ETSI TR 103 748 V1.1.1 (2022-06)

Core Network and Interoperability Testing (INT);
Artificial Intelligence (AI) in Test Systems and Testing of AI Models;
Use and Benefits of AI Technologies in Testing

Reference
DTR/INT-00166

Keywords
artificial intelligence, testing

ETSI

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

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

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

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

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

Foreword

This 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.

Executive summary

The present document covers the use of Artificial Intelligence technology, mainly Machine Learning, in the context of testing of software systems that are in scope of ETSI. It accounts for the application of these AI techniques covering all testing aspects of the entire lifecycle of a software product, including development and operation. Moreover, it reflects the current industrial practice in this area and reports about new trends in research. The work presented in the present document is a part of a series of documents being produced in the joint work of ETSI TC INT and ETSI TC MTS on "Artificial Intelligence (AI) in Test Systems, and Testing of AI Models". The present document focuses on the subject of Use and Benefits of AI Technologies in Testing. Some of the key outputs created by ETSI in the present document are the following:

- Descriptions of the Benefits brought by "AI Technologies in Testing" to diverse Stakeholders.
- Descriptions of the Benefits brought by "AI Technologies in Testing" to various technical areas of Testing Types, e.g. functional testing, performance testing, integration testing, unit testing, regression testing, etc.
- A new concept of Autonomic Test System (ATS) has emerged in the present document, for testing complex systems that include AI-enabled systems and networks.

- The introduction of a "Desirable Framework for Integration of AI-powered Autonomic Management and Control (AMC) Systems and AI-powered Autonomic Online Test Systems during Network Operations", and adaptation of such a Framework to Testing of Autonomics (AMC) software (powered by AI algorithms) implemented at the realm of network management and control architectures and Autonomics implemented with network infrastructures.

- Outline of the Standardization landscape on the use of AI in testing, and the importance of the European Commission (EC)'s Recommendations on Testing and Certification of AI and the mapping with ETSI TC INT AI-Support System.

The present document concludes with "Outlook on Further Work" that outlines important further items to be addressed by the joint work of ETSI TC INT and ETSI TC MTS, such as: Certification Methods and Approaches for AI Models, Components, Systems and AI-powered Information and Communication Technology (ICT) Networks; Types of Standardizable Metrics for Testing and Certification of AI Models; Building an Ecosystem for Certification Labs for AI models and AI systems and "Testing of AI, with Definitions of Quality Metrics".

Introduction

AI/ML technologies are increasingly being incorporated in the design of various systems and ICT networks as discussed in various sources in literature and in emerging and ongoing standardization activities. The ETSI White Paper No.34 [i.57], presents a good outlook on standardization activities on AI in ETSI and outside of ETSI. Some of the insights are as follows:

1) ETSI addresses **key AI requirements** including:

   - Leverage AI for ICT network optimization.

   - Ensure reliability through testing of systems using AI.

   - Manage and characterize data used by AI, e.g. in IoT.

2) The ETSI work reported in this Report **is aligned** with EC Recommendations on AI and seeks to continue addressing the following aspects (among the various topics of importance to the industry):

   - Standardizable metrics for measurements and assessments in testing/certifying AI models in Autonomic/Autonomous Networks (ANs) and their AMC components.

   - Methodologies and customizable frameworks for test system providers.

   - Support for AI test centers and certification authorities.

At this point in the trends on AI/ML incorporation in systems and ICT network infrastructures, notably in autonomic and autonomous systems and ICT networks (i.e. in the area of Autonomic/Autonomous Networking (ANs) described in ETSI TR 103 747 [i.59] and ETSI TS 103 195-2 [i.1], the topic of "Artificial Intelligence (AI) in Test Systems and Testing of AI Models" is increasingly a "hot" topic because test methodologies, frameworks and standards are critically required by the industry in order to address the question of testing and certification of AI Models, Components, Systems and AI-powered ICT Networks (i.e. ANs). ETSI has produced the de-facto standard for Multi-Layer Autonomics and Multi-Layer AI as a Framework for Autonomic Management and Control (AMC) of Networks and Services, called the Generic Autonomic Networking Architecture (GANA) Reference Model specified in ETSI TS 103 195-2 [i.1]. From ICT networks point of view (the scope of the GANA), the GANA enables to design and implement ANs. The question of Trust and Confidence Building in ANs is another hot topic in the industry now. Testing ANs (and their AI/ML algorithms they employ) is one of the Evaluation Methods of Trust and Confidence Building in AN by Network Operators (e.g. CSPs) and Enterprises that target to deploy and operate ANs. In applying appropriate Test Methods for ANs built based on the ETSI GANA Framework for example, it becomes very important to consider ALL Factors of Multi-Layer Autonomics and Multi-Layer AI and Cross Domain Federation Aspects for ANs as defined by the ETSI GANA Framework in ETSI TS 103 195-2 [i.1].

The present document focuses on the subject of "Use and Benefits of AI Technologies in Testing". A new concept of Autonomic Test Systems has emerged in ETSI for testing complex systems that include AI-enabled systems and networks as described in the present document. As such there is a need for applying autonomics & AI in Test Systems due to rising complexity of the System and Network Under Test (SUT/NUT).

1 Scope

The present document presents a part of a series of documents being produced in the joint work of ETSI TC INT and ETSI TC MTS on "Artificial Intelligence (AI) in Test Systems, and Testing of AI Models". The present document focuses on the subject of "Use and Benefits of AI Technologies in Testing".

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

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

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

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

[i.1] ETSI TS 103 195-2 (V1.1.1): "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.2] Tariq King: "The Current State & Future Trends of AI in Software testing".

NOTE: Available at https://www.perfecto.io/blog/ai-in-software-testing.

[i.3] T. M. King, J. Arbon, D. Santiago, D. Adamo, W. Chin and R. Shanmugam: "AI for Testing Today and Tomorrow: Industry Perspectives", 2019 IEEE International Conference On Artificial Intelligence Testing (AITest), Newark, CA, USA, 2019, pp. 81-88.

[i.4] Paul Merrill: "AI in testing: 13 essential resources for QA pros".

NOTE: Available at https://techbeacon.com/app-dev-testing/ai-testing-13-essential-resources-qa-pros.

[i.5] Joe Colantonio: "8 Innovative AI Test Automation Tools for the Future: The Third Wave".

NOTE: Available at https://testguild.com/7-innovative-ai-test-automation-tools-future-third-wave/.

[i.6] Testnet: "Testing with Artificial Intelligence"; TestNet workgroup 'Testing and AI'; Whitepaper, version 1.0 - May 2019.

NOTE: Available at https://www.testnet.org/testnet/download/werkgroep-testen-met-ai/testing-with-artificial-intelligence-v1.0..pdf?1558371859.

[i.7] ETSI EG 203 647: "Methods for Testing and Specification (MTS); Methodology for RESTful APIs specifications and testing".

[i.8] ETSI TC INT AFI WG 5G PoC White Paper No. 5: "Artificial Intelligence (AI) in Test Systems, Testing AI Models and ETSI GANA Model's Cognitive Decision Elements (DEs) via a Generic Test Framework for Testing GANA Multi-Layer Autonomics & their AI Algorithms for Closed-Loop Network Automation".

NOTE: Available at https://intwiki.etsi.org/images/ETSI_5G_PoC_White_Paper_No_5.pdf.

[i.9] Altran White Paper (published in 2019): "Autonomic Test Automation for Total Testing in the Digital Era: Extending Test Automation Approaches to Newer Technology Domains".

[i.10] Böhme, M., Pham, V. T., Nguyen, M. D., & Roychoudhury, A. (2017, October): "Directed greybox fuzzing". In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 2329-2344).

[i.11] Böttinger, K., Godefroid, P., & Singh, R. (2018, May): "Deep reinforcement fuzzing". In 2018 IEEE Security and Privacy Workshops (SPW) (pp. 116-122). IEEE.

[i.12] Kuznetsov, A., Yeromin, Y., Shapoval, O., Chernov, K., Popova, M., & Serdukov, K. (2019, July): "Automated software vulnerability testing using deep learning methods". In 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON) (pp. 837-841). IEEE.

[i.13] Elbaum et al. (2014): "Techniques for improving regression testing in continuous integration development environments", in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York, NY, USA: Association for Computing Machinery (FSE 2014), pp. 235-245. doi: 10.1145/2635868.2635910.

[i.14] Khatibsyarbini et al. (2018): "Test case prioritization approaches in regression testing: A systematic literature review", Information and Software Technology, 93, pp. 74-93. doi: 10.1016/j.infsof.2017.08.014.

[i.15] Marijan et al. (2013): "Test Case Prioritization for Continuous Regression Testing: An Industrial Case Study", in 2013 IEEE International Conference on Software Maintenance. 2013 IEEE International Conference on Software Maintenance, pp. 540-543. doi: 10.1109/ICSM.2013.91.

[i.16] De Castro-Cabrera et al. (2020): "Trends in prioritization of test cases: 2017-2019", in Proceedings of the 35th Annual ACM Symposium on Applied Computing. New York, NY, USA: Association for Computing Machinery (SAC '20), pp. 2005-2011. doi: 10.1145/3341105.3374036.

[i.17] Ali et al. (2019): "On the search for industry-relevant regression testing research", Empirical Software Engineering, 24(4), pp. 2020-2055. doi: 10.1007/s10664-018-9670-1.

[i.18] ETSI White Paper No.16: "The Generic Autonomic Networking Architecture Reference Model for Autonomic Networking, Cognitive Networking and Self-Management of Networks and Services".

NOTE: Available at http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp16_gana_Ed1_20161011.pdf.

[i.19] High-Level Expert Group on Artificial Intelligence: "Ethics Guidelines for Trustworthy AI", 2019.

NOTE: Available at https://ai.bsa.org/wp-content/uploads/2019/09/AIHLEG_EthicsGuidelinesforTrustworthyAI-ENpdf.pdf.

[i.20] Bainczyk, A., Schieweck, A., Steffen, B., & Howar, F. (2017): "Model-based testing without models: the TodoMVC case study". In ModelEd, TestEd, TrustEd (pp. 125-144). Springer, Cham.

[i.21] Ernst, M. D., Perkins, J. H., Guo, P. J., McCamant, S., Pacheco, C., Tschantz, M. S., & Xiao, C. (2007): "The Daikon system for dynamic detection of likely invariants". Science of computer programming, 69(1-3), 35-45.

[i.22] Feng, L., Lundmark, S., Meinke, K., Niu, F., Sindhu, M. A., & Wong, P. Y. (2013, November): "Case studies in learning-based testing". In IFIP International Conference on Testing Software and Systems (pp. 164-179). Springer, Berlin, Heidelberg.

[i.23] Herbold, S., & Harms, P. (2013, March): "AutoQUEST--automated quality engineering of event-driven software". In 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (pp. 134-139). IEEE.

[i.24] McMinn, P. (2004). Search-based software test data generation: a survey. Software testing, Verification and reliability, 14(2), 105-156.

[i.25] Meinke, K., & Sindhu, M. A. (2011, June): "Incremental learning-based testing for reactive systems". In International Conference on Tests and Proofs (pp. 134-151). Springer, Berlin, Heidelberg.

[i.26] Meinke, K., & Sindhu, M. A. (2013, March): "LBTest: a learning-based testing tool for reactive systems". In 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation (pp. 447-454). IEEE.

[i.27] Zalewski, M. (2014): "American fuzzy lop". [as of 2020-06-18].

NOTE: Available at https://lcamtuf.coredump.cx/afl/.

[i.28] Bekrar, S., Bekrar, C., Groz, R., & Mounier, L. (2012, April): "A taint based approach for smart fuzzing". In 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation (pp. 818-825). IEEE.

[i.29] DeMott, J., Enbody, R., & Punch, W. F. (2007): "Revolutionizing the field of grey-box attack surface testing with evolutionary fuzzing". BlackHat and Defcon.

[i.30] Duchene, F., Rawat, S., Richier, J. L., & Groz, R. (2014, March): "KameleonFuzz: evolutionary fuzzing for black-box XSS detection". In Proceedings of the 4th ACM conference on Data and application security and privacy (pp. 37-48).

[i.31] ETSI TR 101 583 (V1.1.1): "Methods for Testing and Specification (MTS); Security Testing; Basic Terminology".

[i.32] Godefroid, P., Peleg, H., & Singh, R. (2017, October): "Learn&fuzz: Machine learning for input fuzzing". In 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 50-59). IEEE.

[i.33] Rajpal, M., Blum, W., & Singh, R. (2017): "Not all bytes are equal: Neural byte sieve for fuzzing". arXiv preprint arXiv:1711.04596.

[i.34] Saavedra, G. J., Rodhouse, K. N., Dunlavy, D. M., & Kegelmeyer, P. W. (2019): "A review of machine learning applications in fuzzing". arXiv preprint arXiv:1906.11133.

[i.35] She, D., Pei, K., Epstein, D., Yang, J., Ray, B., & Jana, S. (2019, May): "NEUZZ: Efficient fuzzing with neural program smoothing". In 2019 IEEE Symposium on Security and Privacy (SP) (pp. 803-817). IEEE.

[i.36] Sparks, S., Embleton, S., Cunningham, R., & Zou, C. (2007, December): "Automated vulnerability analysis: Leveraging control flow for evolutionary input crafting". In Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007) (pp. 477-486). IEEE.

[i.37] Yan, G., Lu, J., Shu, Z., & Kucuk, Y. (2017, August): "Exploitmeter: Combining fuzzing with machine learning for automated evaluation of software exploitability". In 2017 IEEE Symposium on Privacy-Aware Computing (PAC) (pp. 164-175). IEEE.

[i.38] Grano, G. et al. (2019): "Branch coverage prediction in automated testing", Journal of Software: Evolution and Process, 31(9), p. e2158. doi: 10.1002/smr.2158.

[i.39] Fraser, G. et al. (2015): "Does Automated Unit Test Generation Really Help Software Testers? A Controlled Empirical Study", ACM Transactions on Software Engineering and Methodology, 24(4), p. 23:1-23:49. doi: 10.1145/2699688.

[i.40] Almasi, M. M. et al. (2017): "An Industrial Evaluation of Unit Test Generation: Finding Real Faults in a Financial Application", in 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP). 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), pp. 263-272. doi: 10.1109/ICSE-SEIP.2017.27.

[i.41] Grano, G. et al. (2019): "Branch Coverage Prediction in Automated Testing". doi: 10.1002/smr.2158.

