, Intelligence Composition Component, Action Component, and a dedicated user interface.

4.6 Introduction to the GANA Reference Model for Autonomic Networking, Cognitive Networking and Self-Management

4.6.1 Overview

ETSI TS 103 195-2 [i.9] defines the concept of Autonomic Manager element (called a "Decision-making-Element" (DE) in the GANA terminology) as a functional entity that drives a control-loop meant to configure and adapt (i.e. regulate) the behaviour or state of a Managed Entity (i.e. a resource) - usually multiple Managed Entities (MEs).

The ETSI GANA Standardized Framework for AMC (ETSI TS 103 195-2 [i.9]) defines an Intelligent Management and Control Functional Block called GANA KP that is an integral part of AMC Systems that provides for the space to implement complex network analytics functions performed by interworking Modularized and specialized DEs.

The KP DEs run as software in the Knowledge Plane and drive self-* operations such as self-adaptation, self-optimization, self-monitoring objectives for the network and services by programmatically (re)-configuring Managed Entities (MEs) in the network infrastructure through various means possible: e.g. through the NorthBound Interfaces available at the OSS (Operations Support System), Service Orchestrator, Domain Orchestrator, SDN controller, EMS/NMS, NFV Orchestrator, etc.

The GANA KP consists of multiple modularized DEs. In contrast to non-modularized management systems, each DE is expected to be a module (as atomic block) and that it should address a very specific "management domain (scope of management aspects/problems)" such that it can run as a "micro service".

Examples of autonomic manager elements (i.e. DEs) are: QoS-management-DE, Security-management-DE, Mobility-management-DE, Fault-management-DE, Resilience & Survivability-DE, Service & Application management-DE, Forwarding-management-DE, Routing-management-DE, Monitoring-management-DE, Generalized Control Plane management-DE.

DE components of the GANA KP are "macro" autonomic managers (atomic and modular) that drive logically centralized network-wide with slow control loops that operate in "slower timescale" than similar control-loops introduced to run in Network Elements (NEs) and operating as "fast control-loops". Macro autonomic managers (GANA KP DEs) should be complemented by "micro" Autonomic Manager components (DEs injected into NEs) that can be introduced in the Network Elements (physical or virtualized) for driving local intelligence within individual network elements to realize "fast control-loops" in network elements. Macro autonomic managers (GANA KP DEs) policy control the "micro" autonomic managers (GANA DEs in NEs - i.e. the so-called GANA Level-2 and Level-3 in the ETSI TS 103 195-2 [i.9]).

ETSI TC INT AFI WG's work on E2E autonomic networking involves introducing self-manageability (autonomics) properties (e.g. self-configuration, self-diagnosis, self-repair, self-healing, self-protection, self-awareness, etc.) within network nodes/functions themselves and also enabling distributed "in-network" self-management within the data plane network architectures (and their embedment of "thin control planes").

This low level intelligence (autonomics) achievable by so-called "GANA DEs" that should be instantiated to drive fast control-loops within network nodes/elements and to drive horizontal self-adaptive collaborative "in-network" behaviour involving the collaboration of certain autonomic nodes is also called "Micro level" autonomics ("fast control loops").

The low level autonomics should be complemented and policy-controlled (governed) by higher level autonomics ("slow control loops") (at "Macro level") achievable and driven by higher level "GANA DEs" responsible for network-wide and logically centralized autonomic management and control of networks and services. At "Macro level", the autonomics paradigm (control loops) is introduced outside of network elements, in the outer, logically centralized, management and control planes architectures of a particular target network.

This "realm" for implementing the much more complex, cognitive and analytics algorithms (including Artificial Intelligence (AI) Algorithms) for autonomics that operate on network-wide views is called the GANA Knowledge Plane (GANA KP). The three key Functional Blocks of the GANA KP are summarized below:

- **GANA Network-Level DEs:** *Decision-making-Elements (DEs)* whose scope of input is network wide in implementing "slower control-loops" that perform policy control of lower level GANA DEs (for fast control-loops) instantiated in network nodes/elements. The Network Level DE are meant to be designed to operate the outer closed control loops on the basis of network wide views or state as input to the DEs' algorithms and logics for autonomic management and control (the "Macro-Level" autonomics). The Network-Level-DEs (Knowledge Plane DEs) can be designed to run as a "micro service".

- **ONIX (Overlay Network for Information eXchange)** is a distributed scalable overlay system of federated information servers). The ONIX is useful for enabling auto-discovery of information/resources of an autonomic network via "publish/subscribe/query and find" mechanisms. DEs can make use of ONIX to discover information/context and entities (e.g. other DEs) in the network to enhance their decision making capability. The ONIX itself does not have network management and control decision logic (as DEs are the ones that exhibit decision logic for Autonomic Management & Control (AMC)).

- **MBTS (Model-Based Translation Service)** which is an intermediation layer between the GANA KP DEs and the NEs ((Network Elements) - 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. KP DEs can be programmed to communicate commands to NEs and process NE responses in a language that is agnostic to vendor specific management protocols and technology specific management protocols that can be used to manage NEs and also policy-control their embedded "micro-level" autonomics. The MBTS translates DE commands and NE responses to the appropriate data model and communication methods understood on either side. The value the MBTS brings to network programmability is that it enables KP DEs designers to design DEs to talk a language that is agnostic to vendor specific management protocols, technology specific management protocols, and/or vendor specific data-models that can be used to manage and control NEs.

The "GANA" reference model combines perspectives on GANA DE ("Micro-Level" autonomics (defined by the so-called GANA levels-1 to Level-3 illustrated in Figure 1)) and the interworking GANA KP DE (with "Macro-Level" autonomics (realized by the GANA Knowledge Plane)) as well as the responsible Functional Blocks and Reference Points that enable developers to implement autonomics software, with all perspectives combined together so as to capture the holistic picture of autonomic networking, cognitive networking and self-management design and operational principles.

This ETSI GANA Framework is illustrated in Figure 1.

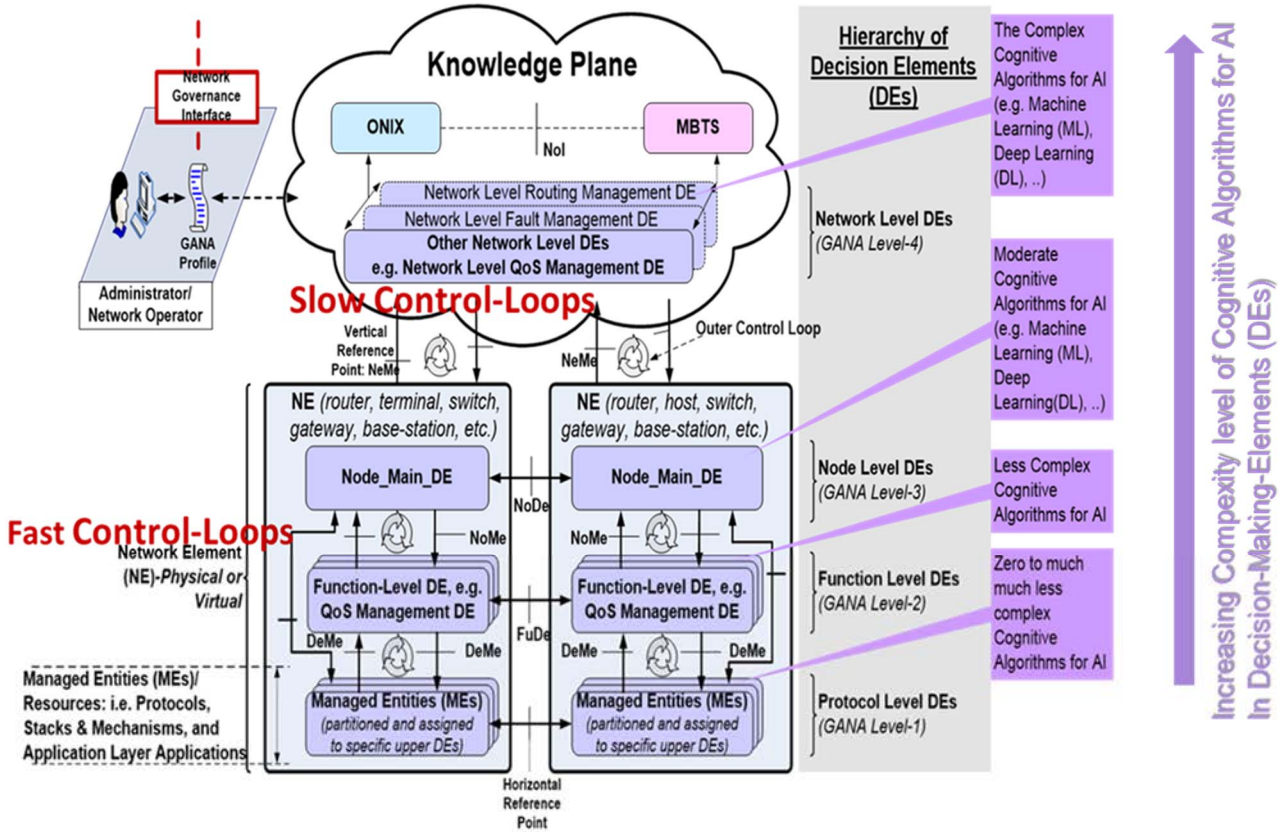


Figure 1: Snapshot of the GANA Reference Model and Autonomics Cognitive Algorithms for Artificial Intelligence (AI)

The following technical white papers provide a lot of useful insights regarding the subject of how to interwork fast control-loops and slow control-loops.

NOTE: Clause 7 contains the mappings of the concepts of the WiSHFUL architecture and ORCA concepts to the ETSI GANA Model in order to illustrate how they are used to implement the GANA Framework.

4.6.2 Examples of Autonomic Management & Control (AMC) domains

The list below shows examples of Autonomic Management & Control (AMC) Aspects (involving knowledge on Implementing DE Autonomics (Control-Loops)) for which AMC Domain Experts should be engaged in Design and Implementations of Cognitive DEs (for more details refer to the White Paper 16 [i.8] from which the text on examples of AMC domains has been extracted):

- **Autonomic QoS-management & control domain:** implies the need to Design/Implement Forwarding-Management_DE.
- **Autonomic Security-management & control domain:** implies the need to Design/Implement Security-Management_DE.
- **Autonomic Mobility-management & control domain:** implies the need to Design/Implement Mobility-Management_DE.
- **Autonomic Fault-management domain:** implies the need to Design/Implement Fault-Management_DE.
- **Autonomic Resilience and Survivability management & control domain:** implies the need to Design/Implement Resilience & Survivability Management_DE.
- **Autonomic Service & Application management domain:** implies the need to Design/Implement Service & Application Management_DE.
- **Autonomic Forwarding-management & control domain:** implies the need to Design/Implement Forwarding-Management_DE.

- **Autonomic Routing-management & control domain:** implies the need to Design/Implement Routing-Management_DE.
- **Autonomic Monitoring-management domain:** implies the need to Design/Implement Monitoring-Management_DE.
- **Autonomic Generalized Control Plane management & control domain:** implies the need to Design/Implement Generalized Control Plane Management_DE.

NOTE: The various types of Decision Elements (DEs) listed above are defined in ETSI TS 103 195-2 [i.9], and their associated mappings to their types of Managed Entities (MEs) - i.e. resources and configurable parameters that should be under the responsibility of the specific DE, are also defined in ETSI TS 103 195-2 [i.9] and in concrete GANA instantiations onto a particular target implementation oriented network architecture and its management and control architecture (e.g. Broadband Forum (BBF) architectures (ETSI TR 103 473 [i.10]), 3GPP Backhaul and Core Network (ETSI TR 103 404 [i.11]).

5 WiSHFUL Architecture

5.1 Overview

5.1.1 General overview of the WiSHFUL Concepts

This clause provides a background on WiSHFUL architecture; this is beneficial for the mapping between the WiSHFUL architecture and GANA principles. WiSHFUL permits to **abstract the internals of the nodes with a unified configuration interface** able to work on completely different hardware and software platforms.

The WiSHFUL project is devised to wireless experimentation. But the concepts introduced by WiSHFUL can be applied in general. Autonomous and intelligent systems are generally re-configurable systems and this opens special issues in case of wireless systems, whose co-existing in the same radio spectrum demands harmonization, control, orchestration and coordinated adaptations of multiple parameters in different protocol layers and in multiple network devices, with cross-layer and cross-node mechanisms.

NOTE 1: As illustrated later, the **unified configuration interface** can be viewed as a way to implement the GANA Model's **Rfp_GANA-Level2&3-AccessToProtocolsAndMechanisms Reference Point** within a GANA Node.

The WiSHFUL architecture is composed by the following main components:

- **Unified Program Interfaces (UPIs)** to implementers for easily prototyping novel and adaptable wireless solutions on different radio platforms;

NOTE 2: As illustrated later, UPIs implement Rfp_GANA-Level2&3-AccessToProtocolsAndMechanisms Reference Point within a GANA Node.

- **Control Programs (CPs)** that contain the logic to program devices singularly and in groups;

NOTE 3: As illustrated later, CPs within a Network Node implement GANA-Levels 2 and 3 DEs.

- a **Control Framework (CF)** for supporting dynamic on-the-fly reconfigurations of the network nodes according to time-varying estimates of the network operating conditions.

NOTE 4: As illustrated later, the GANA Knowledge Plane provides the space to implement the CF.

The unified interfaces include UPI_R, UPI_N, UPI_G, UPI_HC, UPI_M.

The UPI_R permits to configure the node behaviour at the lower MAC and PHY layers, the configuration of the transceiver (transmission formats, spectrum, antennas, etc.) and the configuration of the time-critical access rules for utilizing the wireless resources.

NOTE 5: As illustrated later, the (re)-configurable parameters and node behaviour at the lower MAC and PHY layers are associated with what are called Managed Entities (MEs) in GANA that are assigned to specific GANA DEs.

The WiSHFUL control framework permits to configure the single radio and provide dynamic radio adaptation and to work on a global control program that can in a coordinated way on groups of nodes.

NOTE 6: As illustrated later, the "global program" can be realized collectively by the multiple DEs of the GANA Knowledge Plane (KP), which undergo coordination and synchronization in their actions as described in ETSI White Paper No.16 [i.8], ETSI TS 103 195-2 [i.9], and also in the White Paper No.4 [i.33] of the ETSI PoC.

The WiSHFUL architecture is shown in Figure 2. It interacts with wireless nodes based on heterogeneous radio platforms.

The WiSHFUL architecture has one global **Monitoring and Configuration Engine (MCE)** that orchestrates several remote MCEs residing on each wireless node of the testbed. The ***global MCE*** provides monitor and configuration services that can be used by the implementer to write a *Global Control Program (GCP)*, controlling the behaviour of the solution under test by means of the **UPI_G** interface.

On the other hand, *local control programs* running on ***local MCEs*** control single devices by means of the **UPI_R** and **UPI_N** interfaces, respectively for radio and network control. The same UPI_R and UPI_N functions are exposed on the heterogeneous platforms by means of adaptation modules, hence the need adaptation modules perform some translation service between the two sides of a UPI.

WiSHFUL control framework permits to control programmable wireless nodes by means of unified interfaces. The framework allows orchestrating the utilization of both the UPI_R and UPI_N interfaces at a global and local level, thus supporting dynamic adaptations of the wireless nodes according to the aggregation of radio parameters monitored by different nodes and estimates of the network state.

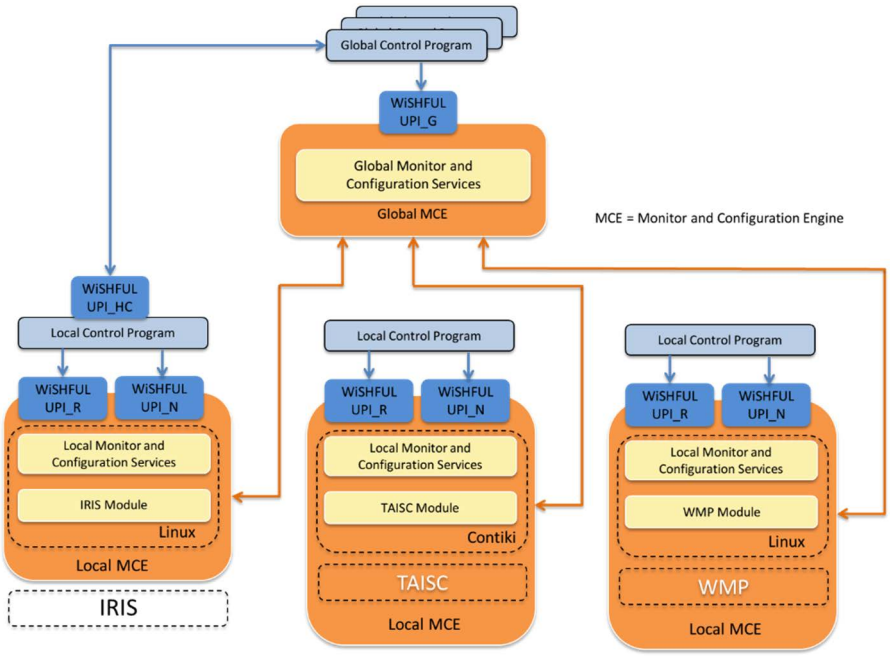


Figure 2: WiSHFUL architecture, UPIs and supported platforms

A hardware system and relevant software modules exposing a configuration UPI interface and abstract programming model is referred in what follows as platform. This definition generalizes the concept of radio platform to any hardware system, which does not necessarily include a radio transceiver (such as intelligent antennas or measurement sensors) and can be added to wireless nodes for providing new capabilities.

According to this vision, a wireless node can be equipped with multiple platforms, including at least one radio platform providing communication capabilities; all the platforms are orchestrated as a whole by the WiSHFUL control programs running on the wireless node.

In Figure 3 it is reported an exemplary wireless node (e.g. a multi-technology gateway) which integrates heterogeneous hardware technologies, such as ZigBee, WiFi and a configurable antenna system, and the relevant software architectures. Thanks to the adaptation modules available for each platform, the node exposes to WiSHFUL the aggregated capabilities in a list of available UPI functions.

The functions abstract the specific node architecture and provide the implementer the possibility to communicate with ZigBee and WiFi nodes and to steer the antenna beam in a desired direction. The implementer exploits the complete list of supported functions for writing the desired control program.

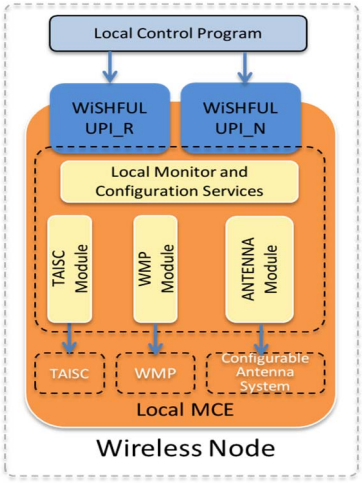


Figure 3: Example of wireless node supporting three different platforms

Further details about the WiSHFUL framework can be found in WiSHFUL deliverables [i.3], [i.4], [i.5] and [i.6].

5.1.2 How Control Programs in the WiSHFUL Architecture are the means to realize (implement) specific GANA Decision Elements (DEs)

The GANA Standard (ETSI TS 103 195-2 [i.9]) defines and standardizes various Autonomic Functions (Decision-making- Elements (DEs)) that can be instantiated to operate at NE/NF level and/or within the GANA Knowledge Plane (KP). The process called GANA instantiations onto an implementation-oriented network architecture and its associated management and control architecture establishes the kinds of GANA DEs that should be instantiated to operate in an NE/NF and/or in the GANA Knowledge Plane.

Various types of Decision Elements (DEs) are defined in ETSI TS 103 195-2 [i.9], and also their associated mappings to their types of Managed Entities (MEs) - i.e. resources and configurable parameters that should be under the responsibility of the specific DE.

Specific DEs and their mappings to specific MEs are then further detailed in concrete GANA instantiations onto a particular target implementation oriented network architecture and its management and control architecture (e.g. Broadband Forum (BBF) architectures (ETSI TR 103 473 [i.10]), 3GPP Backhaul and Core Network (ETSI TR 103 404 [i.11])).

In order to instantiate and implement specific GANA DEs as "control programs" using the WiSHFUL framework, local program implementers should use the guiding Table 3 specified in ETSI TS 103 195-2 [i.9] to determine the DEs that can be instantiated and implemented to operate at a specific GANA level and the kinds of Managed Entities (MEs) and their configurable parameters that are to be autonomically managed and controlled by a specific DE.

Clause 7.5 of the present document provides more insights on this subject of DE-to-MEs Mappings, and how to use some software code of local programs (local controllers) and global programs (global controllers) already implemented in WiSHFUL Framework to implement the standardized GANA DEs required to operate at specific GANA Levels.

5.2 WiSHFUL platforms and abstractions

The WiSHFUL framework allows the control of heterogeneous platforms, i.e. heterogeneous classes of devices (micro-controller devices, general-purposes devices and software defined radio) and radio technologies by means of unified interfaces and control models.

Note that a given wireless technology can be supported by different radio platforms. i.e. by different hardware and drivers. For example, the WiSHFUL UPI are available for IEEE 802.11 [i.12] nodes based on commercial interfaces (namely, the Atheros™ cards), commercial interfaces with customized non-standard firmware (namely, the Wireless MAC Processor), and software defined radios.

NOTE: The present document is not tied to Atheros™ card.

While some UPI_R functionalities are technology-agnostic, some others refer to specific technologies and therefore it is important to know which technologies are supported by a given platform for accessing these functionalities. Consequently, the radio platforms are categorized as WiFi™, LTE, and Lowpan (IEEE 802.15.4 [i.13]) platforms, according to the wireless technologies that can be supported.

WiSHFUL also abstracts the radio platform programming model, in terms of generic execution engine and radio programs. According to this model, each radio platform offers the possibility to load several MAC/PHY programs, already available for implementers in the WiSHFUL repository, or to define novel wireless protocols and radio behaviours by means of high-level programming languages.

Table 1 provides a summary of the platforms supported in WiSHFUL.

Table 1: WiSHFUL supported platforms

Module name	Description
WMP™	Wireless MAC Processor (WMP™) follows a programming model that decouples the Medium Access Control protocol logic (described in an abstract form via eXtended Finite State Machines - XFSM) from the wireless device design, implementing the radio primitives as well as an XFSM execution engine called "Wireless MAC processor". The core of the architecture is an execution Engine capable of running programs defined as eXtended Finite State Machines (XFSMs). The WMP is implemented on a **Broadcom™ AirForce54G wireless card™** and (partially) on a Software Defined Radio SDR platform (namely, the Wireless Open-Access Research Platform™ (WARP) board).
TAISC™	TAISC™ (Time-Annotated Instruction Set Computer) consist of a cross-platform MAC protocol compiler and execution engine. The cross-compilation approach allows developers to design MAC protocols once, and then compile them for reuse on different radio platforms. This approach has been successfully implemented for IEEE 802.15.4 [i.13] MAC protocols on embedded wireless nodes (**RM090™ and Zolertia™ RE-Mote**) and on a Xilinx™ Zynq-based SDR platform™ [i.24] and [i.25].
IRIS™	IRIS™ is a software defined radio framework that allows users to design and construct radios from the composition of user defined signal processing blocks. The processing blocks of IRIS are written in C++ and run on the general purpose processor of a computer with a Linux™ based operating system. This computer is then interfaced to a universal software radio peripheral™ (**USRP**) frontend device, which handles the radio frequency aspects of the radio, which are limited to basic up or down conversion and minor filtering in the typical case.
Atheros™ platform	Atheros™-based IEEE 802.11 [i.12] platform is a Commercial off-the-shelf IEEE 802.11 [i.12] compliant chip on a Linux™ platform. Following the Software-Defined Networking (SDN) paradigm the control plane can be separated from the data plane and provide an API to allow local or global control programs to configure the channel access function. In particular, this allows configuring the airtime sharing protocol access like define the number and size of time slots in which the transmission is enabled. Moreover, for each time slot a medium access policy can be assigned which allows restricting the medium access for particular stations (identified by their MAC address) and traffic identification (e.g. VoIP or video). The latter can be used to program flow-level medium access.
GNU radio	GNU Radio [i.22] is a free software development toolkit that provides signal processing blocks to implement software-defined radios and signal-processing systems. It can be used with external compatible Radio Frequency (RF) hardware in order to deploy SDR transceiver; moreover GNU Radio allows deploying innovative solutions in simulation-like environment.
RAS antenna	The Reconfigurable Antenna Systems (RAS) has been developed in the Open Call 1 extension of the WiSHFUL project. The antenna is capable of steering the radiation pattern dynamically on demand from typical omnidirectional to directional shape in the azimuth plane. RAS antenna is fully supported from WiSHFUL that provides UPI function to set the antenna direction.

5.3 Adaptation Modules

In the initial WiSHFUL architecture, it was initially assumed that each wireless node would have been built on top of a single radio platform. For this reason, adaptation modules were designed for mapping the generalized UPI interface into platform-specific function calls, thus hiding the implementation details of each platform.

Wireless nodes can integrate heterogeneous platforms by:

i) **exposing different hardware capabilities and software functions**; and

ii) **supporting standard and/or non-standard radio technologies**.

To cope with this generalized view, the WiSHFUL architecture exploits **multiple adaptation modules** in the same wireless node, thus decoupling the wireless node capabilities from a specific radio platform.

The UPI interface exposed by a wireless node is given by the collection of functions supported by the adaptation modules, which have been installed for driving the available platforms. Moreover, the concept of adaptation has been further generalized for addressing the purely software architectures **implementing the higher layers of the protocol stack,** such as the operating systems or the traffic source generators.

In other words, adaptation modules provide a set of UPI functions available in a given wireless node because of the installed platforms, operating system and software tools. The complete list of loaded adaptation modules and capabilities for each node are reported to the control program by the Monitoring and Configuration Engine (MCE).

Indeed, only the UPI functions presented in the loaded modules can be called by the control program. All the local MCEs and adaptation modules are implemented in Python [i.17] except for Contiki™ sensor nodes [i.21] where, in addition to the Python implementation residing on a host Linux™ Personal Computer (PC), also a native C software module exists that is used as an interface to the GITAR (Generic extension for Internet-of-Things Architectures) reconfiguration services on the node [i.23].

GITAR middleware offers a generic solution to integrate a vertical control plane within the protocol stack of constrained sensor devices.

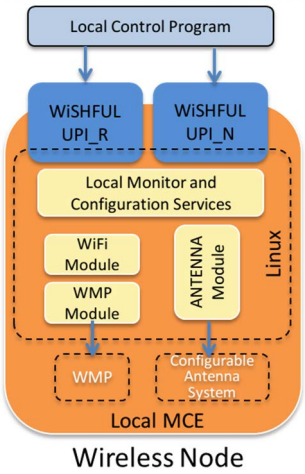


Figure 4: WiSHFUL adaptation modules

Figure 4 shows an example of wireless node with multiple loaded adaptation modules refer both to hardware radio platforms and to protocols.

5.4 Unified Program Interface

5.4.1 Overview on WiSHFUL Unified Program Interfaces (UPIs)

WiSHFUL Unified Program Interfaces are grouped according to their goal: management, network control and radio control. Additionally, UPI functions are categorized by technology-independent functions and those that are technology-specific devoted, respectively, to Wireless Sensor Networks, LTE and WiFi. The same functions can be used both locally and globally. The full definition of UPIs can be found in [i.26].

5.4.2 UPI_M

The UPI_M interface provides management functions at any layer, both local and global. UPI_M are used for deploying, installing and activating software packages. WiSHFUL considered how to use the concept of a software package in the execution environment to deploy such UPIs in the platform. Moreover, it is possible to use the UPI_M interface to deploy the radio program on platforms, setup wireless links, etc. The UPI_M is used to enable the use of a specific execution environment in a node and also for the initialization of nodes before starting an experiment.

NOTE: Turning to the ETSI GANA Model, the GANA Knowledge Plane (KP) Platform.

5.4.3 UPI_N

Unified Programming Interface - Network (UPI_N) is a set of functions that ensures uniform control of the upper MAC and network layer protocol behaviour on heterogeneous devices. The functions forming the interface are generic, their implementation is hardware and platform specific and is provided by the Local Monitoring and Configuration engine. The user is able to manipulate a wide range of network layer functionality like routing, flow control, queue management, priority control and more. UPI_N functions are organized into the following functional groups:

- Address management
- Protocol attribute manipulation
- Traffic control
- Topology detection and routing control

NOTE: The UPI_N functions map to what are called Managed Entities (MEs) in the ETSI GANA Model, and the MEs should be individually assigned to specific GANA DEs responsible for the autonomic orchestration and management and control of their MEs and collaborating with each other to realize a global autonomics objective(s).

The "ME-to-DE" assignments follow the principles of "ME ownership" prescribed in ETSI TS 103 195-2 [i.9] by which an ME is owned by only one (1) DE that autonomically manages and controls it. As such, the types of UPI_R functions (i.e. ME Types) determine which types of GANA DEs can be implemented to use the UPI_N, either as standalone programs or as a "bundled/merged" single program.

From the implementation point of view, UPI_N functions rely on three modules: module_net_linux, module_iperf for the Linux OS; and module_net_contiki for the Contiki OS [i.21].

5.4.4 UPI_R

The UPI_R interface permits monitoring and configuration of radio behaviours of the nodes. Radio control is devised to easily prototype novel wireless solutions, which can include dynamic adaptations of the MAC/PHY of the devices. The solutions can be platform-agnostic, thanks to the abstractions provided by the UPI_R interface, and can work on heterogeneous hardware platforms, including sensors, wireless cards and software-defined-radio.

UPI_R is responsible of tuning the radio operating frequency, selecting the transmission format, activating wireless links towards neighbour nodes, collecting statistics and configuring the medium access logic. The interface acquires information about the platform radio capabilities, because different platforms can support different programmability models and configuration parameters.

Then, according to the available capabilities, the interface functionalities can work on three aspects: configuring the experimentation platform, at both the hardware and radio program levels, monitoring the node and network conditions by accessing all the signals and internal state information of the experimentation platforms, adapting on-the-fly the node behaviour by loading and activating - on the fly - context-specific radio programs.

NOTE: Like in the case of the UPI_N, the UPI_R functions above map to what are called Managed Entities (MEs) in the ETSI GANA Model, and the MEs should be individually assigned to specific GANA DEs responsible for the autonomic orchestration and management and control of their MEs and collaborating with each other to realize a global autonomics objective(s). The "ME-to-DE" assignments follow the principles of "ME ownership" prescribed in ETSI TS 103 195-2 [i.9] by which an ME is owned by only one (1) DE that autonomically manages and controls it. As such, the types of UPI_R functions (i.e. ME Types) determine which types of GANA DEs can be implemented to use the UPI_R, either as standalone programs or as a "bundled/merged" single program.

5.5 WiSHFUL Control Framework

5.5.1 Control Concepts and programmability enablers implemented in the environments that were considered by WiSHFUL

The WiSHFUL control framework facilitates prototyping of novel control solutions in heterogeneous wireless networks according to the WiSHFUL principles:

- Coordinated ***collection*** of information from nodes and ***execution*** of control actions on different protocol layers (***cross-layer***), heterogeneous devices (***cross-technology***) and multiple nodes (***cross-node***).
- Existence of a global and consistent view of the entire network, i.e. knowledge about the state of all devices and their relationships.
- Possibility to implement logically centralized and physically distributed control programs, i.e. placing time-sensitive tasks close to device and off-loading resource greedy tasks to powerful servers.
- Support for multiple levels of control for scalability reasons, i.e. local control programs handle frequent commands and events, while global/hierarchical control programs handle rare events.
- Support for detecting network changes in proactive and reactive control schemes in control programs.
- A high-level API for control of operation of individual wireless devices and groups of devices.
- Location transparency that permits to use same API syntax for execution of commands on local and remote devices.
- Possibility to execute commands on group of nodes/devices.

Control and optimization of operation of wireless network usually involves tuning parameters of network devices being in proximity of each other, i.e. in wireless communication/interference/sensing area. Examples are the radio channel and transmit power assignment to co-located Access Points in WiFi [i.12] networks.

Hence, the control plane requires mechanism to ***discover*** the ***wireless devices*** in the network and their (wireless) relationship. Moreover, this information has to be monitored and updated at run-time. Having a global view of the entire wireless network enables control programs to efficiently manage and control of wireless devices. Changes in the ***network state*** can be detected in two ways, namely proactive and reactive.