[i.42] Watson, C. et al. (2020): "On learning meaningful assert statements for unit test cases", in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. New York, NY, USA: Association for Computing Machinery (ICSE '20), pp. 1398-1409. doi: 10.1145/3377811.3380429.

[i.43] ETSI TR 103 629: "Evolution of Management towards Autonomic Future Internet (AFI); Confidence in autonomic functions; Guidelines for design and testability". TC INT Work on Trust and Confidence Building in Autonomic/Autonomous Networks (ANs).

NOTE: Available at https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=54807.

[i.44] White Paper by KEYSIGHT Technologies, PATHWAVE: "The TestOps Manifesto A Blueprint for Connected", Agile Design and Test: March 19, 2019.

[i.45] Subhadeep Chakraborty: "How to use AI/ML Systems to Revolutionize Testing and Test automation?", In International Journal of Scientific & Engineering Research Volume 10, Issue 1, January-2019 290, ISSN 2229-5518.

[i.46] Jason Arbon, White Paper by Test.ai: "The QA Director's Guide to TestOps", February 1, 2021.

[i.47] Helge Spieker, Arnaud Gotlieb, Dusica Marijan, Morten Mossige: "Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration", In Proceedings of 26th International Symposium on Software Testing and Analysis (ISSTA'17) (pp. 12--22), 2017. ACM: doi: 10.1145/3092703.3092709.

[i.48] Alex Groce, et al: "Learning-Based Test Programming for Programmers", October 2012: doi: 10.1007/978-3-642-34026-0-42.

[i.49] Miroslav Bures: "Habilitation Thesis: Model-based Software Test Automation", Czech Technical University in Prague Faculty of Electrical Engineering: 2017.

[i.50] Srinivas Pinisetty, Thierry Jéron, Stavros Tripakis, Yliès Falcone, Hervé Marchand, et al.: "Predictive Runtime Verification of Timed Properties". Journal of Systems and Software, Elsevier, 2017, 132, pp.353 - 365. ff10.1016/j.jss.2017.06.060ff. ffhal-01666995.

[i.51] Reza Babaee Cheshmeahmadrezaee (2019): "Predictive Runtime Verification of Stochastic Systems". UWSpace. A thesis presented to the University of Waterloo in fulfilment of the thesis requirement for the degree of Doctor of Philosophy in Electrical and Computer Engineering: Waterloo, Ontario, Canada, 2019.

NOTE: Available at http://hdl.handle.net/10012/14876.

[i.52] Stefan Wagner: "A Bayesian Network Approach to Assess and Predict Software Quality Using Activity-Based Quality Models", In ScienceDirect Journal: Information and Software Technology Volume 52, Issue 11, November 2010, Pages 1230-1241.

NOTE: Available at https://doi.org/10.1016/j.infsof.2010.03.016.

[i.53] Andrea Baldini, Alfredo Benso, Paolo Prinetto: "A Dependable Autonomic Computing Environment for Self-Testing of Complex Heterogeneous Systems", Elsevier, Electronic Notes in Theoretical Computer Science 116 (2005) 45-57: doi:10.1016/j.entcs.2004.02.087.

[i.54] Sam Brooks & Rajkumar Roy (2021): "An overview of self-engineering systems", Journal of Engineering Design, doi: 10.1080/09544828.2021.1914323.

[i.55] White Paper No.3 of the ETSI 5G PoC: "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/index.php?title=Accepted_PoC_proposals.

[i.56] ETSI TR 103 763: "Core Network and Interoperability Testing (INT); Description of Test Requirements and Approach for E2E Federated Testbeds".

NOTE: Available at https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=59577.

[i.57] Frost Lindsay, et al: ETSI White Paper No. #34: "GANA - Generic Autonomic Networking Architecture", 1st edition - June 2020, ISBN No. 979-10-92620-30-1.

[i.58] Meriem Tayeb Ben, et al: ETSI White Paper No. #16: "Artificial Intelligence and future directions for ETSI, Reference Model for Autonomic Networking, Cognitive Networking and Self-Management of Networks and Services", 1st edition - October 2016, ISBN No. 979-10-92620-10-8.

[i.59] ETSI TR 103 747 (V1.1.1): "Core Network and Interoperability Testing (INT/ WG AFI); Federated GANA Knowledge Planes (KPs) for Multi-Domain Autonomic Management & Control (AMC) of Slices in the NGMN(R) 5G End-to-End Architecture Framework".

[i.60] IEEE 1633™: "IEEE Recommended Practice on Software Reliability".

[i.61] ISO/IEC/IEEE 24765: "Systems and software engineering -- Vocabulary".

3 Definition of terms, symbols and abbreviations

3.1 Terms

3.1.1 Common Terms in Artificial Intelligence

- Artificial Intelligence (AI).
- Artificial Intelligence Model.
- Machine Learning (ML).
- ML Model.
- Model Inference.
- Natural Language Processing (NLP).
- Reinforcement Learning.
- Supervised Learning.
- Unsupervised Learning.

3.1.2 Terms in network operation and testing

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

Autonomic Test System (ATS): Test System that employs a Closed Control-Loop concept in adaptively testing and (re)-tuning the configuration of its associated System Under Test (SUT)

NOTE 1: The feedback received by the ATS from the SUT, coupled with information from the SUT's operational environment and any other sources that may be relevant is continuously used in intelligently composing new Test cases or in selectively executing Test Cases over the SUT.

NOTE 2: The Autonomic Test System itself exhibits properties of an autonomic system such as self-configuration, self-diagnosing, self-healing, self-protecting, self-optimization, and other self-X features.

AI-powered Autonomic Management and Control (AMC) System

NOTE: An Autonomic Management and Control (AMC) System such as the ETSI GANA Knowledge Plane (KP) Platform is always powered by AI, hence wherever there is talk of an Autonomic Management and Control (AMC) System it is an AI-powered Autonomic Management and Control (AMC) System.

3.2 Symbols

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

CNif	CoNfiguration interface
MRif	Monitoring and Retrieval interface
TEif,	Test Execution interface

3.3 Abbreviations

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

AFI	Autonomic Future Internet
AFL	American Fuzzy Lop
AI	Artificial Intelligence
AI-SS	AI Support System
AMC	Autonomic Management and Control
AN	Autonomic Network
API	Application Programming Interface
ATS	Autonomic Test System
BNG	Broadband Network Gateway
BSS	Business Support System
CertaaS	Certification as a Service
CI	Continuous Integration
CSP	Communication Service Provider
CSS	Call Stability Score
CUT	Component Under Test
DC	Data Center
DE	Decision Element
E2E	End 2 End
GANA	Generic Autonomic Networking Architecture
GUI	Graphical User Interface
ICT	Information and Communication Technology
IoT	Internet of Things
IP	Internet Protocol
ISO	Intrnational Standard Organisation
ISTQB	International Software Testing Qualifications Board
ISV	Independent Software Vendor
IT	Information Technology
KP	Knowledge Plane
KPI	Key Performance Indicator
LSTM	Long Short Term Memory
ML	Machine Learning
NE	Network Element
NF	Network Function
NFV	Network Functions Virtualisation
NLP	Natural Language Processing
NMT	Neural Machine Translation
NUT	Network Under Test
OPEX	Operational Expenditure
OSS	Operation Support System
PoC	Proof of Concept
QoE	Quality of Experience
QoS	Quality of Service
RegaaS	Regulation as a Service
RTS	Regression Test Selection
SBA	Service Based Architecture
SDN	Software Defined Networks
SDO	Standard Development Organization
SUT	System Under Test
TCM	Test Case Minimization

TCP	Test Case Prioritization
TCS	Test Case Selection
TesaaS	Testing as a Service
TR	Technical Report
UI	User Interface
VNF	Virtual Network Function
WG	Working Group

4 Benefits of AI to testing stakeholders

4.1 Overview

This clause provides a discussion of AI business cases in the area of Testing.

The introduction of AI in testing is expected to impact the testing process in several ways all contributing to more comprehensive and efficient testing improving the quality of telecommunication software systems. The AI-based testing techniques will affect current stakeholders in the testing process differently. The testing stakeholders are hence categorized in the categories: test engineers, test solution providers and network and software suppliers, to identify how they specifically may benefit from the use of AI in the software testing process.

4.2 The need for innovation in testing

Existing testing stakeholders are faced with several urgent issues such as automating still existing current manual tests and widely deploy test generation techniques.

A common problem of user interface testing is that it is slow, brittle and requires a lot of maintenance. This is due to the changing interface and the rigid methods used today to identify the objects, which also disallow automatic identification of new objects. AI can learn the properties of object groups (partly from previous test campaigns, which are used as the training data) and when an object changes or new objects are introduced, it can still identify it with a given certainty.

Test case selection using results analysis from previous test executions is another area where further innovation is needed. The test case selection process is described in more detail in clause 5.8. The test result analysis may include clustering of test results and trouble tickets. This can be the initial step for a more sophisticated fault classification. Using the log files from the tests performed during product development may enable further refinement of the most typical fault causes. In production, the test result analysis also allows to discover the most important use cases, and to cluster the users into a number of categories based on their actions.

This fault classification may then be used to support the localization of faults and provide input to the test case selection task.

4.3 Benefits to test engineers

This clause discusses the impact of AI on "test engineers" as a general umbrella term, and attempts to answer questions like "Does the introduction of AI in Test Systems make their life easier, how?" Such benefits need to be looked at from the broad perspective of addressing the entire test process and test activities and roles: modelling, designing, implementation, execution, analysing, certifying systems.

Of relevance to consider in this respect is also the question of what the testing community views as the Evolution of testing, in terms of how existing activities change, new activities, interfacing of test engineers with other stakeholders and what is changing as illustrated by the diagram on "marketplace" presented in the ETSI PoC White paper No. 5 [i.8] (in figure 5, p. 22 of [i.8]).

NOTE: For Further consideration in the evolution of the present document: The provision of a Summarizing Table that presents stakeholders vs current work and summary of benefits from AI (impact). Readers should look forward to the availability of this item in the evolved version of the present document.

An example of an area where AI in Testing brings benefit to test engineers is the area of Test Case Generation. Test case generation can be enhanced using the following technologies:

- Natural Language Processing (NLP) methods can help in analysis of documentation, which are typical input to test description and test case design, such as a requirement, functional or technical design documents [i.6].
- A specific area is the emerging use of RESTful APIs, where the API specification is done by means of semi-formal description languages and techniques [i.7]. By training the AI system using other API specifications and corresponding test descriptions (and test cases if such exist), the AI system can be able to extract test cases for the new specification.
- Objects recognition in GUI testing (e.g. web element location) [i.6].

4.4 Benefits to test solution providers

4.4.1 Overview

Generally speaking, test solution providers are stakeholders which supply test solutions in terms of tools and services to other stakeholders which benefit from the test solution(s), e.g. customers that then apply the solution(s) in testing their components, systems, or networks such as ICT networks.

Benefits that ML/AI can bring to test solution providers can be categorized in two forms:

- Methods and features increasing their customer's satisfaction, thus strengthening their market position. This is more often related to increasing efficiency of the different steps of the testing process, from test specification to test result evaluation.
- New business opportunities by providing solutions to new, ML/AI enabled systems, including autonomous systems; further details on this is described in clause 7.2.

AI algorithm suppliers may build new business relationships with test solution suppliers such that they may procure such algorithms and use them in their test solutions for AI-powered test systems. In addition, Open-Source AI related projects may have an impact as well on the business models of test solution suppliers/providers.

4.4.2 Suppliers of new types of test systems

During the last few years several (many cases new) vendors started providing AI-driven or AI assisted testing. Majority of these tools targeting system-level testing of mobile applications or Web UI applications [i.2] and [i.5], like functional testing of web and mobile applications, visual testing of user interfaces, UI element location and auto-correcting element selectors, etc. These solutions typically use autonomous intelligent agents, often called "test bots", to automate activities such as application discovery, modelling, test generation, and failure detection. A combination of different machine learning techniques is used to implement test bots. These include but are not limited to decision tree learning, neural networks, and reinforcement learning.

Examples of AI-driven testing approaches that have formed over the last decade include [i.2]:

- Differential testing - comparing application versions overbuilds, classifying the differences, and learning from feedback on the classification.
- Visual testing - leveraging image-based learning and screen comparisons to test the look and feel of an application.
- Declarative testing - specifying the intent of a test in a natural or domain-specific language, and having the system figure out how to carry out the test.
- Self-healing automation - auto-correcting element selection in tests when the UI changes.

Network virtualization that will be accomplished by the deployments of 5G networks - especially when introducing network slicing - creates new opportunities for vendors providing testing tools and services to network operators, specifically in the area of network management.

NOTE 1: Perspectives covered in clause 5 complement what has been covered in this clause.

It is likely that new types of test systems emerge as a result of deploying AI/ML technology to handle the testing of autonomous and self-adaptive systems in the future.

For example, this means traditional suppliers/providers of test solutions may benefit from such opportunities by providing solutions for AI-powered Test Systems and/or Autonomic Test Systems, or new kinds of Test Solutions Suppliers may emerge which may be focused on supplying AI-powered Test Systems (non-autonomic or autonomic) only.

NOTE 2: For a full description of AI-powered Systems further details may be found in the following clauses, and also in clause 8 with respect to the subject of Autonomic Test Systems powered by AI/ML.

4.5 Benefits to network infrastructure and software suppliers

4.5.1 Overview

This clause covers benefits of AI and ML in testing to stakeholders which are Communication Service Providers (CSPs) and enterprises that own and operate ICT infrastructures.

There are three aspects (contexts) to be considered regarding the benefits of AI in test systems from the perspective of CSPs and Enterprises Networks (CSPs and Enterprises as Stakeholders):

- The need for test systems that leverage AI in testing existing communication networks and services provisioning technologies that are already in deployments in CSPs and enterprise environments.
- The need for test systems that leverage AI in testing emerging networks like 5G (with its enablers like SDN, NFV, AMC (Autonomic Management and Control), Orchestration, etc.) and associated services that can be delivered by 5G.
- Benefits of AI in test systems from the perspective of Network Operations' desirable framework for integration of AI-powered Autonomic Management and Control (AMC) systems and AI-powered autonomic online test systems. Test Systems that may be made to exhibit the property of being AI-powered autonomic online test systems could be the very test systems that are used for objectives such as testing of services at instantiation time (e.g. at instantiation of a network service like VNF or new service chains), internal VNF certifications, or orchestrated assurance of newly instantiated (on boarded) services.