In a proactive approach, the network controller is periodically polling the network entities, while in a reactive approach the execution of control program functions is triggered by events generated by the nodes in the network. It should be up to the implementer to define the preferred control strategy.

The WiSHFUL control framework was prototypically implemented. Particular attention was paid to enhance code reusability and support for different programming languages as well as enabling the use of specialized external software libraries.

The main prototype is implemented in Python language [i.17] , which makes it possible to run on multiple different Operating System host types (Linux, OpenWRT, Mac OS and Windows) and allows for rapid prototyping of control programs. An overview of the implementation is presented in Figure 5.

As WiSHFUL project used only standard and common Python libraries [i.17], it proves that it is possible to run and test WiSHFUL implementation on multiple platforms, including x86, "Advanced RISC Machine" (ARM) and "Microprocessor without Interlocked Pipelined Stages" (MIPS). In order to also support constrained devices, a lightweight C version of the agent-side of the framework was also implemented in Contiki [i.21].

In order to support delayed and time-scheduled function execution, the Agent class is equipped with a scheduler (Python Apscheduler[i.17]).

Note that when coordinating multiple nodes by means of time scheduled execution, the nodes in the network require a common notion of a global clock (e.g. obtained through use of Global Positioning System (GPS) or time protocols like Precision Time Protocol (PTP) or Network Time Protocol (NTP)).

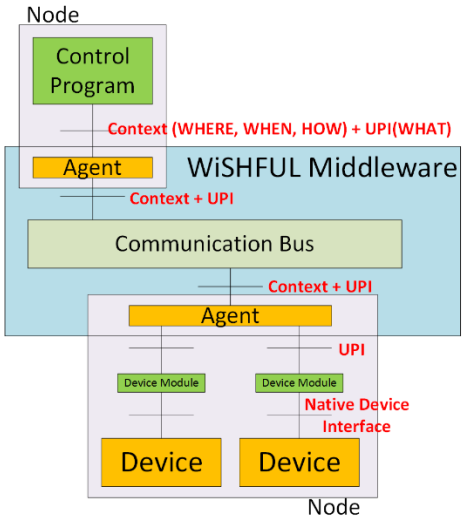


Figure 5: Implementation overview of WiSHFUL control framework

5.5.2 Interaction models

The WiSHFUL control framework supports both ***proactive and reactive approaches.*** In the proactive behaviour, local or global control programs triggers the execution of UPI functions on wireless nodes like polling. The ***reactive approach*** is also supported. It includes triggers that occur when specific conditions are fulfilled, and then registered callback functions are executed.

The WiSHFUL MCE supports two types of UPI calls: synchronous blocking UPI calls where the caller, i.e. the WiSHFUL *control program (local or global)*, is blocked until the callee returns. The second option is an asynchronous non-blocking UPI function call. Here any UPI call returns immediately. The caller has the option to register a callback function so the return value of the UPI call can be received at a later time.

5.5.3 Immediate and delayed commands

Beside the possibility of immediate execution of UPI functions either using a blocking or non-blocking scheme the WiSHFUL MCEs provide the possibility for time-scheduled execution of UPI functions at a particular instant. This is important if nodes need to coordinate their actions in time, e.g. a set of nodes perform a time-aligned switching to a new channel. The possibility for time-scheduled execution of UPI functions is especially important for global control programs and can be used in together with control by delegation, when the connection between the global MCE and local ones is not reliable or is not fast enough to meet strict time requirements of low-level protocols.

5.5.4 Local and remote execution

WiSHFUL provides ***full location transparency***. Any UPI function can be executed either locally by a *local control program* or remotely by a *global control program*. In the latter case, the WiSHFUL global MCE transparently serializes all input and output arguments. Finally, as with the local execution also the execution of remote functions can be time-scheduled. This is especially important if a given UPI function needs to be executed at the same time on a set of wireless nodes.

5.5.5 Synchronization

A wide range of WiSHFUL applications like the centralized control of channel access requires tight time synchronization among wireless nodes for time-sensitive control of devices. The way the wireless nodes are time synchronized is platform and architecture-dependent. Basically, WiSHFUL distinguishes between systems where a backbone network exists. Here in order not to harm the performance of the wireless network the nodes are time synchronized using the backbone (e.g. Ethernet) and in some cases using protocol like "Precision Time Protocol" (PTP). Wireless nodes without a backbone have to rely on other techniques for time synchronization (e.g. GPS).

5.5.6 Packet monitoring and manipulation

WiSHFUL provides a wide range of functionality for packet forgery, sniffing and injection. Control programs can use this to create and inject network packets into the network stack of a node or to receive copies of packets. All WiSHFUL nodes support the sniffing and injection on IP layer (layer 3). Packet manipulation in WiSHFUL is provided with an object-oriented approach and is based on *iptables*, packet marking and setting Type-of-Service value. Data flows can be marked with ***FlowDesc*** then new rules can be installed in *iptables* of the system under test.

5.5.7 Node handling

The WiSHFUL control framework uses ZeroMQ Realtime Exchange Protocol (ZRE) [i.27], for Linux-based nodes that are connected to a dedicated control network. This peer-to-peer protocol provides automatic node discovery. This functionality is useful for defining radio and network programs that control a set of nodes that is unknown or can change over time.

5.5.8 Extensibility of UPI functions

WiSHFUL provides an open and extensible architecture, which can be easily extended by new UPI functions. Any new introduced UPI function can be implemented in a different way for different platform and architecture. Therefore, in WiSHFUL for each platform there is an adaptation module that maps the general UPI call into platform-specific implementations.

Technical details and examples regarding the WiSHFUL control framework are available in [i.3], [i.4], [i.5], [i.6] and [i.25].

5.6 Hierarchical Control Model

The orchestration provided by the WiSHFUL framework uses UPI interfaces for controlling components at a global and local levels as well as using a hybrid paradigm.

The control framework provides basic services for coordinating the UPI_R calls, which include **time synchronization** among the nodes (for relying on a common temporal signal), **blocking or non-blocking** interface calls, **time-scheduled** and **remote execution** of UPI_R functions, **loading** of local control programs on the nodes. These services can be exploited for the definition of the control programs, which work on both radio and network control.

The WiSHFUL architecture supports a ***two-tier control hierarchy:*** one global Monitoring and Configuration Engine (MCE) and several ***local MCEs*** that control single devices by means of the **UPI_R** and **UPI_N** interfaces.

The global MCE orchestrates several remote MCEs residing on wireless nodes. Monitor and configuration services are defined through *Global Control Programs (GCPs).* These two tiers work in a coordinated manner, being orchestrated at the global level. Indeed, global control programs can instantiate local control programs on wireless nodes, performing a sort of *control by delegation,* or can act directly on the wireless nodes in a coordinated manner.

Control by delegation is needed when the reconfiguration decisions or the parameters to be monitored have strict time constraints, which cannot be guaranteed by the control network. In fact, the physical channel used for conveying control messages to/from the global controller can be unreliable and introduce some latency. Since radio performance depends on highly variable network conditions (e.g. channel propagation, fading, interference, access timings, etc.), control by delegation is particularly important for radio control. The architecture also supports hybrid approaches, in which some control operations are managed at the global level, while some others are demanded to wireless nodes. The coordination between global and local control programs is obtained by using the **UPI_HC**, which is the main driver for this hierarchical control.

Figure 2 illustrates how the WiSHFUL radio control works on heterogeneous radio platforms. The global MCE runs remotely and allows implementing node configurations that depend on network-level decisions and can be executed in a time-coordinated fashion among multiple nodes.

Each of the WiSHFUL enabled nodes runs a local MCE that offers the same local services and the same UPI_R functions on different radio platforms. Interactions between components of the WiSHFUL general architecture are reported in Figure 6.

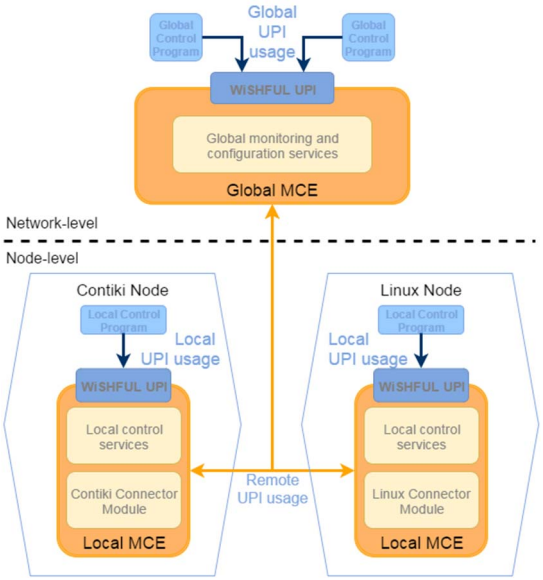
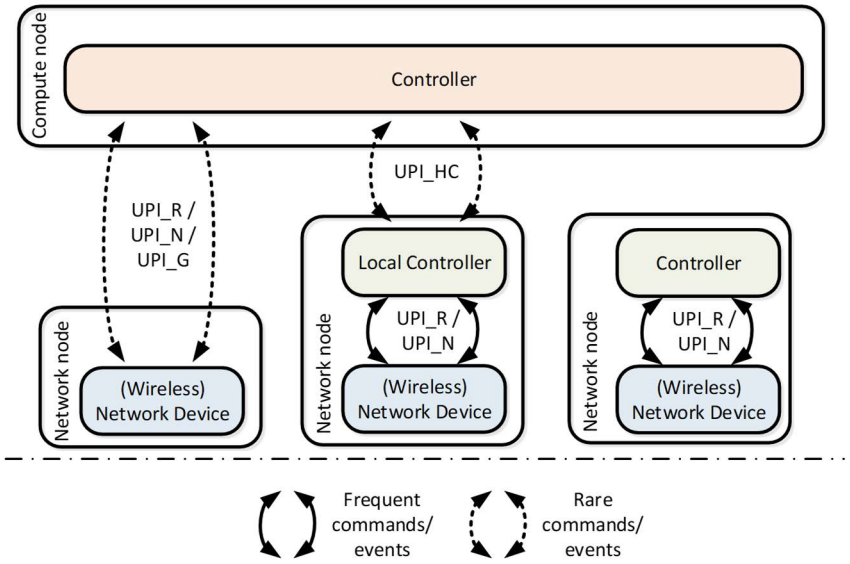


Figure 6: Overview of the components in the general WiSHFUL architecture

For ***coordinated control*** among multiple devices of different nodes, the framework API has to support time synchronized execution of functions across multiple network devices. Examples are the coordinated channel switching of multiple devices due to appearance of an interference source.

While it is natural that the device programming interface is different for each wireless technology, in most cases it also varies across different implementation of the same technology, i.e. wireless devices of different vendors. The unification of the different Native Device Programming Interface (NDPI) is achieved by the introduction of the UPI_R and UPI_N interfaces which allow controlling the devices of a heterogeneous network in a unified way.



NOTE: Global controllers (left) handle rare events and commands while local controllers (right) are able to handle frequent commands and events. In hierarchical control (middle) there are two control loops, i.e. outer and inner.

Figure 7: Levels of control in WiSHFUL control framework

In the general SDN concept the control plane is ***logically centralized*** enabling control programs to have a global view of the entire network. This approach simplifies the development of control programs significantly. However, from a practical point of view a centralized controller would introduce a significant delay in the control plane, which could in turn prevent time sensitive control logic to be implemented.

Moreover, transporting all monitoring data from devices to a central node would create a high load on the control plane. Sometimes ***pre-processing data locally*** at the device is feasible. In this way, the control logic may be partitioned into smaller control programs where parts of them would run on the network nodes and others on the central compute node, i.e. hierarchical control.

Another advantage of such a split is the possibility to reuse control programs. For example, an averaging filter may be implemented once as a control program and used as a local component in the implementation of more complex controllers in the future.

Figure 7 shows the levels of control in the WiSHFUL control framework. Local control programs handle frequent commands and events, while global control programs handle rare events. There is also the possibility for hierarchical control where exchange of events between the global control programs and the local control programs is also rare.

WiSHFUL control framework allows for running multiple control programs communicating with each other and provides them with interfaces for controlling wireless devices in a coordinated way.

5.7 Monitor and configuration engines and services

WiSHFUL implements its hierarchical control model using Monitoring and Configuration Engines (MCEs), as already indicated in clause 5.6. The global MCE orchestrates several remote MCEs residing on each wireless node of the testbed.

The global MCE provides monitor and configuration services that can be used to write a Global Control Program (GCP). This is the logic to control and modify the behaviour of wireless nodes. WiSHFUL MCEs collect measurements (e.g. throughput, air-time usage, PHY errors, etc.), take decisions, and update the configuration of heterogeneous hardware and software platforms.

MCEs apply their decisions of radio configuration on the radios, this can be done using different programmability models that depend on the platform in use (e.g. the Wireless MAC Processor is programmed through xFSMs, Time Annotated Instruction Set Computer (TAISC) is programmed with a time-annotated instruction set, etc. as reported in Table 1).

MCEs have a unified way to configure platforms using dedicated unified program interfaces (UPIs), discussed in clause 5.4. MCEs correspond to decisions elements in the GANA architecture. The logic to take decisions can be composed starting from a WiSHFUL intelligent repository of algorithms for data analysis and manipulation, including machine learning algorithms. This is specifically addressed by the WiSHFUL intelligent framework, which will be discussed in clause 5.9.

5.8 Execution engines, radio and control programs

5.8.1 Overview

The WiSHFUL architecture clearly splits the execution logic (the radio programs), from the execution engines (the actual executor of the behaviour on the programmable platform). This clear decoupling enables maximal exploitation of radio functionalities available in current radio chips, as opposed to today radio drivers that restrict radio functionality.

For example, todays radio drivers for IEEE 802.11 [i.12] do not support TDMA (Time Division Multiple Access) operation, while the hardware perfectly supports it. Furthermore, the clean separation between radio control and protocol logic is in contrast with today's monolithic implementations, which prevent the ability to separately work on the logic for enabling specific protocol features and the definition of these features. In the present document the WMP and TAISC programmable platforms are described as representative examples of heterogeneous programmability models that are based on execution engines.

5.8.2 WMP

The Wireless MAC Processor (WMP) architecture offers the possibility to easy program, load and execute customized MAC protocols, by using a platform-independent, **extended finite state machines** (XFSM) based, high-level programming language. This capability is achieved by developing a firmware which does not implement a specific protocol, but rather a generic protocol executor called MAC Engine.

The MAC programs are specified as **extended finite state machines** (XFSMs), which are built by composing elementary **hardware actions**, in response of specific **hardware events** and **conditions** of the hardware internal registers. The set of events generated by the hardware, the set of actions coded in pre-defined firmware modules and the set of hardware registers whose settings can be tuned and verified, represent the hardware API that cannot be modified by the user.

The MAC program is coded into a transition table and loaded in a memory space deployed on the hardware. Starting from an initial (default) state, the MAC engine fetches the table entry corresponding to the state, and loops until a triggering event associated with that state occurs.

It then evaluates the associated conditions on the configuration registers and triggers the associated action and register status updates (if any). Next it executes the state transition and fetches the new table entry for the destination state. The MAC engine does not need to know to which MAC program a new fetched state belongs to.

Therefore, code switching is achieved by simply moving from the current protocol state to a target state in a different transition table, with a latency of a few Central Processing Unit (CPU) clocks.

5.8.3 TAISC

TAISC (Time-Annotated Instruction Set Computer) aims to simplify the development of new protocols for sensor nodes. It consists of a cross-platform MAC protocol compiler and an execution engine. This design allows to describe MAC protocols in a platform independent language (consisting of a radio platform independent instruction set), followed by a straightforward compilation step, yielding dedicated binary code, optimized for specific radio chips.

The cross-compilation approach allows developers to design MAC protocols once, and then compile them for reuse on different radio platforms. To enable time-critical operation, the TAISC compiler adds exact time annotations to every instruction of the optimized binary code. The execution engine running on the radio platform, will execute the instructions with accurate time control thanks to the provided time annotation.

The overall TAISC workflow to develop and execute a MAC protocol involves the following steps:

1) **device-agnostic MAC protocol creation**.

   First, the MAC protocol designer creates a high-level, platform independent radio program to describe the MAC logic using predefined commands (instructions) in a C-like language, either using high-level C language syntax or using a more intuitive drag- and-drop interface.

   This human readable code consists of a sequence of commands that describe the generic behaviour of the MAC protocol and is largely independent of hardware specifics of the final hardware platform;

2) **device specific compilation**.

   Next, this human-readable sequence is compiled by the TAISC compiler into efficient, device-specific binary byte code that can be executed by the TAISC execution engine running on the radio platform;

3) **protocol dissemination**.

   Afterwards, the byte code is wirelessly transmitted to the target hardware platform and added to the MAC application repository on the local TAISC execution engine;

4) **MAC protocol execution**.

   Finally, the TAISC core executes the byte code.

5.9 Intelligence framework (data collection, intelligence composition, action)

The WiSHFUL Intelligence Framework uses data collected from the network nodes, applies machine learning algorithms and applies configuration actions. The intelligence framework communicates with the control framework by the mean of UPIs. The generic functional view can hence be mapped to the conceptual framework for enabling intelligence shown in Figure 8.

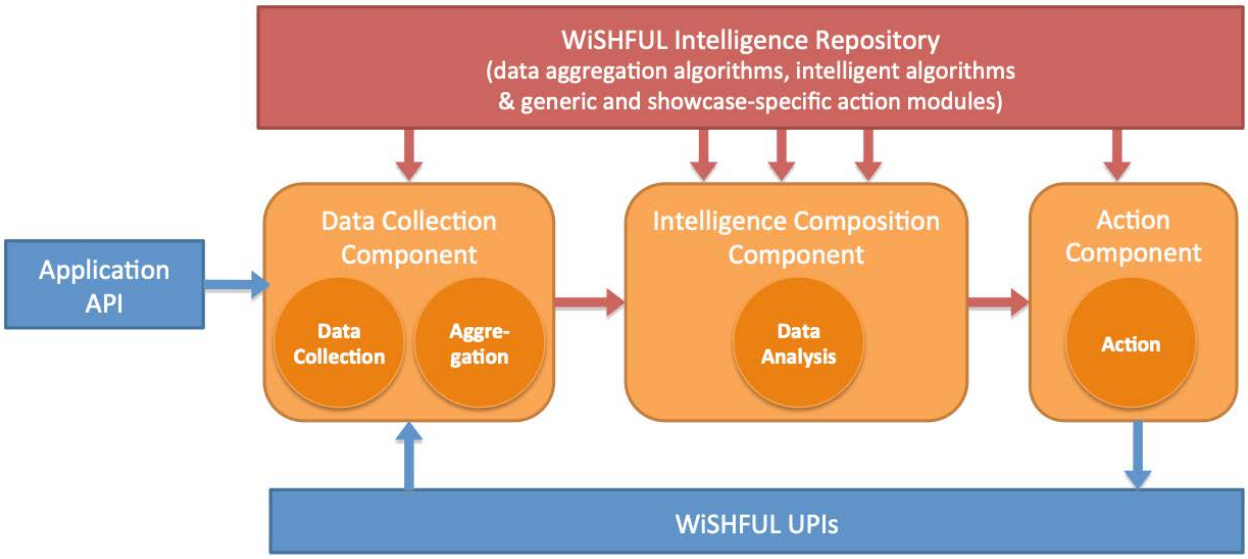


Figure 8: Conceptual framework for enabling intelligence in the WiSHFUL architecture

As the UPIs are unified abstractions that span several wireless technology platforms, the components of the intelligence framework are generic. The Data Collection Component is a generic software module that interacts with the WiSHFUL UPIs to retrieve data about radio and network state (i.e. channel occupancy, link quality indicator (LQI), received signal strength (power) indication (RSSI), Packet Received Rate (PRR), topology, etc.), and with the Application to retrieve information about the application requirements (e.g. max delay, peak throughput, max Packet Error Rate (PER)).

The Data Collection Component also implements aggregation functionality. The Intelligence Composition Module offers support for composing and configuring several algorithms available in the WiSHFUL Intelligence Repository into a self-contained intelligence engine that uses the data provided by the Data Collection Component and triggers network and radio configuration through the Action Component.

The Action Component uses the WiSHFUL UPIs to adjust the configuration of radio and network. The radio and network configuration should be viewed as the output of the intelligence process. Such a configuration can deal with individual parameters (e.g. centre frequency, backoff delay, etc.), radio processing elements (e.g. filter swapping), a waveform (e.g. a modulation and coding scheme) or a protocol (e.g. new MAC scheme).

The framework allows to support the usual Observe, Orient, Decide and Act loop (OODA loop): the data collection component is responsible of gathering data observations and aggregating and filtering the data for extracting the features used in the orient phase; the intelligence composition component is responsible of taking decisions on the basis of the previous observation and orient phase; the action component is responsible of implementing an adaptation decision by reconfiguring the wireless nodes.

6 Impact of Virtualization and Hardware Acceleration Techniques, and Radio Access Network Slicing (RAN Slices), to WiSHFUL Concepts and Principles

The H2020 ORCA project [i.28] extends the WiSHFUL architecture with more advanced software defined radio support. The main focus of ORCA is offer real-time software defined solutions that are reconfigurable and reprogrammable at runtime and that support end-to-end communication between real-life applications in realistic wireless network setting (involving many nodes sharing the same wireless environment).

The ORCA project focuses on 3 key features [i.29], [i.30], as illustrated in Figure 9:

- ***Softwarization (Software'rization)*** is the movement towards using software rather than hardware to perform the processing of network functions. By processing a functionality, which was previously done on dedicated hardware, in software and programmable logic, wireless networks become more reconfigurable. Softwarization is realized by a *programmable framework*, which is a modular, flexible, extensible, and reusable software framework that provides general functionality with a wide set of configurations that can be used for creating specific applications. An example of a programmable framework is the WiSHFUL framework that offers many configuration options for controlling and monitoring of wireless networks. The result of softwarization of radio functionalities is additional flexibility at two different levels:
  - *Parameters level flexibility* refers to the capability to modify reconfigurable parameters within a radio function (e.g. a threshold in a preamble detection block or a modulation scheme in various symbol level processing blocks).
  - *Composition level flexibility* refers to the capability to transfer information in a wireless channel by connecting softwarized radio functions in a chain. Radio functions can be placed on different hardware entities, such as dedicated hardware, programmable logic chips field-programmable gate array (FPGA), embedded processors, general purpose computers, servers in the cloud, etc. From cloud to FPGA, each of the listed options has its own advantage; in general on the cloud end, there is more flexibility and ease of configuration, whereas at the FPGA side there is a higher processing speed. Softwarized radio functions can be further replaced, added, or removed according to real-time requirements. Radio functions flexibility at composition level allows the instantiation of different RATs on the same hardware infrastructure.
- ***Virtualization*** is the partitioning or aggregation of real radio resources (e.g. spectrum, time, space/beam) in order to create isolated radio slices, each slice supporting a specific service and tailoring virtualized radio functions to a specific wireless context of the service. Virtual radio resources can be flexibly sized depending on context, to real radio resources that are fixed in size. For example multiple transceiver chains of virtualized radio functions can be mapped to isolated radio channels within the available spectrum. Channels can have equal or different channels widths according to traffic demands in each radio slice. Alternatively, multiple real radio resources can be combined as if they were one large virtual resource (e.g. multiple carriers within the same RAT or multiple RATs can aggregated to increase the wireless capacity). The entity that is responsible for virtualization is called the *hypervisor*. Virtualization (together with softwarization) enables sharing of the same infrastructure for multiple concurrent RATs by multiple network providers.
- ***Orchestration*** is the placement of functionality in the different hardware entities, and the management and control of softwarized and/or virtualized radio resources. The entity that is responsible for orchestration called the *orchestrator*. An example of orchestrator is the WiSHFUL control program.

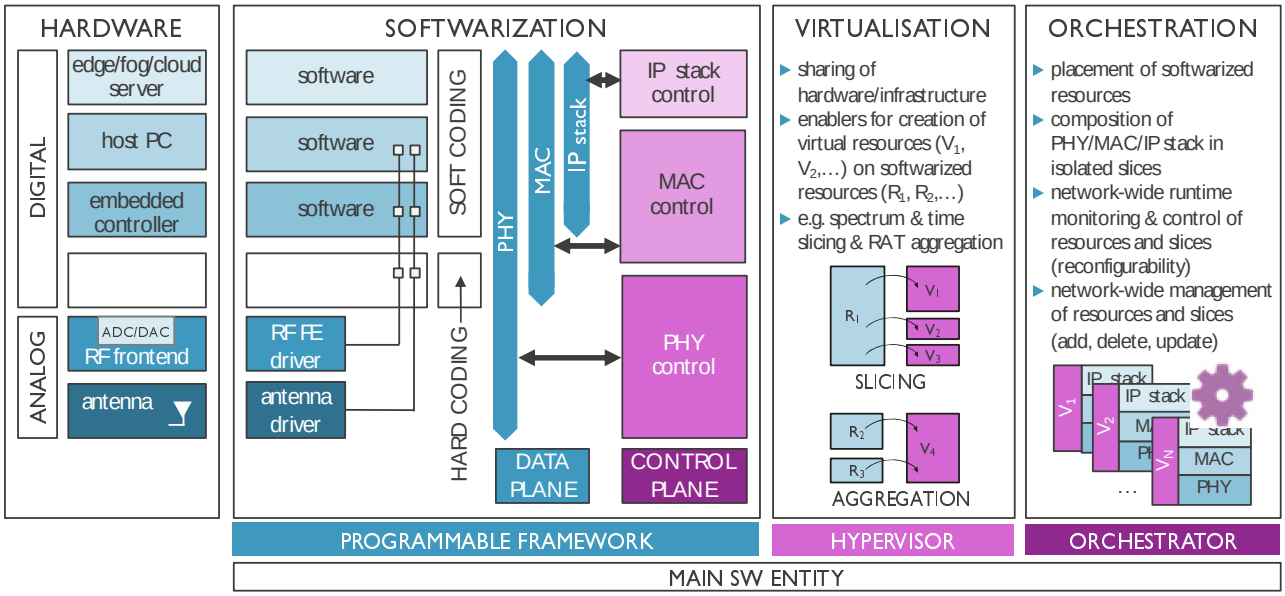


Figure 9: Illustration of Softwarization, virtualization and orchestration principles in the H2020 ORCA project

The ORCA features can be mapped to the WiSHFUL architecture as follows:

- The ORCA programmable framework can be mapped to the WiSHFUL Monitoring and Control Engine (MCE) exposing UPIs. The ORCA project extends the WiSHFUL programmable framework with a flexible PHY and advanced PHY control.
- The ORCA orchestrator can be mapped to the WiSHFUL control program(s). The ORCA project extends the management capabilities with deployment, installation and activation of radio processing function in programmable logic on FPGA.
- While the main focus of WiSHFUL is on Softwarization, the ORCA project also develops hypervisor capabilities for radio virtualization (both in software and programmable logic).

The vision of ORCA for ***end-to-end cross-domain orchestration*** is presented in Figure 10, [i.1], [i.2] and [i.31]. End-to-end networks can comprise multiple network segments, e.g. radio access networks, transport and core networks, and data centre networks, and these network segments are typically built for different purposes.

Network segments also use different media, such as optical fibre, copper cables, and wireless spectrum, and thus employ different technologies and protocols, e.g. xPON, xDSL and LTE, with unique configurations, policy enforcement and QoS management.

Hence, the creation of E2E network slices to provide guaranteed performance requires the slicing of each individual network segment, and the subsequent combination of these network segment slices. Network slices within a network segment are managed by an entity called an orchestrator, that orchestrates the use of network resources and the placement of functionality in a network segment, and also defines the configuration, policies, and management of a network segment.

In the ORCA vision, each network segment should have their own orchestrator, tailored to the segment's particularities, as illustrated in Figure 10 for the case of an E2E network with a wired and a wireless segment.

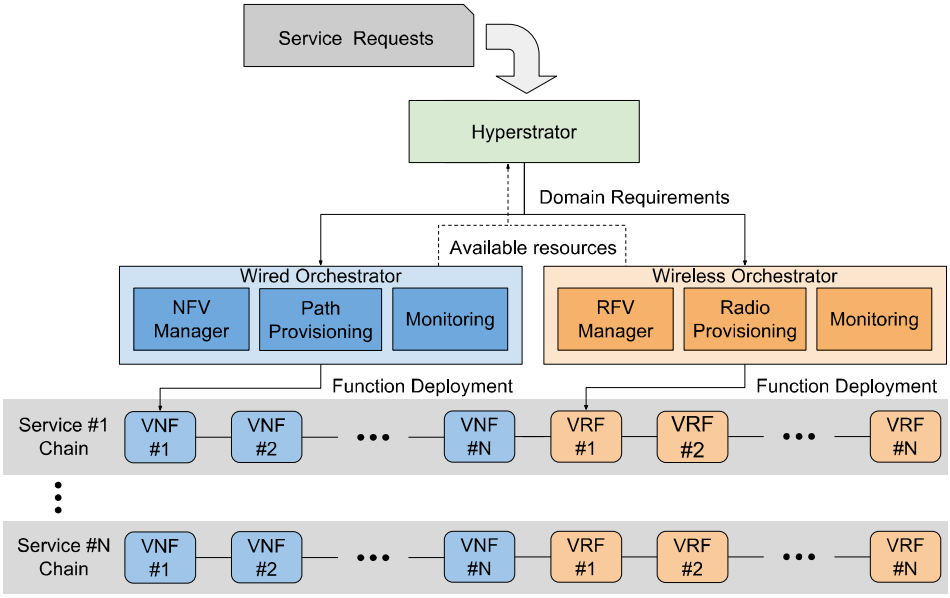
The communities behind each network segment have their own abstractions and models to manage the particularities of each segment. The use of a separate orchestrator for each network segment reduces complexity and breaks down the larger E2E network. In this way, each segment orchestrator can focus on a limited number of well-defined tasks, reducing the software complexity, both in terms of design and implementation.

E2E network slicing will require a combination of multiple types of orchestrators. Different types of orchestrators are deployed according to the type of resources being managed: wired network orchestrators for managing NFV (Network Function Virtualisation) and SDN (Software Defined Networking) and for establishing paths and deploying services; wireless network orchestrators for managing RRHs (Remote Radio Heads) and SDRs for creating RATs and provisioning radio access.

It is expected that there exists an entity with a global view of the available resources and the capabilities of each orchestrator for establishing and managing flexible E2E networks, leveraging the virtualization of each network segment. This entity would be an orchestrator of orchestrators, which coordinates the interaction between the underlying virtualized infrastructure, namely a hyperstrator.

The hyperstrator would sit on top of different orchestrators for controlling the E2E allocation of resources and management of the entire network. It would be the responsible for mapping high-level E2E network requirements into the require sites for the different networks segments, while each of the underlying orchestrators would then map their own requisites into a realization using the available virtualized resources.

The hyperstrator knows the available resources and the status of the current services by gathering information from its underlying orchestrators. Moreover, the hyperstrator should coordinate the combination of slices between network segments for creating E2E network slices. Therefore, it is crucial for the hyperstrator to be aware of the points of presence between network segments, as these are the places where network segments interface and interconnect.



NOTE: The hyperstrator receives service requests and decides the service resource requirements for each network segment and delegates these to each segment orchestrator. Each orchestrator then provisions resources and deploys services as a chain of virtual functions.