4.5.2 AI for testing existing networks and services

This clause covers the value of AI in the context of Test systems that leverage AI in testing existing communication networks and services provisioning technologies that are already in deployments in CSPs and enterprise environments.

The challenges and opportunities associated with this context are as follows: Existing communication networks and services provisioning technologies t that are already in deployments in CSP and Enterprise Environments are expected to last long. Yet testing them using traditional test approaches (both in the testbeds and in production environments) has not been able to expose certain behaviours that need to be captured and understood in order to tune the networks and service parameters to cater for such behaviours. [i.8] and [i.9] describe examples of the "Benefits of AI in Test Systems" that are relevant to testing such existing communication networks. Services of AI-empowered test systems for such a context should exhibit as follows: The AI-powered test systems should be able to expose certain network or service behaviours that are hard to capture and understand using traditional testing solutions that do not employ AI. Other desirable functionalities of AI-empowered test systems for this context include the functional aspects described in sources such as [i.8] and [i.9] and may also include those functional features attributed to an Autonomic Test System defined and described in the present document.

4.5.3 AI for testing emerging networks and services

This clause covers Test systems that leverage AI in testing emerging networks like 5G and associated services that can be delivered by 5G.

CSPs are going through a transformation towards intensified software'rization, manifested in some trends such as the following:

- Service Based Architecture (SBA) as specified by the 3GPP and other SDOs.

- Standalone 5G with a full software-based and 5G enabled core, paving the way for end-to-end slicing.
- XaaS or Everything as a Service, where both functional features such as bandwidth, mobility, roaming and non-functional features such as security and QoS/QoE are modelled as a service.

Furthermore, the need for scalability and flexibility in combining/bundling software services and offering them in turnkey form, increases the pressure on CSPs. Scalability and flexibility are also key enablers when there is a need for federating network assets and capabilities from various CSPs to extend existing footprint to allow Application Providers to be on boarded along with their applications to be reachable by a larger customer base without geographical limitation or for a targeted geographical area. This federation could also apply at service or product level as well for composing various software pieces supplied by different CSPs to deliver composite services to customers. The aggregation of federating network assets and capabilities may be done by a single CSP or independent 3rd party enterprise.

At the same time, this transformation and the requirements and challenges it brings along causes an OPEX increase and a time challenge for CSPs when adopting new software-intensive models and deploying software-based modules.

This is where AI and ML come into the scene as a powerful and often essential leverage instrument for CSPs and their ecosystems as CSP communities they try to build through various federation models in order to be able to compete in the new hyper scale environment.

Potential ways in which those trends can help CSPs tackle their challenges are, among others:

- Structuring of massive test-case variants into a hierarchical or group model for more efficient execution and result evaluation; using ML techniques such as pattern recognition and data mining.
- Automating a test sequence or series of testing flows based on the software architectural structure of the use case scenario (e.g. the matrix of software functions and common resources they share or access in an SBA scenario); here basic AI, principally autonomics, is a strong leverage for automating the procedures and collecting, consolidating and cascading their results.
- Federated Testbeds (Testbeds that are interconnected in such a way that they can be used seamlessly as a whole):
  - White Paper [i.8] presents a case for the need of Federated Testbeds in the Testing Federated ETSI GANA Knowledge Plane (KP) Platforms that are AI powered and drive Autonomic Management & Control (AMC) operations over various network segments and across administrative domains (e.g. across multiple network operators).
  - ETSI TR 103 763 [i.56] also presents ongoing work in this respect.

The nature and desirable functionality that AI-empowered test systems for this context should exhibit are as follows: The AI-powered Test System should be able to expose certain network or service behaviours that are hard to capture and understand using traditional testing solutions that do not employ AI. Other desirable functionalities of AI-empowered test systems for this context include the functional aspects described in sources such as [i.8] and [i.9] and may also include those functional features attributed to an Autonomic Test System defined and described in the present document. It can be noted that these requested properties are similar to those identified for AI-empowered test systems for Communication Networks and Services Provisioning Technologies that are already in deployments in CSPs and Enterprise Environments.

4.5.4 AI for testing in CSP autonomic network infrastructures and autonomic management & control systems environments

In environments of CSPs or network operators, AI in Autonomic Management and Control (AMC) software needs to interwork with AI in autonomic online test systems. These online test systems should have the ability to trigger on-demand monitoring (e.g. on-demand monitoring of services or resources in the network). This means, an autonomic online test system should be able to communicate the on-demand monitoring requirements to the autonomic monitoring functionality implemented by the AMC System(s) of relevance to achieving the dynamic monitoring needs.

NOTE: The concept of an AI-powered autonomic online test system is described in more detail in clause 7.1 on the emergent concept of an autonomic test system.

4.6 Benefits to network infrastructure and software suppliers

The perspectives covered in this clause pertain to the emerging types of ICT network infrastructures that are characterized as autonomic, autonomous, intelligent network infrastructures (see ETSI White Paper No.16 [i.58] and ETSI TS 103 195-2 [i.1], for example). Such intelligent networks call for the need for AI-powered test systems for testing them. This aspect is elaborated in clause 8 (technical clause).

There are four aspects to be considered regarding the benefits of AI in Test Systems from the perspective of network infrastructure and software suppliers such as Independent Software Vendor (ISV) who supply solutions for autonomic network and service management m and control systems. The four aspects are described below.

NOTE: The aspects described below are complemented by further details provided in clause 7 (Application cases of AI/ML in testing). The elaboration covers concepts, such as AI-powered Autonomic Test System or simply Autonomic Test System, AI-powered AMC Systems, and CSPs' Desirable Framework for Integration of AI-powered AMC Systems and AI-powered Autonomic Online Test Systems.

More details on how network infrastructure and software suppliers benefit from AI & ML in Test Systems are provided in clause 7.1.3. The four types of benefits for this category are:

1) The need for test systems that leverage AI in testing existing communication networks and services running in the infrastructure of a CSP or enterprises. This perspective and associated benefits of AI in test systems is common for CSPs and network infrastructure suppliers and for management and control systems (e.g. OSS/BSS systems) suppliers as well. They benefit from all test results that a network owner or operator (a CSP for example) may share with them. The insights that can be derived from the test results may help to improve the products used in existing (e.g. already deployed) communication networks and services. The benefits are the same, independently of whether the AI-powered testing is conducted in a test lab environment or in a production network environment.

2) The need for test System(s) that leverages AI in testing emerging networks like 5G with its enablers SDN, NFV, AMC, Orchestration, etc. and associated services.

3) Benefits of AI in test systems from the perspective of network infrastructure and software suppliers' desirable framework for integration of AI-powered systems and AI-powered autonomic test systems that are meant to be used in a test lab. In reference to an AI-powered autonomic test system that is meant to be used in a test lab to test an AMC System for a scenario in which the targeted autonomics software (autonomic manager components, e.g. ETSI GANA model Decision-making Elements (DEs)) to be tested by the AI-powered Test system is only in the AMC system and not in the underlying network infrastructure. Test-systems that may be made to exhibit the property of being AI-powered (and even autonomic test systems) could be the same test systems that are used for objectives such as testing of services at instantiation time (e.g. at instantiation of a network service like VNF or a new service chain), internal VNF certifications, or orchestrated assurance of newly instantiated (on boarded) services.

4) Benefits of AI in test systems from the perspective of network infrastructure and software suppliers' desired framework for integration of AI-powered AMC systems and AI-powered autonomic test systems that are meant to be used in a test lab. AI-powered autonomic test systems test an AMC system for a scenario, in which the targeted autonomics software (autonomic manager components, e.g. ETSI GANA Model Decision-making Elements (DEs)) to be tested by the AI-powered autonomic test system is the high level ("macro level") autonomics in the AMC system and the low level ("micro level") autonomics software in certain Network Elements/Network Functions (NEs/NFs) of the underlying network infrastructure. The autonomic software at the two levels should be first tested separately and then together in their interactions. As mentioned earlier, test systems that may be made to exhibit the property of being AI-powered (and even autonomic) test systems could be the same test systems that are used for objectives such as testing of services at instantiation time chain.

5 AI technologies applied to software testing

5.1 Test derivation in functional testing

5.1.1 Overview

This clause reflects current (known) state of practice in software testing; discusses various AI/ML technologies and their known application for software testing types and testing activities.

Some of the aspects for consideration are:

- AI for test derivation, e.g. evolutionary testing and further search-based testing techniques, intelligent concolic testing to reach code coverage;
- model inference from existing traces (test logs) to derive new tests.

Functional testing is the process of testing the SUT for functional suitability by deriving test cases from the functional requirements. A set of standard methods such as equivalence partitioning and boundary value analysis are available to guide the process of test derivation. Earlier approaches employ search algorithms, such as hill climbing, simulated annealing, or evolutionary algorithms, to generate test data from specifications [i.24].

Automated test derivation often relies on models that represent the behaviour of the SUT and derive test cases from these models. Building such models from requirements is a cost-intensive process that poses a significant impediment for starting automated derivation. This is an area in which AI helps to create such models from traces, e.g. existing test cases [i.26], such as incremental learning [i.25] or usage of a system [i.23]. Model-based testing without models, also known as learning-based testing [i.20], is a combination of model inference through active automated learning and model checking. Tests are automatically derived by applying model checking techniques to explore the inferred state machine. The approach also automates the test evaluation step by comparing the expected output sequences derived from the inferred state machine with the observed sequences from test case execution and thus, provides an automated test oracle.

A third topic is identification of bugs through detection of invariant violation. In this area, AI can automate the expensive step of invariant formalization. Daikon's invariant detection [i.21] constitute an unsound test oracle that gathers likely invariants from execution traces. These can be used in functional testing to detect failures.

5.1.2 Industrial Relevance

Search-based approaches for functional test derivation often suffer from the need to specify a fitness function. This could be a difficult task for functional testing since functional tests require a formal specification so that they can be at least partially generated. The fitness function is the crucial factor of many search-based approaches, and formalizing functional requirements adequately could be its own art.

Learning-based testing is often not feasible for systems of industrial size. To cope with this scalability problem, an incremental learning-based approach has been developed [i.26]. Incremental learning-based testing has been applied in a few industrial case studies, e.g. a web shop, a brake-by-wire embedded system for cars, and a financial portfolio product [i.22]. The financial portfolio product has been tested by a test engineer with no experience in learning-based testing. This suggests that learning-based testing could be transferable to the industry. However, the approach still suffers from efficiency due to the large number of test cases that are generated and the need to model requirements.

Since Daikon's invariant detection constitute an unsound test oracle, it is not a reliable source for fault detection in functional testing. Furthermore, its applicability is limited since invariant detection is computationally expensive and requires limiting the invariant detection to parts of a program and the observed variables [i.21].

5.2 Performance testing

5.2.1 Overview on challenges and trends in performance testing

The White paper [i.8] discusses the status for performance measurements and tools today. It argues that performance requirements are in many cases far from as precisely specified as functional requirements, if specified at all, that furthermore, test cases for system performance measurements are very complex and time consuming to create, and that the test case specifications vary greatly depending on the kind of system performance characteristics that should be measured. The White paper [i.8] also cites the following examples of variations:

- SUT conditions that should apply when system performance measurement data are captured.
- Selection of SUT services that should be used in the performance measurements.
- The required execution time of a test case.
- The required number of simulated users.
- Requested combinations of application protocols, transport protocols, and network protocols.
- Selection of measurement data that should be collected to deliver requested system performance characteristics.
- Specification of transaction data needed to make service requests from simulated users look different.

Other aspects discussed in this White paper [i.8] are:

- Enabling simple and fast test case creation and possibilities to execute performance measurements for every system build is, however, not all. In order to monitor the changes of system performance from build to build, performance measurement results from each build should be evaluated and compared with stated performance requirements and earlier measurement results. This requires a performance measurement database where all collected measurement data from every test run are stored. Other targets for improvements of performance measurement tools are test productivity. System performance tests are generally regarded as costly activities in terms of costs of test tools, costs of required test equipment for SUT and test tools, costs of manpower required for training, creation of test cases, test runs, and evaluation of test results.
- Requirements on performance test tools, including other requirements outside the performance test tools (e.g. the need for a generally accepted syntax for all aspects of performance requirement specifications that enable automated compilation into test cases).

5.2.2 Prerequisites for building AI into performance test tools

The White paper [i.8] argues that the following four requirements from the list provided in [i.8] are prerequisites to enable system performance measurements at the same pace as functional tests and thus do not require AI per-se, but they are also prerequisites for building AI into performance test tools:

- Performance test tools should have the capability to act on signals from development cycles such as system builds. This is a requirement that can be resolved simply with event messages sent from the system build process at the end of a build to the performance measurement system.
- Performance test tools should have the capability to start and run system performance tests autonomously. This requirement does not require AI but is needed for activities initiated by AI.
- Performance test tools should have access to a database for performance measurement data with a granularity in terms of performance of single services offered by the SUT. This is another prerequisite to AI in performance measurement tools.
- Performance test tools should have the ability to quickly perform capacity measurements of individual services of an SUT. This is a requirement to run performance measurements in parallel with functional tests of system builds. This requirement also enables performance measurements of all permutations of two services to detect hidden dependencies that have an impact on system capacity.

5.2.3 The value AI brings in performance test tools

There are reports in literature on Test frameworks that leverage AI in Test Systems, and also on autonomics (closed loops) introduced in Test Systems to make them intelligent and adaptive in the way they continuously perform Testing in DevOps and Agile products development and testing environments. For example, [i.9] presents the concept of Autonomic Test Automation, which according to [i.9] is defined as "an innovative new approach that uses artificial intelligence, machine learning, and test analytics for automated test design, execution, optimization and maintenance of the automation framework". Quoting [i.9], "the benefits of AI and autonomics in Test Systems include self-adaptation, increased test operational efficiency, improved test coverage and extended duration test cycles with fail-safe recovery, and that also, Artificial-Intelligence-driven test analytics can be used to enrich the test model using machine-learning techniques on large sets of historical and production data".

The White paper [i.8] presents the following five requirements on what would enable high level of automation of performance measurements with help of AI built into the test tools:

- Performance test tools should be capable of translating specifications of performance test cases expressed in terms of what characteristics should be measured and for what SUT services into traditional specifications of performance test cases. This requirement will reduce time and manpower to a minimum and lower the requirements on testers.

- Performance test tools should have the ability to evaluate system performance results autonomously. This requirement will further reduce time and manpower required for performance measurements. A generally accepted syntax for all aspects of performance requirements is, however, a prerequisite for high quality performance evaluations.

- Performance test tools should have the ability to deliver system performance results autonomously. This requirement will also reduce time and manpower required for performance measurements. Together with requirements the other requirements this requirement is the last piece in what is required of fully autonomous performance measurements.

- Performance test tools should have the capability to monitor test runs regarding bad performance trends and abort the job if such situations are detected. This is case where AI in performance tools will pay off and save valuable time and test resources for other test tasks. This would save valuable test time when running performance measurements of reliability characteristics, such as stability or availability figures. Such test cases can be configured for several days of continuous test execution.

- Performance test tools should be capable of drawing conclusions from earlier test runs in order to focus coming test runs on the most critical performance requirements of the SUT. Measuring system performance is a learning process of the tested systems behaviour in different situations and the impact of different changes on the system performance. Over 95 % of all performance measurements are regression tests that will over time generate very large amounts of collected measurement data. This is therefore an ideal for computer learning that will eliminate large amount of manpower. With help of a computer learning system and earlier measurement results a performance test tool will also be able to focus future performance measurements for every build on the most performance critical parts of the system. AI in Performance Test Tools can help in improvement in Quality of Measurement.

While over 95 % of all performance measurements are regression tests, the implications are as follows:

- Regression tests of system performance is a learning process in the behaviour of a tested system! An ideal case of AI for computer learning in a performance test tool.

- Frequent performance tests produce huge amounts of measurement data! This is also an ideal case for AI, to analyse collected measurement data, to draw long term conclusions of performance characteristics.

The following aspects provide insights on the value of AI in performance measurement tools. Figure 1 presents the value of AI in Performance Test Systems.

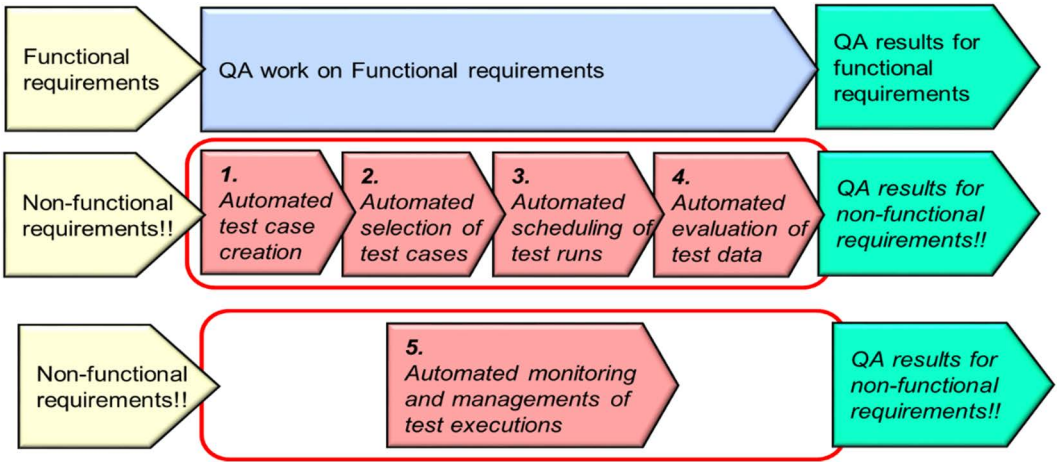

Figure 1: The value of AI in Performance Test Systems (extract from [i.8])

1. Creating performance test cases should be simplified. Creation of test cases is a complex operation that takes time and manpower. Test cases can be created directly from performance requirements. AI in performance test tools could solve this.

2. Selection of test cases for a test run should be automated. This is manual work today that can be eliminated. Automated selection of the most important test cases can be done with AI in performance test tools.

3. Automated performance tests should be executed for every build. This is not done today. Only the most important performance test cases are selected and executed. AI in performance test tools could solve this.

4. Automated evaluation of performance measurement results. This is manual work today that can be eliminated. Evaluation of measurement results can be done fast and precise based on performance requirements with AI in performance test tools.

5. Automated monitoring of performance tests that run for long time. Trend analysis of captured data can be done during execution. Test execution can be aborted when it is pointless to continue-thanks to using AI in performance test tools.

Other perspectives of relevance to the value of AI in Performance Testing are as follows:

1) System performance specifications;

2) Stored performance test cases; and

3) Stored measurement results can be fed into an AI Engine with machine learning algorithms to improve the following aspects:

   - Creation of performance test cases.
   - Selection of performance test cases per build.
   - Selection of performance test cases per build.
   - Evaluation of performance test results.

AI in Performance Testing can also be used for model inference and learning-based testing.

NOTE: [i.8] points to the fact that AI plays an important role in the convergence of performance testing and functional testing in the cases of testing AI models, AI components, and AI systems, in the sense that the assets used for functional testing form the basis for building performance Test suites that can be used for explorative testing of the AI model/component/system. The explorative testing is a kind of testing by which inputs and loads (workloads) on the target AI powered Object Under Test are varied by the performance system so that an AI model that characterizes the Object Under Test can be found out by the performance Tests since the AI Model of the Object Under Test may have been kept not disclosed to the Tester (who may be tasked to find out the AI model behind the Object Under Test and provide data (conditions) that shows how the Object under test performs under specific conditions), i.e. in model inference and learning-based testing. As discussed in [i.8] model inference for non-functional testing plays a role in learning the model from execution traces (online/offline) or through active learning, with the benefit that AI helps achieve a model for further test generation with small or no manual effort.

5.3 Security testing and fuzzing

5.3.1 Overview

Fuzzing is an established security testing technique that generates and feeds a system under test with invalid and unexpected inputs to detect vulnerabilities [i.31]. Its first approaches are called dumb today. They do not know anything about the system under test and generate inputs randomly to provoke crashes. Since this approach is simple and thus, achieves wide acceptance by industry and is a de-facto standard technique for security testing. However, random fuzzing is limited to detect simple bugs and is quite inefficient. Fuzzing has been evolved since then by using more and more protocol information to generate input data not completely random but more systematically aiming at generating so-called semi-valid input. Such inputs deviate only to a minor degree from valid input data and thus, are expected to find more subtle bugs. The drawback of this approach is that it requires much effort to provide the protocol information in a formalized way such that it can be used to generate data from it. If a fuzzing process have discovered bugs on the test item, post-fuzzing steps are necessary to de-duplicate bugs, determine their reproducibility, analysing their root cause and exploitability [i.34].

5.3.2 Research directions

Applications of AI to improve fuzzing can be aligned along the several steps of fuzzing, i.e. input generation, input selection, program monitoring, input assessment, and post-fuzzing tasks [i.34].

In input generation also touching program monitoring, genetic algorithms have been applied often in research to improve the fuzzing process. Genetic algorithms belong to the class of metaheuristic algorithms or search-based approaches that do not know anything specific about the problem to be solved. In case of genetic algorithms, candidate solutions, e.g. test cases, test inputs or protocol models, are crafted by biological evolution, i.e. mutation, recombination and fitness evaluation. A first seed of inputs is evolved in several iterations of these steps. Fitness evaluation is problem-specific component of the algorithm. Fitness functions were used to learn the protocol model [i.29], assess the feedback of the test item for specific vulnerabilities, e.g. reflection of malicious user inputs from web applications to detect cross-site scripting vulnerabilities [i.30] and increase code coverage [i.27]. More advanced approaches use Dynamic Markov Models to assess coverage along the control flow graph more precise [i.36]. Since the fitness function is the most critical component of a genetic algorithm, the performance of a fuzzing tool strongly depends on the ability of the fitness function to assess progress of security testing. Deep learning approaches have been used to create input grammars for complex file formats, e.g. long-term-short-term memory recurrent neural networks (LSTM) [i.32] which requires to balance between the ability of LSTMs to generate well-formed inputs and the goal fuzzing to generate malformed or semi-valid inputs [i.34]. Deep learning approaches have also been used to predict code coverage of newly generated inputs ([i.33], NEUZZ [i.35]). Increasing code coverage is a problem if fuzzing tools do not know the input format, e.g. magic bytes, and thus miss important parts of the test item. Reinforcement learning has been used based on state machines to select to learning which generated inputs or mutations on it are most promising in finding new errors, increasing coverage of program functions and corruption or delay of response messages [i.28] and [i.11].

A crucial step in the vulnerability assessment is the classification of the vulnerability type and estimation of its exploitability, e.g. what are the preconditions and skills an attacker requires to use a vulnerability to an intrusion. Exploitmeter [i.37] describes an approach based on static analysis and different fuzzing tools to assess a vulnerability using a Naïve Bayes Classifier. Finally, the approach computes an exploitability score for the whole software.

5.3.3 Industrial Relevance

Genetic algorithms have been successfully established in fuzzing with the rise of the open-source fuzzing tool American Fuzzy Lop (AFL). This tool has been used in many industrial projects. However, its drawbacks have led to further improvements of AFL that basically confirm these drawbacks. The problem is that certain paths are not reachable since AFL does not know anything the input format and is not efficient when it is faced with magic bytes and specific offsets that would activate the execution of certain program paths and can make its application quite inefficient. A barrier for deep learning application in security testing is the training time [i.34]. Pre-trained models may be an approach to cope with this issue. However, it is not well understood in which situation, with regard to the specific input format and communication protocol of the certain test item, deep learning approaches are increasing the performance of fuzzing tools. Approaches of reinforcement learning are promising but suffering from the understanding of a good reward function.

5.4 Test execution automation

5.4.1 NLP in automating manual legacy tests

NOTE: This perspective is a subject for further study in the evolution of the present document.

5.4.2 Reinforcement learning in explorative testing

NOTE: This perspective is a subject for further study in the evolution of the present document.

EXAMPLE: Aspects such as use of AI in driving testers to discover the need of SUT for further study.

5.4.3 Anomaly detection and runtime verification

NOTE: This perspective is a Subject for Further Study in the evolution of the present document. However, some of the aspects that belong to this scope are as follows: Anomaly detection through neural networks (autoencoder) and other statistics; Scope: use of AI in reliability testing of ordinary and intelligent systems; reliability of systems or networks, compare with IEEE 1633 "SW Reliability" [i.60]; the conclusion could be that there is a need in industry for an AI-driven framework for reliability testing of systems.

5.4.4 AI-empowered object recognition in GUI testing

NOTE: This perspective is a subject for further study in the evolution of the present document.

5.5 Online testing

This clause discusses the value (benefits) of AI in Online Test Systems, whereby as defined in [i.8], an Online Test System is one used for testing a System or Component that is integrated and actually running in a production/real environment, in which it is designed to operate.

Aspects of AI that can play a role in Online Testing include Reinforcement Learning, Model Inference and Learning-based Testing, Predictive Runtime Verification through AI, e.g. Predictions via Bayesian Networks and Deep Learning.

NOTE 1: [i.8] provides useful insights on the benefits of AI in Online Testing Systems.

The following perspectives are examples of application of AI in Online Testing:

1) AI in Online Test Systems used for inferring, computing/deriving values of certain metrics or KPIs in mobile access networks (e.g. 5G mobile access networks) while applying AI (e.g. ML) in such processes. An example of such systems is the Online Testing for Voice call stability measurements/assessments as one of the main targets for mobile network optimization, by which Call Stability Score (CSS) is a new KPI to measure call stability using machine learning as described [i.8]. A neural network is trained with a large dataset of real call tests, to output a normalized value from 0 to 1 that represents how far a call is from the drop calls the model has seen, as described in [i.8].

2) Use of AI in Online Testing of Service Performance in an ICT network using Active Probing and/or Passive Probing (i.e. using Active Testing and/or Passive Testing). In such a case, data that is directly of relevance to measuring and trending the performance of a single service or multiple services is collected and correlated with other data that may be relevant to consider (e.g. time of day or weather) in order to determine factors that positively or negatively impact service performance, using AI algorithms (e.g. Machine Learning algorithms) in Analytics. In some networks, e.g. in data center networks, a service may consist of multiple components and communication flows (e.g. IP communication flows) between the components in delivering the service to the user. Whereas in other networks like telco networks a service may consist of components, interfaces, protocols and communication flows among the components that are used to deliver the service to the service consumer (e.g. an end user that uses an end terminal). In service performance testing data from performance of individual flows that belong to the service is also collected for the analytics concerning the service performance. As discussed in [i.8], both Active Testing of Services and Passive Testing of Services (including use of Passive Probing and Traffic Analytics) can play complementary roles in Network Testing and E2E Services Testing.

3) Use of AI (e.g. Machine Learning) in Online Active Testing System (and/or Passive Testing System) for Testing and Measuring and Analysing the impact of the Multi-Layer Autonomic Functions (e.g. GANA Decision Elements (see ETSI TS 103 195-2 [i.1])) and their AI Models when the Autonomic Functions (as software modules) are activated to run in a "closed-loop" mode to dynamically adapt the network(s) resources, parameters and services and then toggled to run in "open-loop" mode (or even turned-off) such that service or network performance testing is performed for both cases and KPIs data are then analysed using AI (e.g. in the analytics) and compared across the two cases in order to deduce the impact of the autonomics software (autonomic functions) on service and network performance. With the Autonomic Functions interchangeably configured to operate in "open-loop" (or even turned-off) and "closed-loop" active service or network performance testing and/or passive testing can be performed and service performance KPIs or network performance KPIs data gathered and analysed using AI to infer the impact of the Autonomic Functions on service and network performance and/or even resilience (when impairments are injected into the network as well during the tests). Such Test Systems and Test Procedures may be necessary in cases of indirect testing of AI powered cognitive Autonomic Functions (e.g. Cognitive GANA DEs) as "black-boxes" without needing to know the AI models employed by the Autonomic Functions, as described in [i.8]. [i.8] provides an example of such a Test System and the case of E2E Service Testing of an E2E Network to Measure and Analyse the impact of the Multi-Layer GANA Autonomic Functions (Decision Elements) when the DEs are activated to run in a "closed-loop" mode to dynamically adapt the network(s) parameters, resources or services.

NOTE 2: Such Test Systems need not be Online Test Systems as such since similar test objectives can also be carried out by AI powered Non-Online Test Systems that could use in Lab Environments hosting the NUT (i.e. the NUT is not a production online network carrying production traffic).

4) Use of AI in Test Results Analytics in general.

Some example perspectives on the application of the following techniques to Online Testing include the following (but there are more other techniques that are relevant to AI in Online Testing found in other literature):

- Reinforcement Learning [i.47];
- Model Inference and Learning-based Testing [i.48] and [i.49];
- Predictive Runtime Verification through AI, e.g. predictions via Bayesian Networks, in sources such as [i.50], [i.51] and [i.52].