Figure 10: Cross-domain orchestration

7 Instantiation of GANA Functional Blocks by Mapping WiSHFUL architecture components to GANA Concepts and Architectural Principles

7.1 General Mapping of WiSHFUL Architectural Concepts and Principles to GANA Concepts and Principles

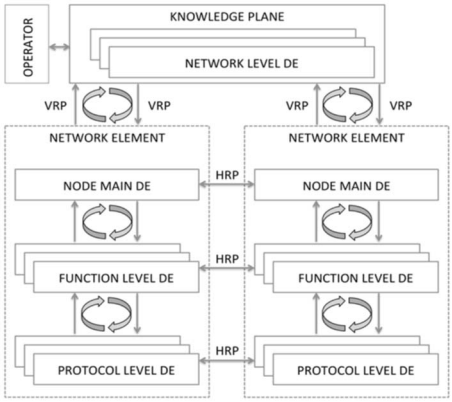
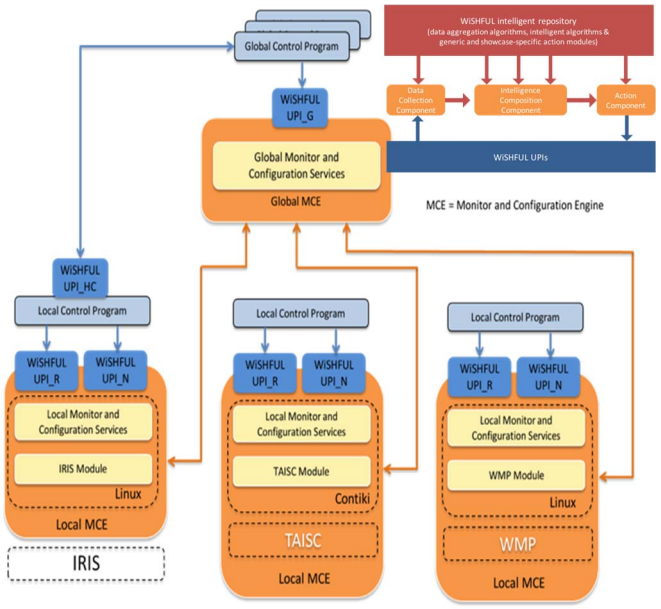
This clause provides a mapping between functional components of the GANA and the WiSHFUL architectures. It is worth noting that the two architectures have been designed independently and that the WiSHFUL architecture was not specifically designed for autonomic networks, as its main focus is for simplifying wireless experimentation on heterogeneous radio platforms.

However, they both enable combining and interworking centralized and distributed control for network and services [i.8]. The comparative analysis of the two architectures reveals a good matching of their components, justified by the common concepts and principles that are behind, indicated in clause 4 (network compartmentation, function atomization and composition, closed control loop, context recognition and adaptation).

In some cases, the mapping of the functional blocks is almost perfect, with small differences only in naming conventions, in other cases there are significant differences in concepts and functional elements.

The architectural components of GANA and WiSHFUL are compared in tabular form in Table 2 with specific focus on the enabling elements for autonomic networks.

Table 2: Comparison and Mapping between GANA and WiSHFUL concepts

GANA	WiSHFUL ARCHITECTURE
	
Node-Main-DE - Decision-making Element	Local MCE - Local Monitor and Configuration Engine
Function level DE Protocol level DE	These Decision-making Elements (DEs) are embedded in the programmable platforms and are out of the scope of WiSHFUL goals. However, WiSHFUL provides adaptation modules to make such function and protocol levels available to node-level decision-making elements in a unified form (UPIs). Additionally, clause 7.3, provided in the present document, provides insights on how Function Level DEs (GANA Level 2 DEs) can be implemented jointly with the Node-Main-DE (GANA Level 3) within the same space of abstraction in a Node but still preserving the GANA hierarchy and interactions among the GANA Level 3 and Level 2 DEs and making both levels use a unified API (UPI) to dynamically and autonomically configure the Managed Entities (MEs) of the Node.
Network level DE	Global Control Program (Controller), with a Global MCE - Global Monitor and Configuration Engine that serves the multiple Knowledge Plane (KP) DEs. The clause "Network-level programmability and the Mapping to GANA Network Level (Knowledge Plane (KP) Level) Autonomics", provided in clause 7.4, provides insights on how the GANA Knowledge Plane DEs (Network Level DEs) and other entities of the Knowledge Plane such as MBTS and ONIX can be implemented in the WiSHFUL framework.
Reference Point (Rfp)	UPI (Various UPIs defined and implemented in the WiSHFUL framework can be used to realize some corresponding Reference Points defined in the GANA Framework). More details can be found in clause 7.7 provided in the present document.
Horizontal Reference Point (for communication between peer components of Network Elements)	WiSHFUL does not define, by proposal, any specific communication interface between peer components because this choice is left to the implementer, which can use existing protocols or define new ones.
Vertical Reference Point (for communication between Network Elements and the knowledge plane)	UPI_R and UPI_N
Network Element	Programmable wireless node

GANA	WiSHFUL ARCHITECTURE
Managed Entity (ME)	• Radio Program. • Control Program. • Wireless Node. • Other Configurable resources and parameters in the Node, including Protocol Stacks, Network Interface Cards (NIC), monitoring components and mechanisms, physical and virtual resources such as memory, configurable system (operating system) functions, and other types of MEs that derive from Table 3 in ETSI TS 103 195-2 [i.9] (more details on this subject are found in the clause 7.5.
Knowledge Plane (KP)	Global Control Programs, Global MCE + WiSHFUL intelligence Repository

WiSHFUL does not distinguish the four levels of abstractions provided by GANA (protocol-level, function-level, node-level, network-level) but only radio and network levels.

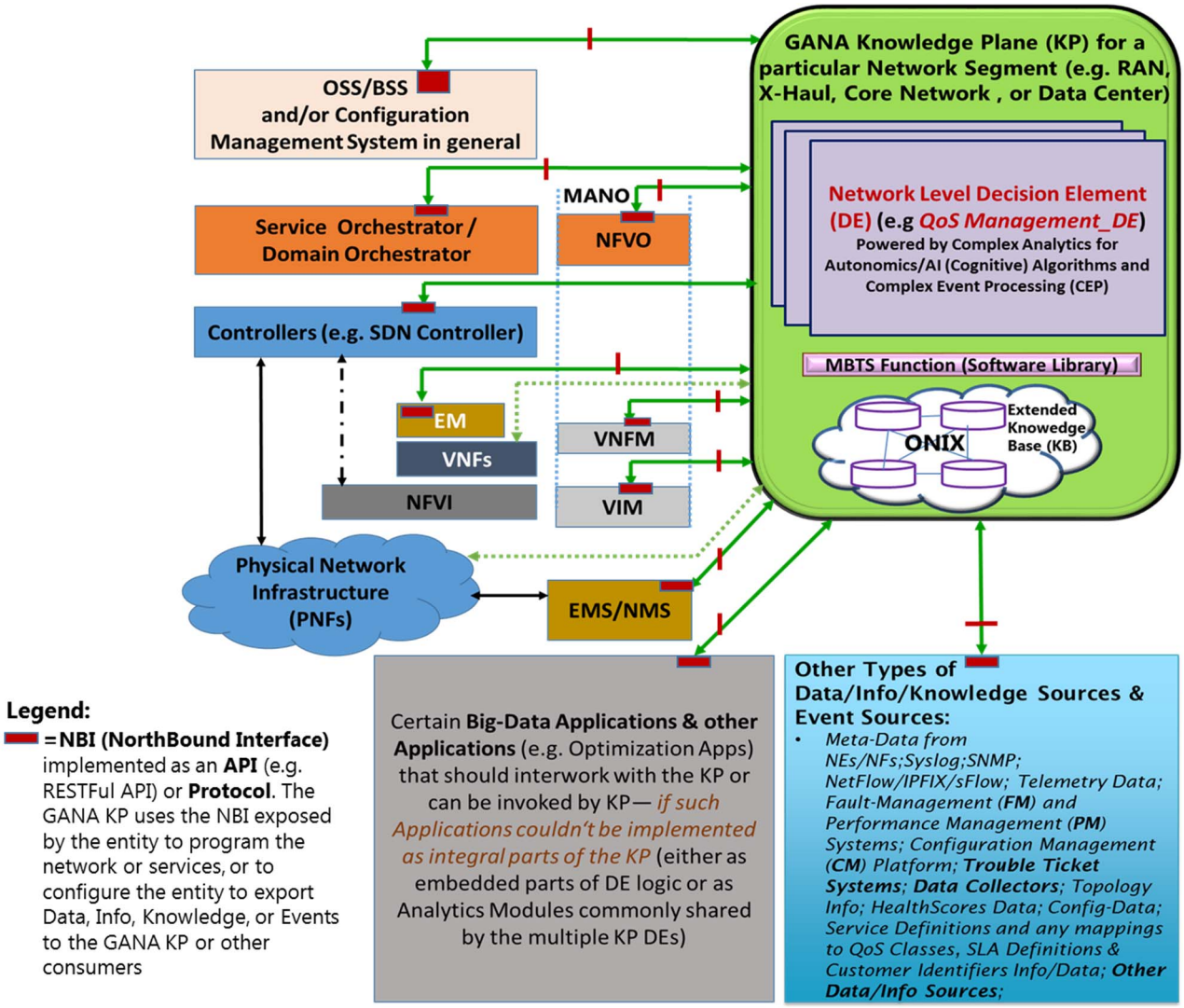
In WiSHFUL, decision-making entities (DE) at protocol and function levels are embedded in the programmable platforms that are supported by the project and that were defined externally to the project. This means that those DE have heterogeneous reference points therefore WiSHFUL provides adaptation modules in order to expose unified vertical reference points, which are named Unified Program Interface (UPI) in WiSHFUL.

This vision does not contrast with the statement in ETSI TR 103 495 [i.7], that is reported in the following: "*Since the Protocol-Level involves embedding an intrinsic control loop within an individual protocol, it may not be necessary to introduce such "intelligence" into individual protocols, but rather to focus on introducing autonomicity (control loops) at higher levels of abstraction, starting from the level directly above (i.e. the Function-Level that defines "functions" which abstract individual protocols and mechanisms), up to the Network-Level".*

7.2 Autonomic networks and General GANA integration with SDN, NFV, Big Data Analytics Applications, OSS/BSS Systems, Orchestrators, and Other Management and Control Systems

The question of how to apply GANA principles for Automated and Autonomic Management & Control (AMC) in environments involving Software Defined Networking paradigm (SDN), NFV, Big Data Analytics and other management and control systems that may be targeted for use in those environments, is answered by work already done in ETSI on GANA and SDN, GANA and NFV and the Unified architecture for ETSI GANA, SDN, NFV, Big Data, and E2E Orchestration, as illustrated on Figure 11 and Figure 12 taken from [i.20] and [i.32]. The following resources provide much more additional useful information on this subject:

- ETSI TS 103 195-2 [i.9]
- ETSI White Paper No.16 [i.8]
- ETSI TR 103 473 [i.10]



NOTE: This figure also addresses the subject of KP integration with Event Sources, Data Sources and Info/Knowledge Sources.

Figure 11: The Integration of the GANA Knowledge Plane (KP) with various management and control systems through which the Knowledge Plane can selectively program the network

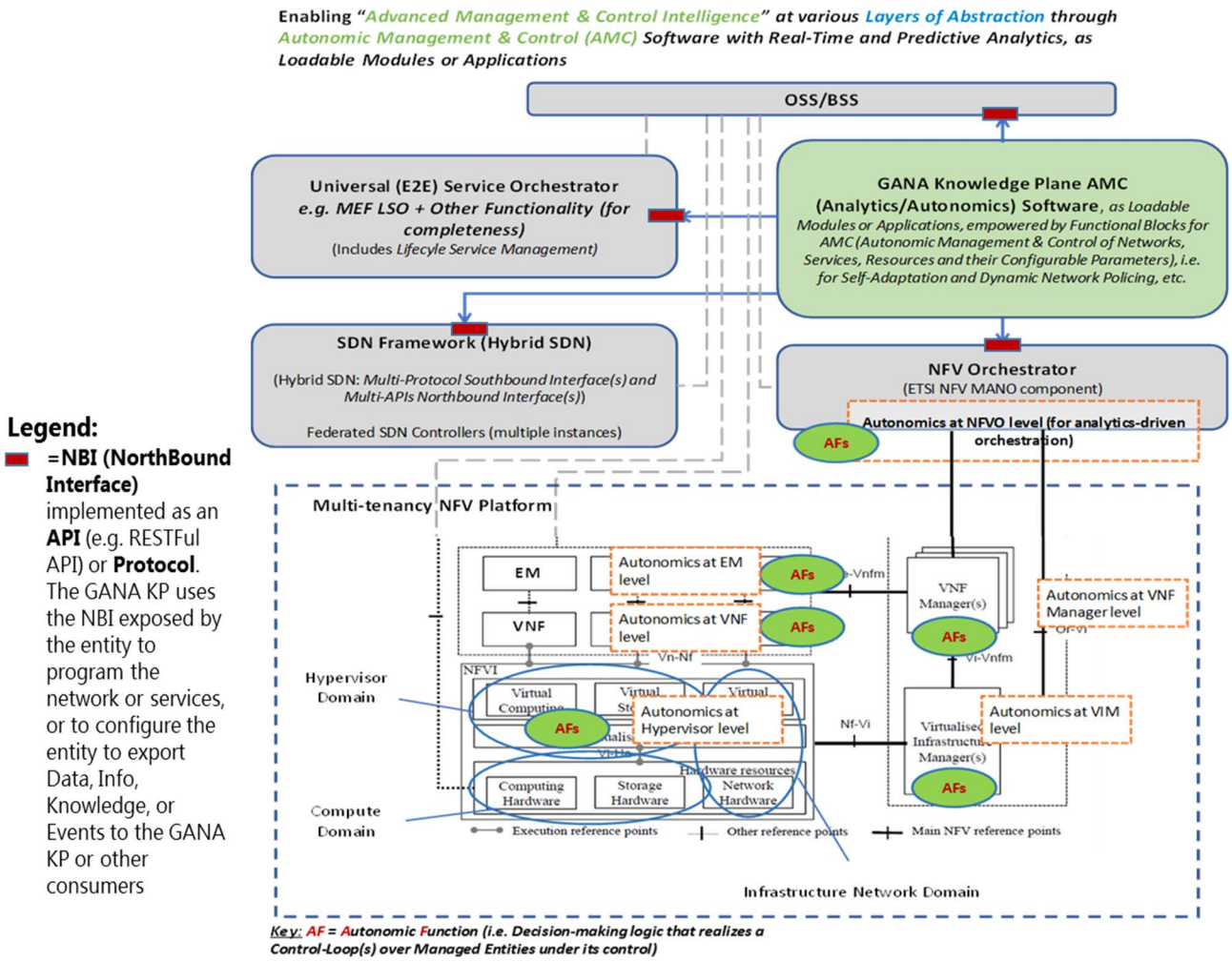
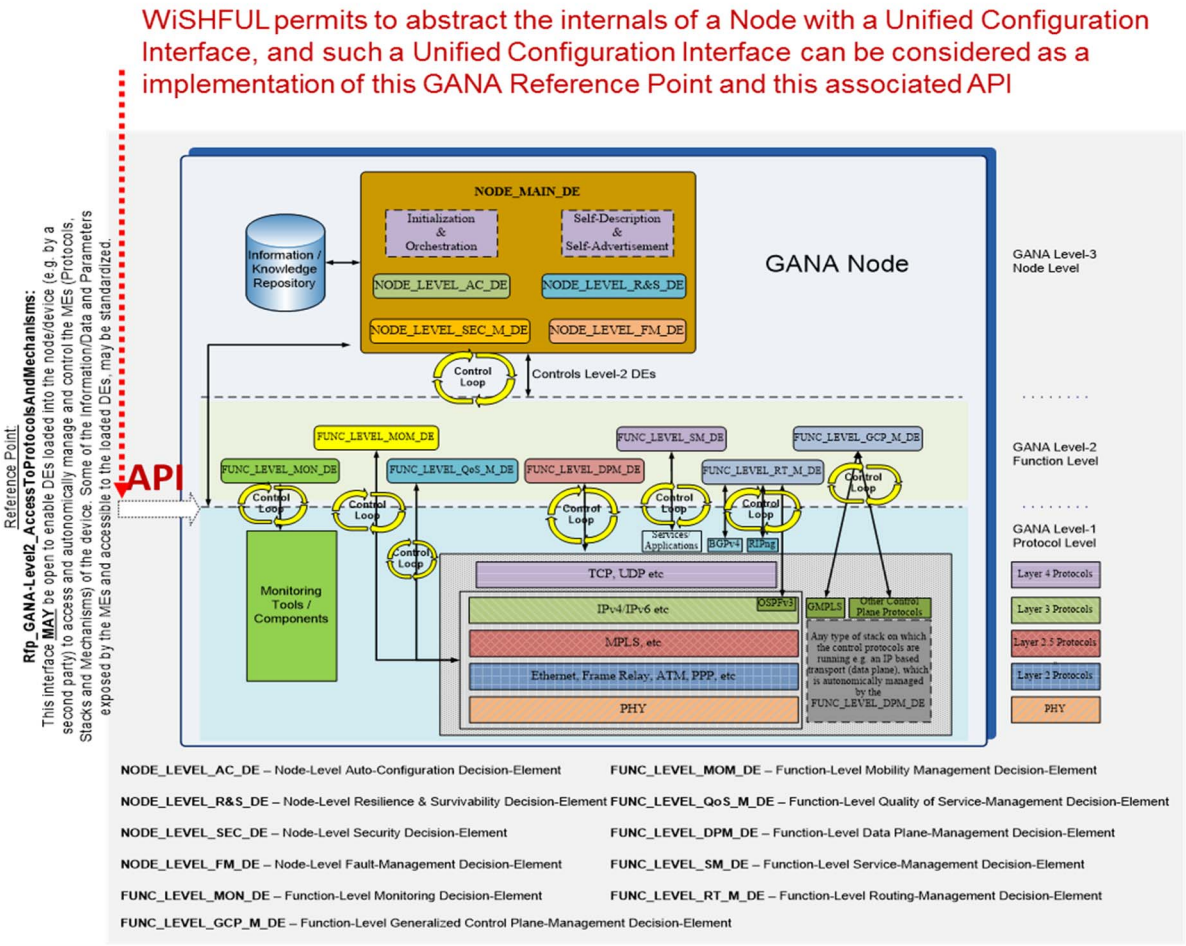


Figure 12: Multi-Layer Autonomics and the integration of the GANA Knowledge Plane with Orchestrators, SDN Controllers, NFV, and OSS/BSS systems

7.3 WiSHFUL Node-level programmability and Mapping to GANA Node-Level and Lower Levels Autonomics

Figures 13, 14, 15 and 16 illustrate how the WiSHFUL implementation provides for an approach to implementing the GANA Node's internal API that enables GANA Levels 2 and 3 DE innovators to implement and load the DEs to drive the autonomic operations of a network node (Network Element/Function (NE/NF)) as described in ETSI TS 103 195-2 [i.9].