NOTE 3: The AI techniques are also applicable for benefits of AI for robustness testing and reliability testing.

5.6 Test Orchestration and TestOps

Test orchestration relates to dynamic instantiations of test components (if no instances already exist that can serve the test plan to be executed) and triggering the execution of the test plans and associated test objectives, and such orchestrations may be dynamically triggered by the results from AI techniques such as Machine Learning over data gathered about the System Under Test (SUT), Component Under Test (CUT), or Network Under Test (NUT).

The concept and culture of TestOps has recently emerged following the success of DevOps concept and culture in software development and lifecycle management. There are many sources in literature that describe TestOps and the value it brings, especially when combined with DevOps. For example, [i.44] (White Paper) and [i.46] describe what TestOps is about and provides various insights on the following principles of TestOps that apply to both, software products implementations and operations, and hardware products implementations and operations: TestOps as "DevOps-Style Automation for Design and Test"; TestOps as an approach to extend the well-understood DevOps strategy to engineering design and test workflows; and Principles of Connected, Agile Design and Test (which can be considered generic to some extent in as far as they are application to various types of products). There are various aspects of TestOps described in [i.44] (White Paper) and [i.46]. There are various sources in literature that describe the value of AI in TestOps, such as [i.45] and [i.46].

Some example perspectives on the application of the following techniques in TestOps include the following (but there are more other techniques that are relevant to AI in TestOps found in other literature):

- Reinforcement learning [i.47];
- Model Inference and Learning-based Testing [i.48] and [i.49];
- Predictive Runtime Verification through AI, e.g. predictions via Bayesian Networks, in sources such as [i.50], [i.51] and [i.52].

5.7 Test result analysis and fault localization

NOTE: This perspective is a subject for further study in the evolution of the present document. However, the following aspects are of relevance to consider and study in this scope of use of AI in fault localization:

1) AI applied to faults localization in functional testing;
2) root cause analysis and filtering false positives and true negatives via precision and recall;
3) Fault Isolation/Diagnosis/ Localization in the scope of Network Management and Operations Tasks (including in the scope of autonomic management and control operations by autonomics software).

5.8 Selection of regression test cases

5.8.1 Background and Terminology

Most software-based systems need to continue to evolve due to the ever-changing operational context, defined by changing user requirements, changing regulations, changing hardware and platforms, and other factors. Regression testing is a common approach to ensure that the modifications to the systems do not introduce new faults. Given a system S and a corresponding test suite T, regression testing is concerned with validating a modified version S' of S, and in particular ensuring that the outcomes of running the test suite T on S' are not worse than the outcomes of running the test suite on the unmodified system S. Ideally, the test suite should be executed after every change to identify potential problems as soon as possible. For large and complex systems, running the tests after every modification can be an expensive and time-consuming task. Especially considering that modifications of mature systems typically affect only a small portion of the system. It is therefore essential to reduce the costs of running the tests to the extent possible, while maintaining their effectiveness.

Different approaches to address these challenges have been proposed in research on regression testing. Fundamentally these can be categorized as Test Case Selection (TCS), also known as Regression Test Selection (RTS) and Test Case Prioritization (TCP), as outlined by Elbaum et al. [i.13] test case selection is concerned with identifying a subset T' of the test suite T containing the test cases that are relevant and important to re-run. Test case prioritization, on the other hand, assigns priorities and reorders the test cases in T in order to meet test objectives sooner or ensure that the test effectiveness is maximized given a limited testing budget. Related to these approaches, Test Case Minimization (TCM), also referred to as test suite reduction, aims to identify and eliminate redundant test cases from the test suite.

5.8.2 Recent Advancements in Research

Khatibsyarbini et al. 2018 [i.14] categorized research on prioritization of tests in regression testing up to 2017 based on:

- Coverage - concerned with code coverage.
- Requirements - considering information related to requirements for prioritization based on traceability, completeness, change impact analysis for requirements, as well as the impact of a failure on requirements.
- Risk - concerned with risk assessment.
- Search-based - concerned with the application of search-based algorithms such as genetic algorithms and ant colonies.
- Faults - focusing on fault-revealing capability of tests regarding specific types of faults, often while minimizing costs.
- History - using historical data, such as execution history or change information for code and/or tests.
- Other - including predictive models to estimate the probability of detecting an error, cost-aware approaches, and others.

There is plenty of research literature on the topic, with the search-based and coverage-based categories being the most prevalent. More recently there has been an increasing presence of AI-based techniques as well as multi-objective optimization techniques and especially the consideration of costs. While defect prediction is also a very active topic in research promising to help with test case prioritization, it has yet to find a widespread issue as there are still many concerns. Instead, continuous integration and DevOps principles are increasingly adopted in practice, which has led to increasing research interest to study and enhance these practices.

Marijan et al. 2019 [i.15] outlined a practical learning algorithm for optimizing continuous integration testing by learning and predicting the effectiveness of tests using historical test records combined with coverage-based redundancy metrics in order to reduce test redundancy. The approach was evaluated in an industrial case study of highly configurable video conferencing system at a major software and hardware provider. Grano et al. 2019 [i.41] evaluate the use of source-code metrics to predict the coverage of generated tests with different machine learning algorithms in order to optimize test effectiveness within a given search budget. In contrast to widely spread static prioritization approaches, Pradhan et al. 2018 consider a dynamic test prioritization approach using a rule miner to extract relations among tests based on historical execution, a static prioritizer using multi-objective search, and a dynamic executor and prioritizer to update the test order during test execution based on the results from the completed test cases. The approach showed some promise in an initial evaluation with two industrial and three open-source case studies.

De Castro-Cabrera et al. 2020 [i.16] propose an extension of the categorization by Khatibsyarbini et al. 2018 [i.14] based on latest advancements in the field reported in the literature since 2017. The proposed extension included categories based on:

- Machine learning - using models based on historical data, execution times, and descriptions in natural language, etc.
- Neural networks - similar to machine learning but focusing on neural network models.
- Empirical-studies - evaluating and comparing different approaches for prioritization in larger studies.

The extended categorization indicates that AI-based techniques are beginning to gain traction in research on regression testing and specifically on test case prioritization. Given the fact that the publications are recent, while some of the techniques show promise, their suitability for industrial deployment and application needs to be evaluated.

5.8.3 Industrial Relevance

Overall, there is a large body of research literature on regression testing in general. However, aspects related to industrial application are frequently not considered extensively, as reported by Ali et al. 2019 [i.17].

In their effort to direct researchers towards regression testing research with more focus on industrial relevance and applicability, and to facilitate the industrial adoption of regression testing approaches described in research, they considered eleven literature reviews on regression testing published since 2010.

Their findings indicated that research results are often hard to adopt for the practitioners, in part due to differences in terminology, and due to simplified experimental settings rather than considering the full complexity of an industrial setting. While there are considerable reports of industrial evaluations of regression testing, these are often hard to assess due to the lack of conceptual models verified by practitioners to interpret, compare, and contrast different the different approaches. Their findings indicate, that while code coverage is a popular measurement in literature, for example, it was not considered relevant by a designated focus group of practitioners. Instead, more focus is to be dedicated to feature coverage, detection of severe faults, and reduction of cost, for example.

6 AI-enabled test technologies applied in unit and integration testing

6.1 Overview

This clause discusses the types of AI technologies introduced in clause 5 that can be applied best in certain test phases. This includes an analysis on how each test phase could benefit from AI and associated AI technology type. It then provides some guidelines for the use of AI in testing along the different test phases.

6.2 Unit testing

6.2.1 Background and Terminology

The scope of consideration in this case is the application of AI test methods on software components, i.e. at a source code level.

Unit testing is concerned with the testing of individual software or hardware components. The definitions of units and unit tests vary across the literature and even standards (e.g. ISO/IEC/IEEE 24765 [i.61] and the ISTQB glossary). Unit testing is an important practice in both traditional and agile software development.

While unit tests focus on small, isolated pieces of functionality, with large and sophisticated systems composed of a large number of units, unit testing becomes difficult to apply at scale, where unit tests are no longer added or updated for new or modified functionality. Numerous techniques for test generation have been proposed to assist with scaling up the use of unit tests. These techniques seek to address common challenges in unit testing, including generating suitable test cases, determining suitable test coverage criteria and the assessment of the test coverage according to the selected criteria, as well as the assessment of the test adequacy, which frequently involves mutation testing.

Assert statements in tests ensure that the outputs of invoked functionalities, as well as pre- and post-conditions are as expected.

Suitable assert statements require an understanding of the purpose and context of the unit test, including the functionality being tested. Effective techniques for test generation need to consider rich contextual information in order to determine the correct type and arguments of assertion statements, which is still a key challenge. Assert statements are frequently generated based on heuristics and randomized approaches which are not always of high quality and therefore lead to missed faults.

6.2.2 Recent Advancements in Research

Techniques for unit testing have not yet adopted AI widely so far. Watson et al. 2020 [i.42] employed Neural Machine Translation (NMT) for the automated generation of meaningful assert statements. The proposed approach showed promise by inferring the exact assert statements as manually written by developers and proposing suitable alternatives. It could be integrated with both other means for automated test generation as well as manual unit test implementation.

Executing large test suites for every software build in Continuous Integration (CI) environments is not always feasible, especially if new tests are also generated as part of the CI workflow. Test selection and test prioritization, especially for generated tests, can benefit from additional information such as expected test coverage. Grano et al. 2019 [i.38] investigated the prediction branch-coverage achieved by data-generation tools in automated test generation. They evaluated different machine learning algorithms trained using source-code metrics and found that coupling-related metrics are most important for branch-coverage prediction.

6.2.3 Industrial Relevance

While unit testing is assumed to be widely adopted in practice, studies have indicated that practitioners may not be very concerned with the exact boundaries between unit, integration, and system tests. It is therefore difficult to estimate whether AI-based techniques will benefit unit testing specifically or rather testing in general, as AI-based techniques are often targeting specific test-related activities rather than the granularity of the tests. Test generation for unit tests has been a topic of active research for several decades, however, recent findings by e.g. Fraser et al. 2015 [i.39] and Almasi et al. 2017 [i.40] cast some doubts on the suitability of the generated tests for finding actual faults in practice. The approach by Watson et al. 2020 [i.42] seeks to address some of the concerns with regard to the quality of assert statements in particular with the help of AI, however, as it is a fairly recent proposal, its industrial relevance remains to be evaluated in the future. Techniques for test suite selection and prioritization discussed in the context of Regression Testing may be relevant for unit testing as well.

6.3 Integration testing

NOTE 1: This perspective is a Subject for Further Study in the evolution of the present document. However, the following aspects are of relevance to consider and study in this scope of use of AI in Integration Testing.

NOTE 2: What could be considered in the further study are: inputs from ETSI TC MTS (regarding methodology); and ETSI TC INT (e.g. testing of 5G network slices and using the 5G PoC program of ETSI TC INT AFI WG to run Demos on the use of AI in Integration Testing that involves E2E Network Slicing and ETSI GANA Multi-Layer Autonomics/AI algorithms in Network Slice Service and Security Assurances).

7 AI-enabled testing in standardization and beyond

7.1 Testing for certification of AI

7.1.1 Overview

Benefits of use of AI in Certification of AI Models, Components, Systems and AI-powered ICT Networks include conformance/interop testing aspects for the AI being targeted for certification.

To show the benefits for a certifying agency it is necessary to look at a system as a whole comprising AI (for example) and provide answers to the following questions:

- How to certify, e.g. metrics that need to be used as the foundation for certification of AI Models, Components, Systems and AI-powered ICT Networks?
- Certification of a separated AI/ML model?
- Integration aspects for AI integrating into a system?

Certification is the ultimate step of a three-step process covering:

- Validation.
- Trustworthiness.
- Certification.

It needs defining:

- Standardized Metrics pertaining to the measurable value of AI empowerment in Components, Systems and ICT Networks targeted for certification.
- Independent Body to deliver verdict and certification label.
- Certification as a Service.

Metrics pertaining to specific classes of AI Models that can be targeted for Testing and Assessment, should be addressed in ETSI because such Metrics AI definitions are largely missing in the work done in the various other Standardization Groups today.

Besides specifying a Framework for Certifying AI Models, this clause also addresses the need for **"Test & Certification Framework for AI Models for AMC"** (Autonomic Management & Control) to support the industry in implementing and achieving Multi-Layer AMC for Autonomous Networks being specified by ETSI and other SDOs/Fora:

- Definitions of Standardized Metrics for Measurements and Assessments in Testing and Certification of AI Models.
- Definitions of Standardized Metrics for Measurements and Assessments in Testing and Certification of AI Models of Autonomic Components/Systems.

7.1.2 Application Case on Metrics for use in Certification: Types of Standardisable Metrics for Testing and Certification of AI Models of Autonomic Components/Systems

There are various metrics that can be used for assessment and differentiation of AI Models such as Machine Learning Models as described in various sources in literature and in [i.8]. There is the need to leverage such metrics and expand them with other metrics that can be standardized across comparable AI Models for a specific domain of AI application. The following are examples of standardisable Metrics for Measurements and Assessments in Testing and Certification of AI Models of Autonomic Components/Systems (and such metrics need to be complemented with other relevant metrics that can also be standardized):

- Stability of the AI Model.
- Speed of Learning of the AI Model.
- Speed of Decision-making cycle of the AI Model after receiving triggering inputs.
- Speed of Convergence of multiple interacting AI Models/components in a larger AI System.
- Quality of Decision-Making of the AI Model.

This list of Standardisable Metrics for Measurements and Assessments in Testing and Certification of AI Models of Autonomic Components/Systems (including Autonomic ICT Networks) is subject to possible expansion by the community and the Metrics are expected to be further detailed in one of the deliverables of the newly launched work item in ETSI on AI in Test Systems and Testing AI Models.

7.1.3 Introducing the concept of an AI-Support System (AI-SS) in testing and certification life cycle

Figure 2 illustrates the AI Model Life Cycle Management process (*Development, Training, Testing, Certification, Deployment*) with the associated three main stakeholders: *AI Regulator / Auditor, 3rd Party AI Model Tester, AI Model-dependent Certifier*. Each of the three mentioned stakeholders provides a related support (Methodology, Function, Process, assessment Metrics, etc.) as a part of what ETSI TC INT AFI WG named an **AI Support System (AI-SS)** as depicted in Figure 2. The rectangle shows the demarcation of the three parts of the AI-SS. Each of the three components of the AI-SS could be provided /offered as a service:

- Regulation as a Service (RegaaS).
- Testing as a Service (TesaaS).
- Certification as a Service (CertaaS).