NOTE: This figure also addresses how the same API can be used by DEs to access and configure other types of managed resources or mechanisms.

Figure 13: How the WiSHFUL Implementation provides an Implementation of the API that enables DEs to access and configure protocol stacks and OSI layer 7 or TCP/IP application layer applications

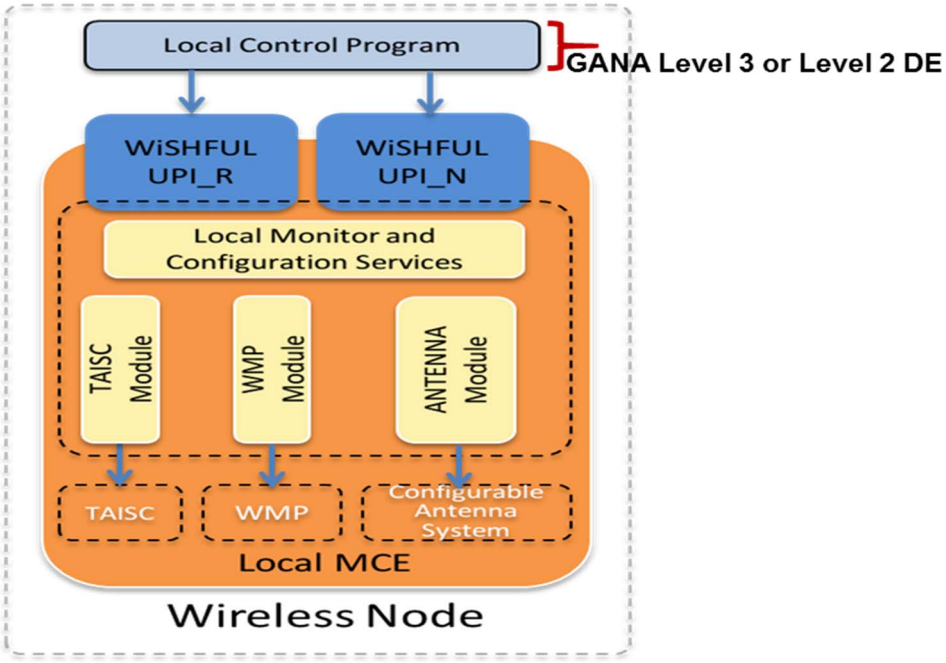
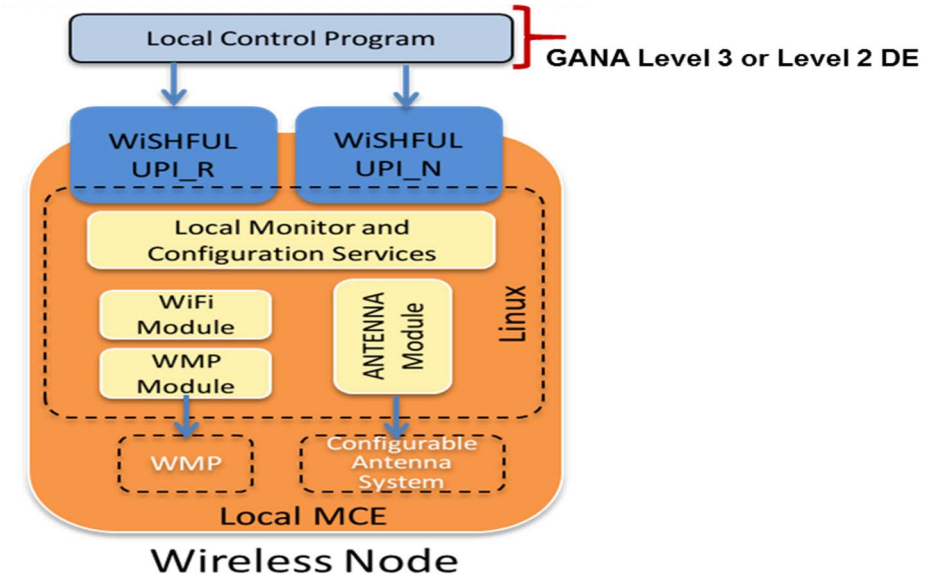


Figure 14: A GANA level 2 or Level 3 DE is Local Control Program in the WiSHFUL Framework



NOTE: This figure also provides illustrations on the Linux environment already experimented within the WiSHFUL Project.

Figure 15: A GANA level 2 or Level 3 DE is treated as a Local Control Program in the WiSHFUL Framework

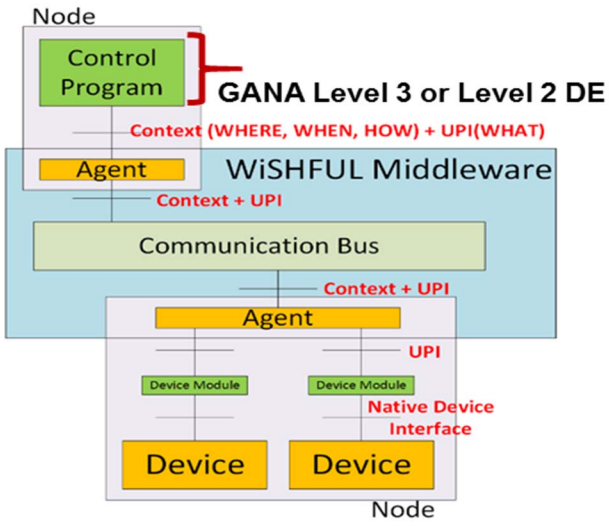


Figure 16: A GANA level 2 or Level 3 DE is treated as a Local Control Program in the WiSHFUL Framework (with additional insights concerning contexts)

7.4 WiSHFUL Network-level programmability and the Mapping to GANA Network Level (Knowledge Plane (KP) Level) Autonomics

Figure 17 illustrates how to use WiSHFUL Network-level programmability components to implement GANA Knowledge Plane level autonomics, as well as illustrating how the integrate that with lower level GANA autonomics in the NE/NF level (GANA Levels 2 and 3 DEs).

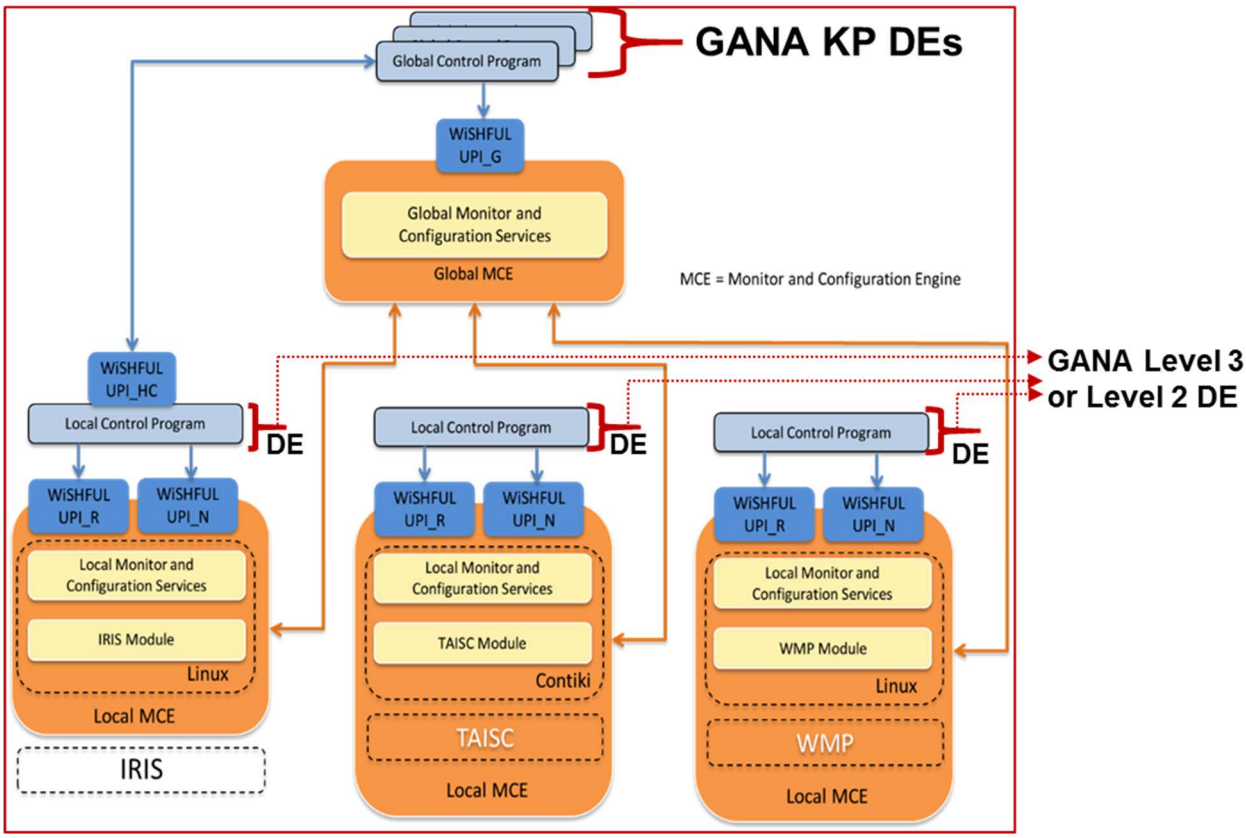
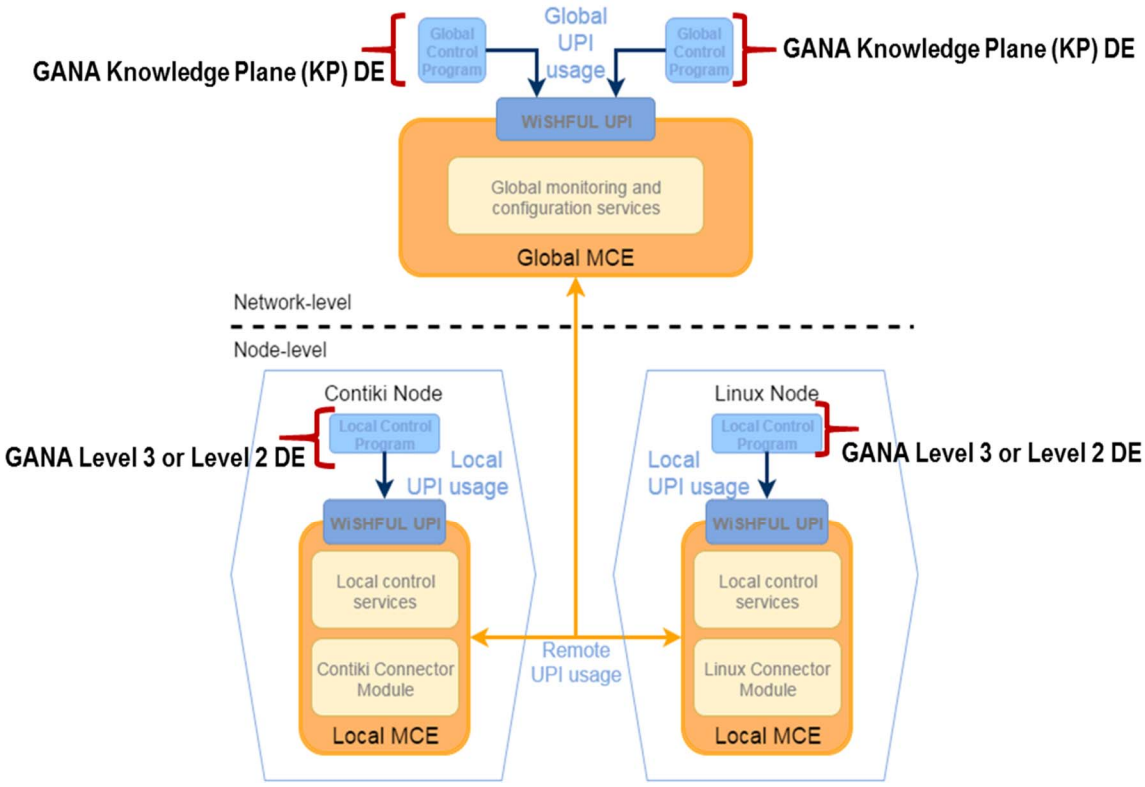


Figure 17: A GANA level 2 or Level 3 DE is treated as a Local Control Program in the WiSHFUL Framework, and a GANA Knowledge Plane (KP) Level DE is a Global Control Program in the WiSHFUL Framework



NOTE: This figure also provides illustrations of the Contiki [i.21] and Linux Nodes experimented with in WiSHFUL Project.

Figure 18: A GANA level 2 or Level 3 DE is Local Control Program in the WiSHFUL Framework and a GANA Knowledge Plane (KP) Level DE is a Global Control Program in the WiSHFUL Framework

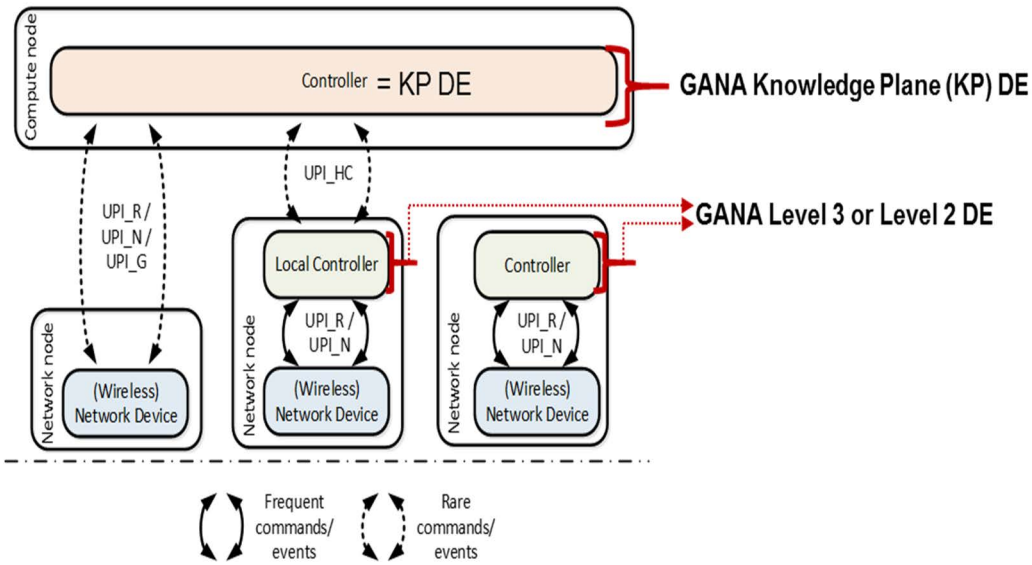


Figure 19: Various implementations scenarios that can be considered in implementing GANA DEs as controllers

7.5 Parameter and Functionality Mappings for DE-to-ME Associations that enable DE implementers to implement DEs

As already discussed in clause 5, the GANA Standard (ETSI TS 103 195-2 [i.9]) defines and standardizes various Autonomic Functions (Decision-making- Elements (DEs)) that can be instantiated to operate at NE/NF level and/or within the GANA Knowledge Plane (KP).