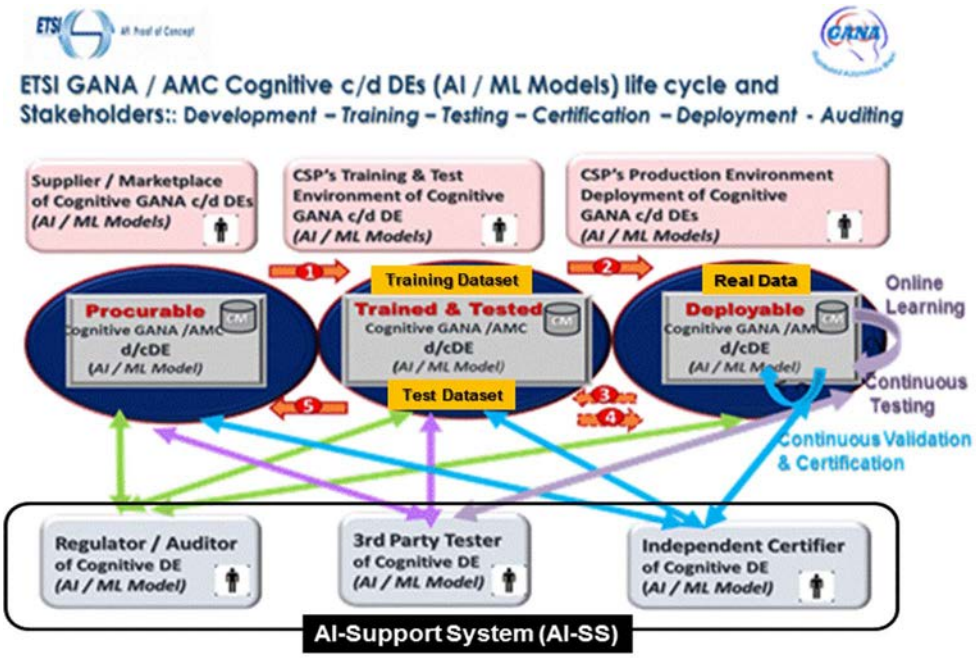

Figure 2: AI / ML Models life cycle and Stakeholders: Development - Training - Testing - Certification - Deployment - Auditing

As indicated in Figure 2, Training, Testing, Validation (Deployment) phases require dedicated Dataset that need to be prepared according to the following steps:

- **Step 1: Dataset preparation:** Data Scientist prepares cleaned and accurate Data after processing raw Data collected from available sources: Operator's Network, OSS/BSS as well as data captured from external sources (social media, etc.).
- **Step 2: Data separation:** Data Scientist splits the resulting Dataset obtained from step 1 into two separate Datasets: Dataset for Training and Dataset Testing. Various mechanisms could be used to achieve this splitting (Random, etc.). Then he trains and tests the GANA DE (AI Model) then he hand overs to the domain expert who takes care of the third and last step GANA DE (AI Model) "Validation".
- **Step 3: Validation:** This step is handled by the domain expert who knows the capabilities of the network. His role in this 3-step process is to provide "an internal certification" that declares the GANA DE (AI Model) is "Operations Ready" meaning it can be deployed in safe manner in the production network. This step consists of exposing the GANA DE (AI Model) to real data that are similar to the ones GANA DE (AI Model) can see in the Production network.

NOTE: There is need to take caution with the concept of "Validation". In this context is some "internal certification" that declare the GANA DE (AI Model) after passing successfully "Training" and "Testing" steps is ready to be deployed in the Production network.

7.2 Introducing an AI-enabled autonomic test system concept

7.2.1 The abstract model of an Autonomic Test System (ATS)

While studying the trending in the introduction of autonomics to systems designs, e.g. autonomics in network devices of functions (virtual or physical), in management and control systems for specific ICT networks, in IT systems, in cyber physical systems, and other kinds of systems, what can be appreciated is the intelligence that autonomics bring to such various kinds of systems. Autonomics is about the science and principles of embedding closed control loops in the systems as described in [i.18] and ETSI White Paper No.16 [i.58] and other sources in literature. The Autonomics paradigm and its associated science of control-loops is an enabler for designing systems and networks that exhibit a property of being "autonomous" in as far as the degree to which they can operate and automate tasks and decisions without human involvement in the intelligent and complex decision-making process is concerned. Thanks to "autonomics" control-loops structures and logics.

"Autonomics" is a bio-inspired concept derived from the concept of the "Autonomic Nervous System" of a Human Body. Autonomics software benefits a lot from being empowered by Artificial Intelligence (AI) Algorithms/Models for use in driving the decisions and actions made in the operations of the Autonomics (Closed Control-Loops). There are tremendous benefits of introducing autonomics in Test Systems as well, and various studies and solution proposals in this area are emerging, such as in [i.9]. This means AI-powered Test Systems can actually be turned into Autonomic Test Systems to make them "intelligent" and self-managing and self-adaptive in some of their Testing Objectives and Behaviours. Some of the autonomics associated properties that should be considered in designing an Autonomic Test System are the following (based on ideas described in sources such as [i.9]):

- **Self-awareness** - The ability of the Autonomic Test System to monitor its own state and behaviour, and maintain a model ("self-model") that characterizes itself.
- **Self-Configuring** - The ability of the Autonomic Test System to reconfigure itself in response to changes in Test Objectives supplied as inputs by the Human (Tester) and/or in response to changes in the interfaces of the System Under Test (SUT), Component Under Test (CUT), or Network Under Test (NUT).
- **Self-managing** - The ability of the Autonomic Test System to automate certain test management tasks such that the human tester is relieved from having to manually perform those tasks, while the Autonomic Test System also has the ability to auto-discover changes on SUT, CUT, or NUT that require the Autonomic Test System to compute (on-the-fly) Test plans, execute the plans, analyse results, notify human operator (tester) and log test results.
- **Self-Diagnosing** - The ability of the Autonomic Test System to employ various Fault-Localization (Fault Isolation) techniques that may be necessary to localize and isolate a fault within itself such that the system can trigger its self-repair and self-healing mechanisms and techniques.
- **Self-Repair** - The ability of the Autonomic Test System to use the results of Fault-Location (Fault-Isolation) to exercise the process of fault-removal by which the system may even automatically fetch and apply software patches on its own and even temporarily freeze operation and reboot itself before resuming the tasks the system was performing prior to the fault-detection event (incident).
- **Self-Healing** - The ability of the Autonomic Test System to fix problems within itself with zero or minimum impact on any ongoing test execution tasks.
- **Self-Optimization** - The ability of the Autonomic Test System to monitor and adjust the test system's resources based on the test workload to achieve test-execution efficiency.
- **Self-Adaptation** - The ability of the Autonomic Test System to automatically select and execute Test Cases (e.g. regression tests) based on Test Results from previous Test Runs, knowledge derived from the environment of the SUT, and any other input or knowledge derived from the SUT and other data sources.

NOTE 1: There are some additional features that could be also be associated with the design of an Autonomic Test System that are described in [i.9].

The value brought by the ATS concept:

- ATS enables testing of complex systems that include AI-enabled systems and networks.

- Driven mainly by the need for introducing autonomics & AI in Test Systems due to rising complexity of System and Network Under Test (SUT/NUT) now commonly encountered in today's ICT world.

Figure 3 presents an Abstract Model of an Autonomic Test System that can be used to guide in designing and implementing Autonomic Test Systems.

NOTE 2: While the present document provides a draft detailed description of an Autonomic Test System, a much more detailed specification is planned for such that a Technical Specification (TS) should be developed in ETSI.

NOTE 3: The instantiation of the concept of an Autonomic Test System means an Autonomic Test System is designed for a specific target System Under Test (SUT), Component Under Test (CUT), or Network Under Test (NUT). The various instantiation cases are out of the scope of the present document (though the subject is touched upon briefly in the clauses that follow), and it is expected that following the anticipated work in ETSI on a Technical Specification (TS) that provides a detailed specification of the concept of an Autonomic Test System some work would also be triggered in ETSI on illustrative example instantiations of the concept and associated Demo (Proof-of-Concept) Cases.

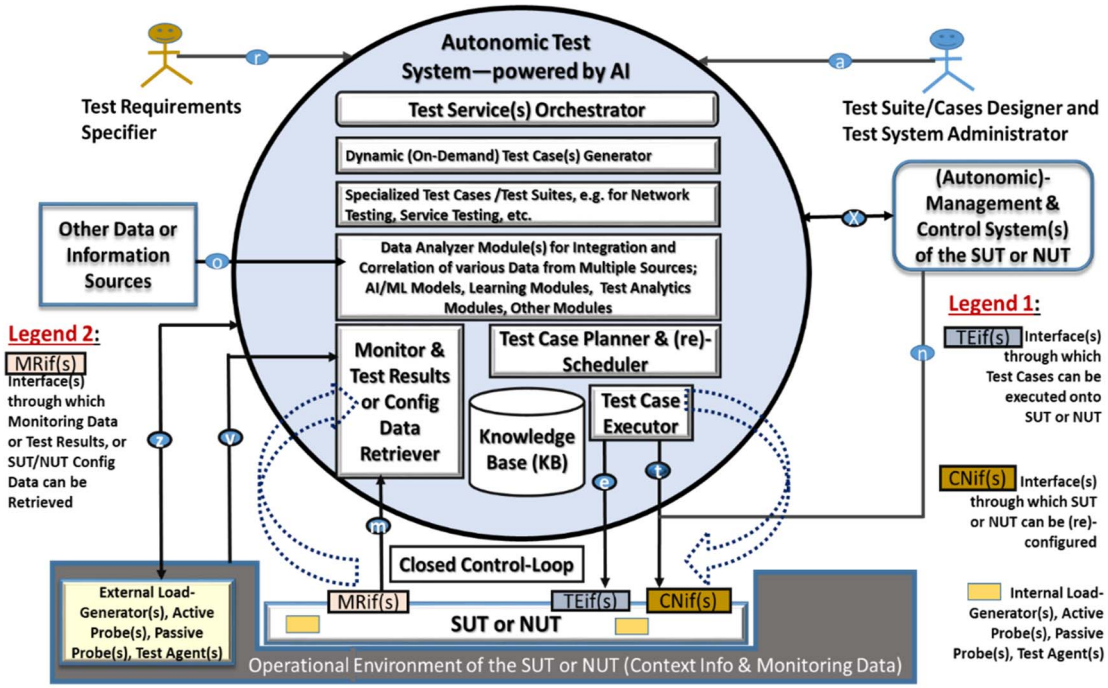

Figure 3: The Abstract Model of an Autonomic Test System (ATS)

NOTE 4: The use of an ATS in testing SUT and NUTs that exhibit intelligence through autonomics and AI models embedded within then can be very instrumental to Certification of such AI powered SUTs and NUTs.

7.2.2 Types of instantiations of the ATS Concept

7.2.2.1 Overview

There are three aspects that pertain to instantiations of the ATS concept, namely:

1) Instantiation of the ATS concept within the scope of a specific technology domain from which the type of the SUT or NUT derives, e.g. the domain of ICT networks, by which an SUT can be a Network Function such as a Router, a switch, or IT server, and a NUT can be a Radio Access Network (RAN) such as a 5G RAN, or NUT can be an X-Haul Transport, a Core Network, or a Data Centre (DC) Network Infrastructure or a Cloud, etc.

2) Instantiation case of the ATS concept that considers Human-in-the-Loop in its Operations, i.e. the case in which Humans play some roles in Test Requirements Specifications and some level of Test Suite/Cases Design and Test System Administration (including reactions to certain situations the ATS requires human interventions during its operations)

3) Instantiation case of the ATS concept that does not necessarily consider human involvement in its operations (i.e. the case of Self-Testing capability of a System such as a Network Function such as a Router (e.g. Broadband Network Gateway (BNG), Switch or a much more complex system that belongs to a certain technology domain)

7.2.2.2 Instantiation case of the ATS concept that considers human-in-the-loop in the ATS's operations

This type of instantiation of the ATS concept considers Human-in-the-Loop in its Operations. This means the case in which Humans play some roles in Test Requirements Specifications and some level of Test Suite/Cases Design and Test System Administration (including reactions to certain situations the ATS requires human interventions during its operations). The Test Requirements Specifier produces various kinds of Test Requirements that the ATS should address at specific times of executing corresponding Test Suites or Test Cases. The Test Suite/Cases Designer translates the Test Requirements into specific Test Suites or Test Cases designs and populates the ATS with such Test Suites/Cases. The Test Suites/Cases can then be executed by Test System Administrator or can be selectively executed by the ATS based on its autonomic operations-by which the ATS reacts to test service requests from privileged entities or *self-programs* testing targets based on its intelligence of testing an SUT, Groups of SUTs or NUT(s) when set into action of running Tests that may last for long (hours, even days, weeks, etc.) of a mix of Functional, Performance and/or Scalability Testing. Such a resultant ATS instantiation is kind of which the ATS is external to the SUT or NUT as shown in Figure 3 (The Abstract Model of an Autonomic Test System). In such instantiations of an ATS, the ATS may involve the Human in the loop such that when there are certain situations the ATS requires human interventions the ATS escalate such situations to the Human (Test System Administrator).

NOTE: Clauses 7.2.3 on "The Desirable Framework for Integration of AI-powered Autonomic Management and Control (AMC) Systems and AI-powered Autonomic Online Test Systems during Network Operations" and 7.2.4 on "Benefits of ATS in multi-vendor network environments" cover this type of Instantiations of an ATS.

7.2.2.3 Instantiation case of the ATS concept without human involvement

This instantiation case of the ATS concept is a kind that does not necessarily consider human involvement in its operations. It is particularly relevant for the case of *Self-Testing capability* of a system such as a Network Function/Element (NE/NF) such as a Router (e.g. Broadband Network Gateway (BNG), a Switch or a much more complex system that belongs to a certain technology domain). In the field of autonomic or autonomous systems self-testing is a desired capability of an autonomic system. There are two perspectives to self-testing capability as a capability of certain type of a system. The first perspective is that there is growing work in research in this area, under the umbrella of Dependable Systems, that look into need for self-testing capability of system as part of the autonomic features of a system, along with the other self-features such as self-configuration, self-healing, self-optimization, self-diagnosis, self-healing, etc. Examples of literature that address this topic are [i.53] and [i.54]. The second perspective of Self-Testing capability of a System is that this capability could be required to run while the system that exhibits such a capability is made to run offline in order to characterize its behaviour over time and under various conditions and challenges meant to test the system robustness before it is put into a production environment, for example. In the first perspective and to some extent in the second perspective as well, the *Self-Testing capability* can be triggered by internal autonomic management and control functions of the system (e.g. ETSI GANA DEs for Autonomic Fault Management, Autonomic Resilience & Survivability, Autonomic Security Management, etc.) such that the test results are consumed by the triggering autonomic management and control function(s) to aid it in its subsequent decisions. The Self-Testing capability can also be triggered by an external management and control system (e.g. autonomic functions such as ETSI GANA DEs running within an external autonomic management and control system) that is meant to manage and control the behaviour of the system hosting the self-testing capability in real-life deployment and operations. Figure 4 presents the instantiation case of the ATS concept that does not necessarily consider human involvement in the ATS's operations (i.e. the case Self-Testing capability of a system). This instantiation case is relevant to the two perspectives on self-testing described above. In this instantiation case the ATS is internal to the SUT, and the interfaces *TEif(s), CNif(s)* and *MRif(s)* are either internal to the SUT or the interfaces are also exposed to the outside world by the SUT. If the *TEif(s),* CNif(s) and *MRif(s)* are not internal to the SUT but are the interfaces that are exposed to the outside world by the SUT, then the internal ATS can access and use these interfaces via a Loopback interface of the SUT.

NOTE: As illustrated in Figure 4, the human roles of Test Requirements Specifier, Test Suite/Cases Designer and Test System Administrator, described in the case of Instantiation case of the ATS concept that considers Human-in-the-Loop in the ATS's Operations, may be involved to some extent in certain contexts.

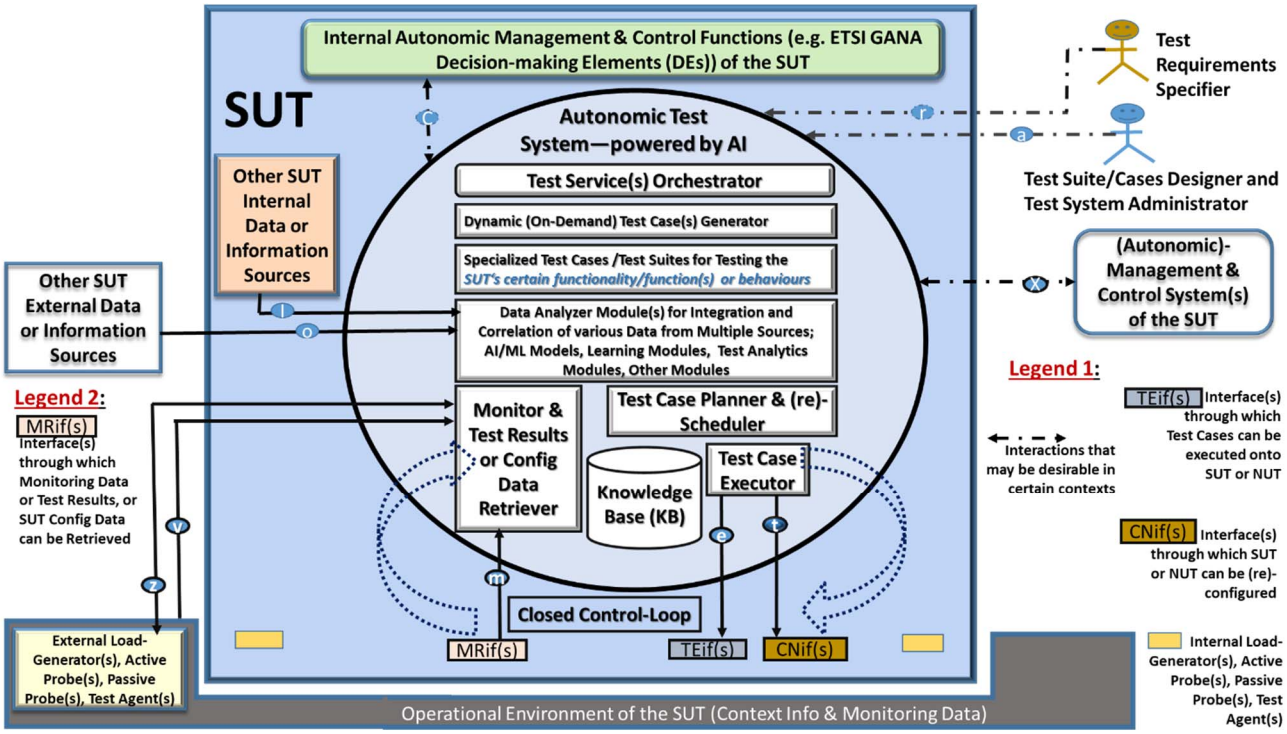

Figure 4: Instantiation case of the ATS concept that does not necessarily consider human involvement in the ATS's operations (i.e. the case of Self-Testing capability of a System)

NOTE: The subject of various detailed instantiations cases is out of the scope of the present document, and it is expected that following the anticipated work in ETSI on a Technical Specification (TS) that provides a detailed specification of the concept of an Autonomic Test System some work would also be triggered in ETSI on illustrative example instantiations of the concept and associated Demo (Proof-of-Concept) Cases.

7.2.3 Integration of AI-powered AMC Systems with an ATS as AI-powered Autonomic Online Test System

NOTE 1: This clause covers a CSP's desirable framework for integrating AI-enabled Autonomic Management and Control (AMC) systems and AI-enabled autonomic (online) test systems.

Figure 5 presents the Desirable Framework for Integration of AI-powered Autonomic Management and Control (AMC) Systems and AI-powered Autonomic Test Systems that take the nature of an Online Test System (i.e. an AI-powered Autonomic Online Test System).

Figure 7 presents the value of having an AI-powered Autonomic Test System that is meant be used in the Test Lab to test an AMC System for a Scenario in which the targeted autonomics software (autonomic manager components, e.g. ETSI GANA Model Decision-making Elements (DEs)) to be tested by the AI-powered Autonomic Test system is the High Level ("Macro Level") Autonomics in the AMC system and Low Level ("Micro Level") Autonomics software in certain Network Elements/Functions (NEs/NFs) of the underlying network infrastructure.

NOTE 2: In the ETSI GANA Model for AMC (see ETSI TS 103 195-2 [i.1]), the "Macro-Level" autonomics control-loops are referred to as "Slow Control-Loops" operating outside of Network Elements/Functions (NEs/NFs), while the "Micro-Level" autonomics control-loops are referred to as the "Fast Control-Loops" operating within NEs/NFs and may span multiple NEs/NFs for cases where distributed algorithms are used to implement "in-network" distributed control-loops. And the "Slow Control-Loops" policy-control the "Fast Control-Loops" in network infrastructure.

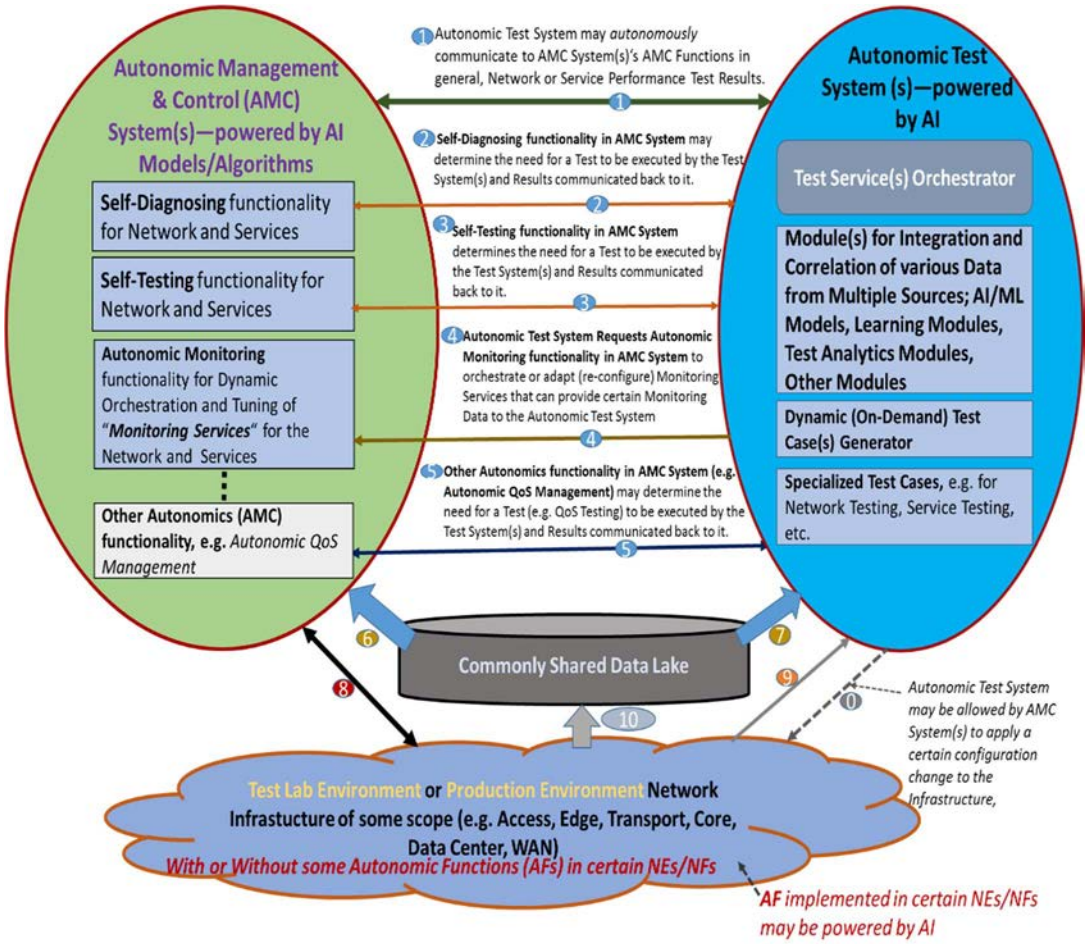

Figure 5: The Desirable Framework for Integration of AI-powered Autonomic Management and Control (AMC) Systems and AI-powered Autonomic Online Test Systems during Network Operations

NOTE 3: The Framework for Integration of AI-powered Autonomic Management and Control (AMC) Systems and AI-powered Autonomic Online Test Systems requires to be further detailed and such work may be covered in the same anticipated ETSI Technical Specification (TS) that provides a detailed specification of the concept of an Autonomic Test System or may be covered in a separate document that covers certain instantiations cases for the Autonomic Test System Concept.

Figure 6 presents an illustration of an example of the value of integration of GANA Knowledge Planes (KP) Platforms for AMC with Autonomic Online Test Systems within a single CSP. More details on this subject are found in [i.8].

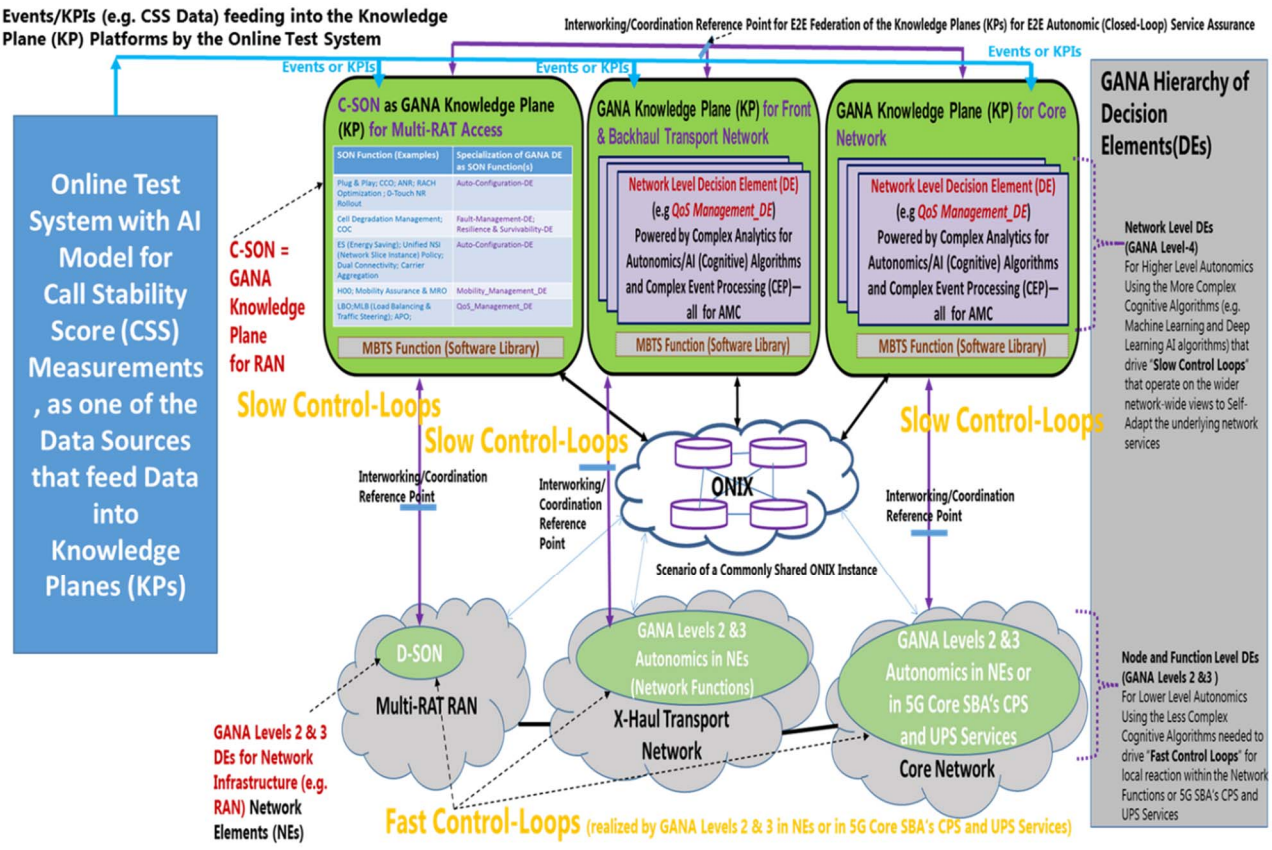

Figure 6: Illustration of an example of the value of integration of GANA Knowledge Planes (KP) Platforms for AMC with Autonomic Online Test Systems within a single CSP

Figure 8 in the ETSI White Paper No. 3 [i.55] presents the Integration of GANA Knowledge Planes (KP) Platforms for AMC with Automated Test System for Orchestrated Assurance that could possibly exhibit the property of being AI-powered Autonomic Online Test System as well.