The process called GANA instantiations onto an implementation-oriented network architecture and its associated management and control architecture establishes the kind of GANA DEs that should be instantiated to operate in an NE/NF and/or in the GANA Knowledge Plane.

Various types of Decision Elements (DEs) are defined in ETSI TS 103 195-2 [i.9], and also their associated mappings to their types of Managed Entities (MEs) - i.e. resources and configurable parameters that should be under the responsibility of the specific DE.

Specific DEs and their mappings to specific MEs are then further detailed in concrete GANA instantiations onto a particular target implementation oriented network architecture and its management and control architecture (e.g. Broadband Forum (BBF) architectures (ETSI TR 103 473 [i.10]), 3GPP Backhaul and Core Network (ETSI TR 103 404 [i.11])).

The following is the approach that should be taken by implementers of DEs when using the WiSHFUL Framework and the code that is already implemented the WiSHFUL Proof-of-Concept (PoC):

1) DEs implementers should first produce a table that provides a Mapping of the GANA standardized DEs to their corresponding Managed Entities (MEs) and the associated configurable parameters of the MEs while respecting the 1-to-1 Mapping of DE to an ME Parameter ("1-ME-Param" to "1-DE Mapping") as described in ETSI TS 103 195-2 [i.9]. This means that for the environment in which the WiSHFUL framework is to be applied in implementing the GANA a table that is a concretization (instantiation) of Table 3 found in ETSI TS 103 195-2 [i.9] should be produced. The produced table should capture the DE-to-ME mappings such that the GANA standardized DEs that can be implemented in a specific NE/NF type and/or in the GANA Knowledge Plane have their corresponding mappings to the ME and parameters they are supposed to autonomically management and control.

2) Once the table of DEs-to-MEs Mappings Table has been created, DE implementers can use some software code of local programs (local controllers) and global programs (global controllers) already implemented in WiSHFUL Framework to implement the standardized GANA DEs required to operate at specific GANA Levels. It may happen that some controllers (local and global programs) already implemented in the WiSHFUL Framework do not necessarily map 1-to-1 with the DEs in the table of DEs-to-MEs Mappings.

Since the GANA framework allows to implement DEs in two ways as illustrated in Figure 20 below (extracted from ETSI TS 103 195-2 [i.9]) it should be possible to still be able to use the code of some controllers (programs) in the WiSHFUL framework to implement GANA DEs in either of the approaches indicated in Figure 20 below.

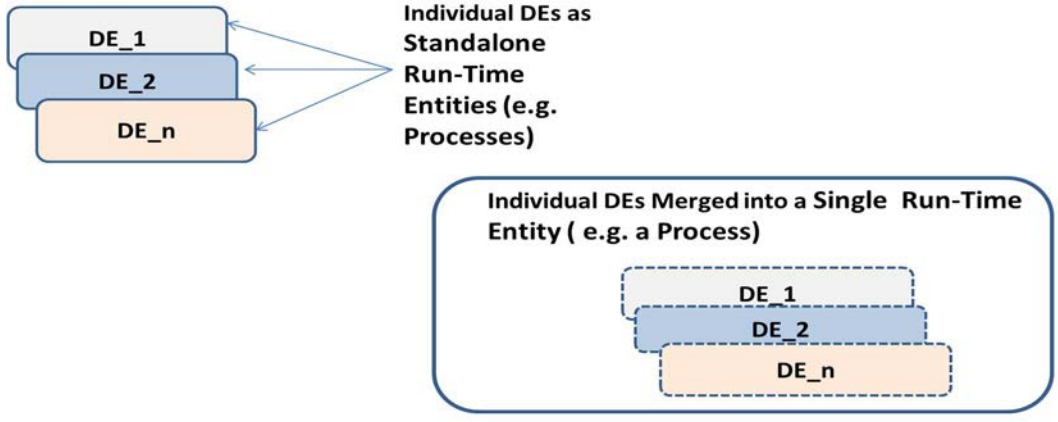


Figure 20: Illustration of possible approach to implementing GANA Levels 2, 3 and 4 DEs at run-time (extract from ETSI TS 103 195-2 [i.9])

7.6 Instantiation of the GANA Knowledge Plane (KP) in the WiSHFUL Intelligence Framework

Figure 21 provides insights on the integration of the GANA Knowledge Plane DEs, MBTS, ONIX, Global MCE, GANA Levels 2&3 and Local MCE in the WiSHFUL Framework.

Regarding the ONIX part, ETSI TS 103 195-2 [i.9] provides more insights on how the ONIX system can be implemented, while the diagram illustrates an example of an Information Server member of an ONIX system (namely a Security Information and Incidents related Repository).

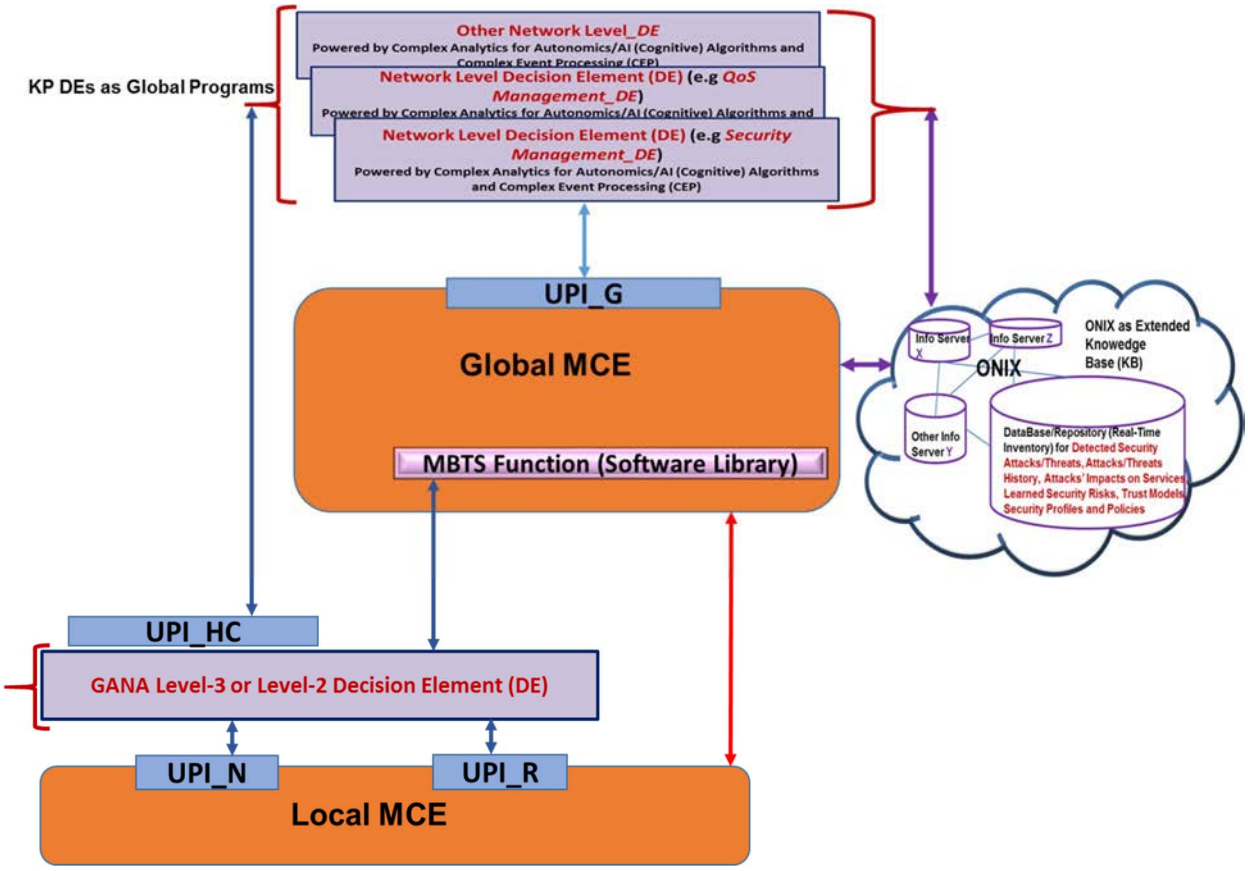
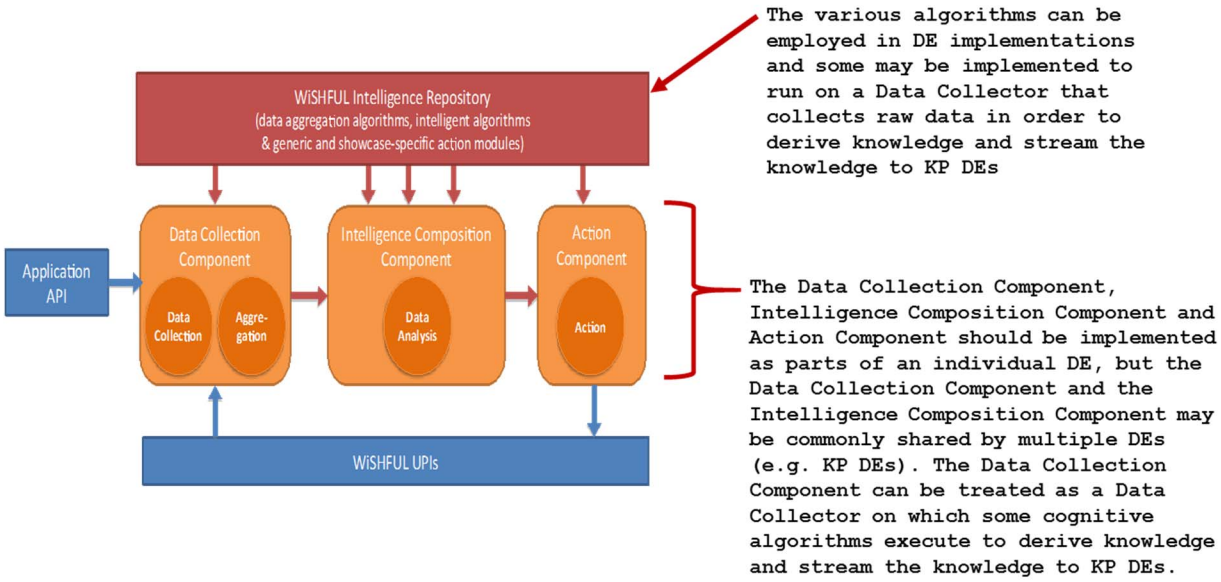


Figure 21: Integration of the GANA Knowledge Plane DEs, MBTS, Global MCE, GANA Levels 2&3 and Local MCE in the WiSHFUL Framework

Figure 22 provides insights on how Data Collection Component, Intelligence Composition Component and Action Component are to be considered in design of a DE, and the various algorithms that can be employed in DE implementations while some algorithms may be implemented on Data Collectors to run as some cognitive algorithms execute to derive knowledge and stream the knowledge to KP DEs.



NOTE: This figure also provides insights on the various algorithms that can be employed in DE implementations while some algorithms may be implemented on Data Collectors.

Figure 22: How Data Collection Component, Intelligence Composition Component and Action Component are to be considered in design of a DE

7.7 Instantiation (Implementation) of GANA Reference Points in the WiSHFUL Architecture Implementation

This clause provides insights on GANA Reference Points implementation in the WiSHFUL Architecture.

NOTE: Table 3 has been extracted from clause 6.3 of ETSI TS 103 195-2 [i.9], and a column (last column) has been indicated to comment on the implementation of the corresponding Reference Point (Rfp) in the WiSHFUL Architecture. All the Reference Points (Rfps) and their associated acronyms are described in ETSI TS 103 195-2 [i.9].

Table 3: Reference Point in GANA and corresponding Reference Point (Rfp) in the WiSHFUL Architecture

Reference Point Name (see note 1)	Alias Name of the Rfp	Characteristic Information communicated over the Reference Point	Additional Comments on where the Reference Point is described	Implementation of the Reference Point in the WiSHFUL Architecture
Rfp_GANA-Level2&3-AccessToProtocolsAndMechanisms (see note 2) See note 5	DeMe	• **Views:** e.g. event notifications, monitoring data are communicated to Function-level-DEs by their specifically assigned Managed Entities (MEs)-i.e. Protocols, Stacks and Mechanisms (see clause 6.4.2.1 in ETSI TS 103 195-2 [i.9] and clauses 9.11.5 and 9.11.6 of ETSI GS AFI 002 [i.15] on assignment of DEs to specific types of MEs). • **Commands** are issued by a specific Function-Level-DE e.g. Function-Level-Routing-Management-DE, to its specifically assigned Managed Entities (i.e. protocols and mechanisms such as routing protocols and mechanisms) in order to (re)-configure and regulate the behaviour of the ME(s).	This node/device internal interface is meant to enable the loading of DEs coming from other parties other than the device vendor. The DEs would access and autonomically manage and control the Protocols and Mechanisms of the device. See clause 9.6 of ETSI GS AFI 002 [i.15]. See Figure 19 in ETSI GS AFI 002 [i.15]. See also clause 6.4.2.1 in ETSI TS 103 195-2 [i.9].	This has been implemented in the WiSHFUL architecture as an API of a Network Node (GANA Node) as discussed in clauses 5 to 7 of the present document.
Rfp_FunctionLevelDE-to-FunctionLevelDE	FuDe FFuDe (for federated AMC across different domains)	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Domain Type(s)** to which a DE involved is bound need to be exchanged. **Domain Identifier(s)** of DEs hosted by entities belonging to different administrative domains need to be exchanged.	See clauses 9.8 and 11.10 of ETSI GS AFI 002 [i.15]. See Figure 20 and Figure 34 in ETSI GS AFI 002 [i.15]. For Domain Type(s) and Domain Identifier(s) refer to the clause on Federation in GANA and in ETSI GS AFI 002 [i.15]. See also clauses 6.4.3.1 and 6.4.4.2 in ETSI TS 103 195-2 [i.9].	This is for Further Study, on the components of the WiSHFUL architecture implementation and/or other communications means (e.g. protocols or APIs) that can be used by DE implementers that innovate distributed algorithms that involve FunctionLevelDE-to-FunctionLevelDE across as set of NEs/NFs along a path in the network infrastructure.

Reference Point Name (see note 1)	Alias Name of the Rfp	Characteristic Information communicated over the Reference Point	Additional Comments on where the Reference Point is described	Implementation of the Reference Point in the WiSHFUL Architecture
		• **Views** can be communicated by a particular Function-Level-DE to other peer Function-level-DEs on other nodes/devices, especially concerning events or issues a function of a node e.g. Routing-Function cannot resolve (by performing some action) without jeopardizing network integrity (objectives). • **Control Information** exchange between Function-Level-DEs via the DE-2-DE interactions to achieve a "network-intrinsic management and control". Such interactions may include the notion of "compartment formation, policies of operation and compartment management" by DE-2-DE communication in a distributed fashion.		
Rfp_NodeMainDE-to-NodeMainDE	NoDe FNoDe (for federated AMC across different domains)	Similar types of Characteristic Information as in the case of the Reference Point "Rfp_FunctionLevelDE-to-FunctionLevelDE". The difference being the scope for which the Characteristic Information applies i.e. this case applies to the scope of the node/device level than a particular Function-Level (lower level).	See clauses 9.8 and 11.10 in ETSI GS AFI 002 [i.15]. See Figure 20 and Figure 34 in ETSI GS AFI 002 [i.15]. See also clauses 6.4.3.2 and 6.4.4.3 in ETSI TS 103 195-2 [i.9].	This is for Further Study, on the components of the WiSHFUL architecture implementation and/or other communications means (e.g. protocols or APIs) that can be used by DE implementers that innovate distributed algorithms that involve NodeMainDE-to-NodeMainDE across as set of NEs/NFs along a path in the network infrastructure.
Rfp_NetworkLevelDE-to-NodeMainDE	NeMe	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Domain Type(s)** to which a DE involved is bound need to be exchanged. **Domain Identifier(s)** of DEs hosted by entities belonging to different administrative domains need to be exchanged.	See clauses 9.8, 9.9 and 9.13.5 in ETSI GS AFI 002 [i.15]. See Figure 21, Figure 22 and Figure 34 in ETSI GS AFI 002 [i.15]. For Domain Type(s) and Domain Identifier(s) refer to the clause on Federation in GANA in ETSI GS AFI 002 [i.15]. See also clause 6.4.2.3 in ETSI TS 103 195-2 [i.9].	This has been implemented in the WiSHFUL architecture by UPIs described in clauses 5 to 7 of the present document.

Reference Point Name (see note 1)	Alias Name of the Rfp	Characteristic Information communicated over the Reference Point	Additional Comments on where the Reference Point is described	Implementation of the Reference Point in the WiSHFUL Architecture
		• **Views** are communicated to Network-level-DEs, especially concerning events or issues a node/device cannot resolve (by performing some action) without jeopardizing network integrity (objectives). • **Commands** may be issued by a Network-Level-DE to the node or to a Function-Level-DE via the Node-Main-DE.		
Rfp_ModelBasedTranslationService-to-NodeMainDE	NeMe	• This is a refinement of the Reference Point "Rfp_NetworkLevelDE-to-NodeMainDE" to involve a case whereby Network-Level-DEs communicate with a Node-Main-DE via a Model-Based-Translation Service (MBTS) that translates COMMANDS from Network-Level-DEs and RESPONSES from nodes/devices to a form usable by the targeted entity.	See clauses 9.13.5 and 11.7 in ETSI GS AFI 002 [i.15]. See Figure 64 and Figure 34 in ETSI GS AFI 002 [i.15]. See also clause 6.4.2.5 in ETSI TS 103 195-2 [i.9].	This has been implemented in the WiSHFUL architecture by UPIs described in clauses 5 to 7 of the present document.
Rfp_AMC-ModelBasedTranslationService-to-ONIX	NoI	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary.	See clauses 9.13.5 and 11.7 in ETSI GS AFI 002 [i.15]. See Figure 64 and Figure 34 in ETSI GS AFI 002 [i.15]. See also clause 6.5.1 NoI Rfp in ETSI TS 103 195-2 [i.9].	This is for Further Study, on how the WiSHFUL architecture implementation (in relation to the Global MCE component and its embedment of an MBTS function) can be integrated with an ONIX implementation.

Reference Point Name (see note 1)	Alias Name of the Rfp	Characteristic Information communicated over the Reference Point	Additional Comments on where the Reference Point is described	Implementation of the Reference Point in the WiSHFUL Architecture
		• **Operations/Messages for Storing and Retrieving Information from the ONIX system.** For example, the MBTS can use the publish/subscribe services of the ONIX that enable Advanced Auto-Discovery of Information and Resources, to retrieve Information about Network Elements/Nodes, such as Capability Description Models of individual nodes/devices, self-advertised/published by an individual node/device upon initialization. Capability Models of a node/device include technological features supported, including management protocols supported and Information about Managed Objects (MOs) of the technologies (e.g. protocols, etc.). Capability Models may include apart from technological features, vendor information.		
Rfp_NetworkLevelDE-to-NetworkLevelDE	NeDe FNeDE (for federated AMC across different domains)	• **"Views"** such as **Policy changes** by the human operator; **challenges to the network's operation** from the perspective of a particular DE e.g. detected faults, threats, etc.; "views" communicated from lower-Level DEs in nodes/devices that require Net-Level-DEs to share and act upon if necessary. • **Domain Type(s)** to which a DE involved is bound need to be exchanged. **Domain Identifier(s)** of DEs hosted by entities belonging to different administrative domains need to be exchanged. • **Negotiations and Synchronization of Actions and Policies.**	This Reference Point between Network-Level-DEs is independent of the types of Network-Level-DEs and so should be considered as a common type of Reference Point between any Network-Level-DEs. See clauses 9.9, 9.13.5 and 11.7 in ETSI GS AFI 002 [i.15]. See Figure 22, Figure 34 and Figure 64 in ETSI GS AFI 002 [i.15]. For Domain Type(s) and Domain Identifier(s) refer to the clause on Federation in GANA in ETSI GS AFI 002 [i.15]. See also clauses 4.3.3 and 6.4.4.4 in ETSI TS 103 195-2 [i.9].	This is for Further Study, on the components of the WiSHFUL architecture implementation and/or other communications means (e.g. protocols or APIs) that can be used by DE implementers that innovate logically centralized algorithms for Knowledge Plane DEs that involve NetworkLevelDE-to-NetworkLevelDE communications requirements.

Reference Point Name (see note 1)	Alias Name of the Rfp	Characteristic Information communicated over the Reference Point	Additional Comments on where the Reference Point is described	Implementation of the Reference Point in the WiSHFUL Architecture
Rfp_NetworkLevelDE-to-Data_Storage	NeI	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages from the DE for retrieval of Data or (Knowledge created out of raw data)** from a storage such as Data Collector that gathers data such as: IPFIX Data, SNMP BulkStats Data, NetFlow Data, Flow Traces, Traffic Matrix, etc., OR Any Data that is not suitable to be stored and shared through the ONIX system. • **Knowledge created out of raw data by Algorithms running on the Data Storage, that operate on raw data and create Knowledge for export to the Knowledge Plane (i.e. to Net-Level-DEs).** • **Data that may need to be communicated by the Storage to the particular DE.**	See clauses 11.7 and 9.13.7 in ETSI GS AFI 002 [i.15]. See Figure 64 and Figure 38 in ETSI GS AFI 002 [i.15]. See also clause 6.5.2 in ETSI TS 103 195-2 [i.9].	This is for Further Study, on the components of the WiSHFUL architecture implementation and/or other communications means (e.g. protocols or APIs) that can be used by DE implementers that innovate logically centralized algorithms for Knowledge Plane DEs and their need to consume Knowledge extracted from raw data stored on a Data Collector by Analytics/Cognitive Algorithms running on the Data Storage component/server (Data Collector). More details on this subject can be found in ETSI TS 103 195-2 [i.9] with regards to the RAT (Representation, Acquisition and Translation) Function that can be implemented on a Data Collector and made to integrate with the GANA Knowledge Plane components.
Rfp_NetworkLevelDE-to-ONIX-System	NeI	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages for Storing and Retrieving Information from the ONIX system.**	See clauses 9.13.4 and 10 in ETSI GS AFI 002 [i.15] on the Use of Network Profiles, Policies, Objectives, Config-Data, and Capabilities of network elements; and 11.7 in ETSI GS AFI 002 [i.15]. See Figure 64 of ETSI GS AFI 002 [i.15]. See also clause 6.5.2 in ETSI TS 103 195-2 [i.9].	This is for Further Study, on how the WiSHFUL architecture implementation (in relating to the KP DEs as global programs) can be integrated with an ONIX implementation.

Reference Point Name (see note 1)	Alias Name of the Rfp	Characteristic Information communicated over the Reference Point	Additional Comments on where the Reference Point is described	Implementation of the Reference Point in the WiSHFUL Architecture
Rfp_NodeMainDE-to-ONIX-System	NoI	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages for Storing and Retrieving Information from the ONIX system.**	See clauses 9.13.4 and 10 in ETSI GS AFI 002 [i.15] on the Use of Network Profiles, Policies, Objectives, Config-Data, and Capabilities of network elements; and clause 11.7 in ETSI GS AFI 002 [i.15]. See also Figure 34 in clause 9.13.5 in ETSI GS AFI 002 [i.15] See Figure 64 and Figure 34 in ETSI GS AFI 002 [i.15]. See also clause 6.5.1 in ETSI TS 103 195-2 [i.9].	This is for Further Study, on how the WiSHFUL architecture implementation (in relating to the Node internal local program implemented as a GANA Node-Main-DE and its interworking with the local MCE) can be integrated with an ONIX implementation.
Rfp_OSS-to-ONIX-System	OsI	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Operations/Messages for Storing and Retrieving (mainly) Information from the ONIX system.**	See clauses 11.7 and 11.8 in ETSI GS AFI 002 [i.15]. See Figure 67 in particular, Figure 64 and Figure 68 in ETSI GS AFI 002 [i.15]. See also clause 6.5.3 in ETSI TS 103 195-2 [i.9].	This is not specific to the WiSHFUL architecture, and there are some solutions that either already exist or can be developed in the industry.
Rfp_OSS-to-Network-Level-DEs	OsDe	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Management COMMANDS normally sent to the network by an OSS** through the so-called "network-adapter interface" need to be rather sent directly to the Network-Level-DE (considering that they are ones that take the full responsibility of performing Autonomic Management of the Network), and NOT to the network directly.	This case applies to configurations where OSS systems are integrated to co-exist and interwork harmoniously with Network-Level-DEs in the overall management of the network. The current (today's) OSS-Network Interface would need to be "re-directed" towards Network-Level-DEs (assuming that Network-Level-DEs take full responsibility for network management and control). See clauses 11.7 and 11.8 in ETSI GS AFI 002 [i.15]. See Figure 67 in particular, Figure 64 and Figure 68 in ETSI GS AFI 002 [i.15]. See also clause 6.4.2.4 in ETSI TS 103 195-2 [i.9].	This is not specific to the WiSHFUL architecture, and there are some solutions that either already exist or can be developed in the industry.

Reference Point Name (see note 1)	Alias Name of the Rfp	Characteristic Information communicated over the Reference Point	Additional Comments on where the Reference Point is described	Implementation of the Reference Point in the WiSHFUL Architecture
Rfp_EMS_OR_NMS-to-NodeMainDE	NeM	• **Trust and Authentication** exchange of messages and other types of messages exchanges necessary. • **Management COMMANDS** targeting nodes/devices designed following GANA principles. A Manager in the sense of a traditional EMS/NMS, may create a "Wrapper packet/message that encapsulates a COMMAND" e.g. a SET/WRITE COMMAND on a Variable, and send the packet/message to the Node-Main-DE of a node/device where the Node-Main-DE extracts the COMMAND and relays it to the appropriate Function-Level-DE responsible for autonomically managing and controlling the ME targeted by the COMMAND. The DE then reasons about whether to apply the COMMAND, and if yes, the DE executes the COMMAND directly on the ME's management-interface OR issues the COMMAND via the "loopback interface" to the local Management Agent (on the node/device) for execution if the DE manages and controls the ME indirectly through the Management Agent (see note 3).	This case applies to configurations where today's management systems are integrated to co-exist and interwork harmoniously with Network-Level-DEs in the overall management of the network. See clauses 11.7 and 11.8 in ETSI GS AFI 002 [i.15]. See Figure 64 and Figure 68, see also Figure 63 in ETSI GS AFI 002 [i.15]. See also clause 6.4.2.5 in ETSI TS 103 195-2 [i.9].	This is not specific to the WiSHFUL architecture, and there are some solutions that either already exist or can be developed in the industry.
Rfp_ONIX-to-ONIX	FOO	• Domain information (including Domain Type(s), Domain Identifier(s)) exchange that may be conveyed through a local ONIX instance to facilitate for federated AMC.	See clause 6.4.4.5 in ETSI TS 103 195-2 [i.9].	This is not specific to the WiSHFUL architecture, and there are some solutions that either already exist or can be developed in the industry.

Reference Point Name (see note 1)	Alias Name of the Rfp	Characteristic Information communicated over the Reference Point	Additional Comments on where the Reference Point is described	Implementation of the Reference Point in the WiSHFUL Architecture
Rfp_FederationMBTS- to-FederationMBTS	FMM	• Domain information (including Domain Type(s), Domain Identifier(s) exchange between F-MBTS instances that enable two domains to exchange information and control messages for enabling federated AMC across the domains-information and control messages that need to be translated if the domains involved use different data models and information types and formats that are all to be employed in federated AMC.	See clause 6.4.4.6 in ETSI TS 103 195-2 [i.9].	This is not specific to the WiSHFUL architecture, and there are some solutions that either already exist or can be developed in the industry.
Rfp_OSS_to_G-MBTS	GoS	• Input to the Knowledge Plane that is generated by an OSS Tool on the Governance interface (refer to OsI, OsDe Rfps) and is input to G-MBTS for translation into the language, data model(s) and data formats employed by the GANA Knowledge Plane DEs.	See clause 6.4.2.6 in ETSI TS 103 195-2 [i.9].	This is not specific to the WiSHFUL architecture, and there are some solutions that either already exist or can be developed in the industry.

NOTE 1: "**to**" in the name does not mean unidirectional communication only. In some of the cases communication may be initiated by either party and can be bi-directional.
NOTE 2: "GANA-Level2" is also called "Function-Level".
NOTE 3: An alternative to this approach is presented in the corresponding clause in ETSI GS AFI 002 [i.15] where this reference point is defined.
NOTE 4: Annex C in ETSI TS 103 195-2 [i.9] presents one of the ways in which this Reference Point can be implemented, that of using a unified API. Another may the reference point may be implemented is through the traditional approach of using means such as SNMP, by which the GANA Level 3 and Level-2 DEs locally manage and control their MEs through the local SNMP agent and the MIBs implemented for those MEs.
NOTE 5: Rfp_GANA-Level2-AccessToProtocolsAndMechanisms defined in ETSI GS AFI 002 [i.15] is synonymous to this Rfp that considers both GANA Level2 and Level3 access to Protocols And Mechanisms that are on the Resources Layer of a GANA node (NE) (see note 4).

8 Additional Resourceful Information that should be considered by Implementers of GANA DEs

The following Technical White Papers from the 5G PoC [i.16] by ETSI TC INT AFI WG provides very useful additional resourceful information that should be considered by implementers of GANA DEs:

1) White Paper No.1 [i.34]

2) White Paper No.2 [i.14]

3) White Paper No.3 [i.35]

4) White Paper No.4 [i.33]

5) White Paper No.5 [i.36]

6) White Paper No.6 [i.37]

9 Conclusions and Further Work

The present document has provided an Instantiation and Implementation of the ETSI Generic Autonomic Network Architecture (GANA) Model onto Heterogeneous Wireless Access Technologies using Cognitive Algorithms by using the WiSHFUL Framework and Components that have been implemented in a European Commission (EC) - funded WiSHFUL H2020 Project.

Moreover, the present document goes further to illustrate how the concepts developed in another European Commission (EC) - funded project, ORCA, can be applied together with components developed in the WiSHFUL project, to implement ETSI GANA's Multi-Layer Autonomics in Heterogeneous Wireless Access Technologies using Cognitive Algorithms.

Therefore, the document answers the question of how to implement the ETSI GANA model using WISHFUL architecture and ORCA concepts.

The UPIs implemented by WiSHFUL Project can now be adopted by the industry and community at large, and the UPIs can be standardized as an ETSI Technical Specification (TS) for example [i.19] provides the definitions of the UPIs that can be standardized and possibly extended by the industry as may be required into the future.

Regarding Further Work, the following are some of the work that could be performed:

1) A further study can be performed regarding how to implement some of the GANA Reference Points (Rfps) in the WiSHFUL Architecture that have been identified in the present document as subjects for further study.

2) DEs implementers should first produce a table that provides a Mapping of the GANA standardized DEs to their corresponding Managed Entities (MEs) and the associated configurable parameters of the MEs while respecting the 1-to-1 Mapping of DE to an ME Parameter ("1-ME-Param" to "1-DE Mapping") as described in ETSI TS 103 195-2 [i.9]. What this means is that for the environment in which the WiSHFUL framework is to be applied in implementing the GANA a table that is a concretization (instantiation) of Table 3 found in ETSI TS 103 195-2 [i.9] should be produced. The produced table should capture the DE-to-ME mappings such that the GANA standardized DEs that can be implemented in a specific NE/NF type and/or in the GANA Knowledge Plane have their corresponding mappings to the ME and parameters they are supposed to autonomically management and control.

3) Standardization of UPIs by producing an ETSI TS (Technical Specification) that is dedicated to detailing and elaborating all the aspects concerning UPIs.

History

Document history		
V1.1.1	February 2020	Publication