NOTE 4: Clause 6 and Figure 88 in the ETSI White Paper No.3 [i.55] present detailed descriptions and scenarios on an Integration of GANA Knowledge Planes (KP) Platforms for AMC with Automated Test System for Orchestrated Assurance that could potentially exhibit the property of being AI-powered Autonomic Online Test System the as well.

7.2.4 Benefits of ATS in multi-vendor network environments

7.2.4.1 Overview

NOTE 1: Discusses the aspect of how network infrastructure suppliers and software suppliers, including independent software vendors (ISVs), benefit from AI-enabled test systems.

Figure 7 presents the value of having an AI-powered Autonomic Test System that is meant to be used in the Test Lab to test an AMC System for a Scenario in which the targeted autonomics software (autonomic manager components, e.g. ETSI GANA Model Decision-making Elements (DEs)) to be tested by the AI-powered Autonomic Test system is the High Level ("Macro Level") Autonomics in the AMC system and Low Level ("Micro Level") Autonomics software in certain Network Elements/Functions (NEs/NFs) of the underlying network infrastructure.

NOTE 2: In the ETSI GANA Model for AMC, the "Macro-Level" autonomics control-loops are referred to as "Slow Control-Loops" operating outside of Network Elements/Functions (NEs/NFs), while the "Micro-Level" autonomics control-loops are referred to as the "Fast Control-Loops" operating within NEs/NFs and may span multiple NEs/NFs for cases where distributed algorithms are used to implement "in-network" distributed control-loops. And the "Slow Control-Loops" policy-control the "fast control-loops" in network infrastructure.

NOTE 3: Figure 7 and Figure 8 describe Scenarios for the application of the ATS concept that are based on adapting the "The Desirable Framework for Integration of AI-powered Autonomic Management and Control (AMC) Systems and AI-powered Autonomic Online Test Systems during Network Operations" presented in Figure 5 to a Lab Testing context rather (not "in-operations" scenario). In such a case the Framework is being applied to Testing of Autonomics (AMC) software (powered by AI algorithms) implemented at the realm of network management and control architectures and Autonomics implemented with network infrastructures, as characterized by the SUT and NUT indicated.

7.2.4.2 Autonomic test system in a test lab environment

Figure 7 presents the value of having an AI-powered Autonomic Test System that is meant to be used in a Test Lab to test an AMC System for a Scenario in which the targeted autonomics software (autonomic manager components, e.g. ETSI GANA Model Decision-making Elements (DEs)) to be tested by the AI-powered Test system is only in the AMC system and not in the underlying network infrastructure.

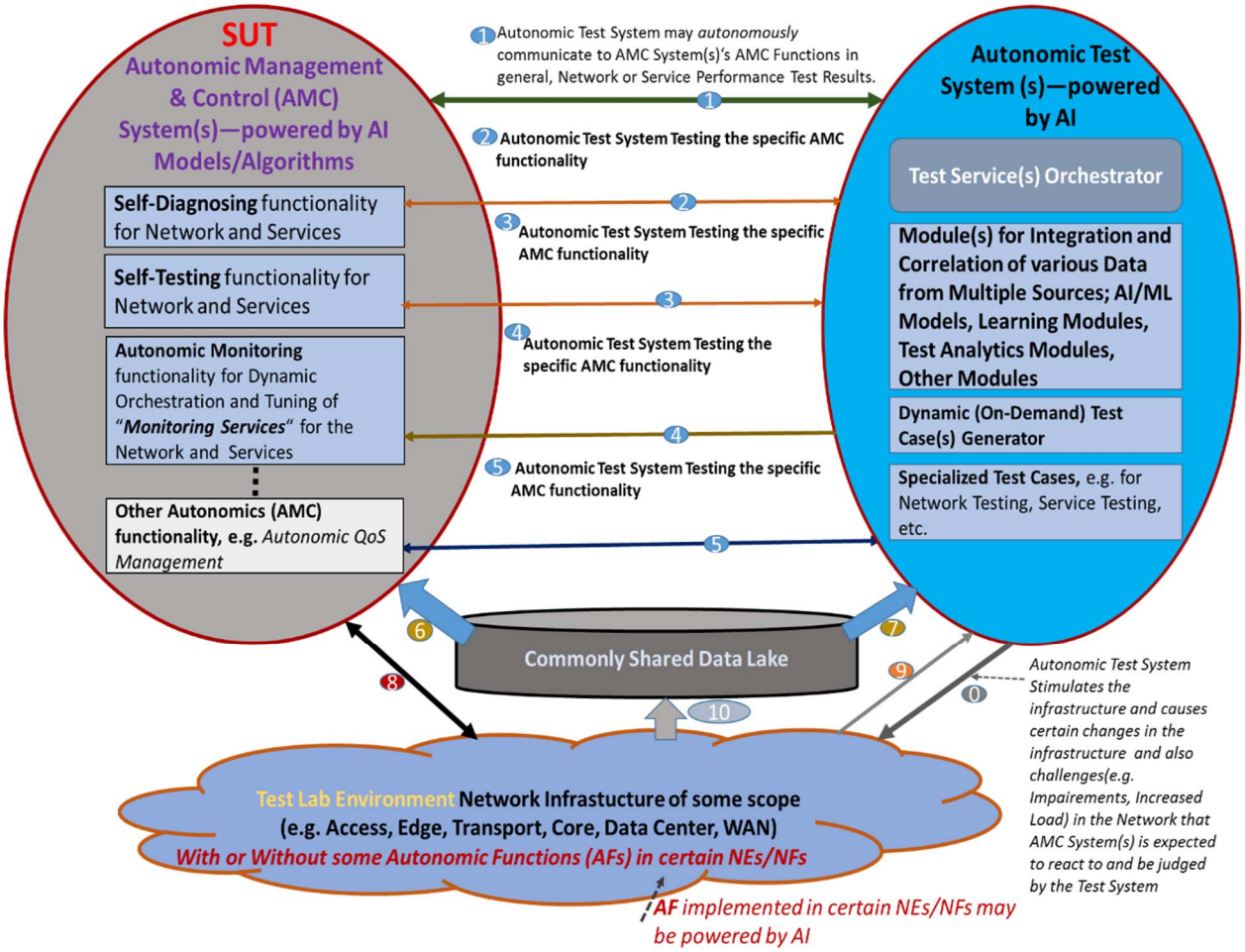

Figure 7: AI-powered Autonomic Test System that is meant to be used in a Test Lab to test an AMC System for a Scenario in which the targeted autonomics software to be tested by the AI-powered Test system is only in the AMC system

Figure 7 presents the value of having an AI-powered Autonomic Test System that is meant to be used in a Test Lab to test an AMC System for a Scenario in which the targeted autonomics software (autonomic manager components, e.g. ETSI GANA Model Decision-making Elements (DEs)) to be tested by the AI-powered Test system is only in the AMC system and not in the underlying network infrastructure.

NOTE: The Framework for AI-powered Autonomic Test System that is meant to be used in a Test Lab to test an AMC System for a Scenario in which the targeted autonomics software to be tested by the AI-powered Test system is only in the AMC system requires to be further detailed and such work may be covered in the same anticipated ETSI Technical Specification (TS) that provides a detailed specification of the concept of an Autonomic Test System or may be covered in a separate document that covers certain instantiations cases for the Autonomic Test System Concept.

7.2.4.3 Autonomic test system in a network environment

Figure 8 presents the value of having an AI-powered Autonomic Test System that is meant be used in the Test Lab to test an AMC System for a Scenario in which the targeted autonomics software (autonomic manager components, e.g. ETSI GANA Model Decision-making Elements (DEs)) to be tested by the AI-powered Autonomic Test system is both, the High Level ("Macro Level") Autonomics in the AMC system and Low Level ("Micro Level") Autonomics software in certain Network Elements/Functions (NEs/NFs) of the underlying network infrastructure.

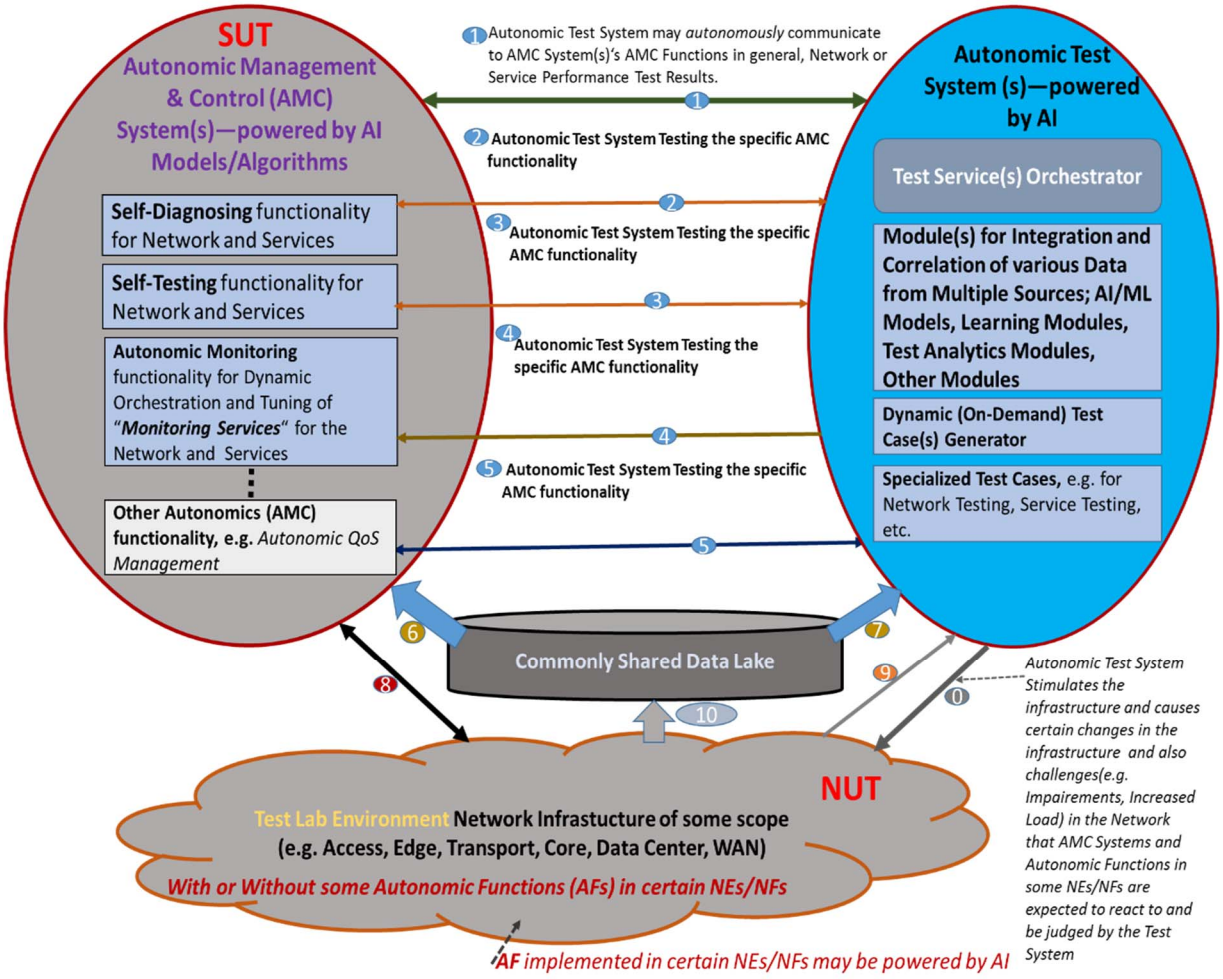

Figure 8: AI-powered Autonomic Test System that is meant to be used in a Test Lab to test an AMC System for a Scenario in which the targeted autonomics software to be tested is in the AMC system and in certain Network Elements/Functions (NEs/NFs) of the underlying network infrastructure

NOTE 1: The Framework for AI-powered Autonomic Test System that is meant to be used in a Test Lab to test an AMC System for a Scenario in which the targeted autonomics software to be tested is in the AMC system and in certain Network Elements/Network Functions (NEs/NFs) of the underlying network infrastructure requires to be further detailed and such work may be covered in the same anticipated ETSI Technical Specification (TS) that provides a detailed specification of the concept of an Autonomic Test System or may be covered in a separate document that covers certain instantiations cases for the Autonomic Test System Concept.

NOTE 2: In the ETSI GANA Model for AMC, the "Macro-Level" autonomics control-loops are referred to as "Slow Control-Loops" operating outside of Network Elements/Functions (NEs/NFs), while the "Micro-Level" autonomics control-loops are referred to as the "Fast Control-Loops" operating within NEs/NFs and may span multiple NEs/NFs for cases where distributed algorithms are used to implement "in-network" distributed control-loops and "Slow Control-Loops" policy-control the "fast control-loops" in network infrastructure.

8 Trustworthiness of AI-enabled test systems

This clause discusses trustworthiness in the context of AI-empowered test systems.

NOTE: This perspective is a Subject for Further Study in the evolution of the present document. However, this clause gives insights on some of the aspects that should be considered in scope as discussed briefly below.

The subject of Trustworthiness of AI with respect to following three aspects is covered [i.19]:

- *it should be lawful, complying with all applicable laws and regulations;*
- *it should be ethical, ensuring adherence to ethical principles and values; and*
- *it should be robust, both from a technical and social perspective.*

Another aspect of trustworthiness in the context of AI-empowered test systems is the area of Autonomic/Autonomous Networking (ANs). In this regard ETSI TC INT is working on Trust and Confidence Building in autonomic systems/networks - an aspect that is very important for Network Operators, CSPs and Enterprises that would deploy autonomic systems/networks as well (ETSI TR 103 629 [i.43]). The Evaluation Methods of Trust and Confidence in ANs are needed by the industry, especially considering all aspects of ETSI GANA Multi-Layer Autonomics and its Multi-Layer AI framework, as well as Cross Domain Federation Aspects for ANs defined by ETSI TS 103 195-2 [i.1] and ETSI TR 103 747 [i.59].

In general, trustworthiness of AI-based test systems reflects the application-dependent criticality regarding functional reliability and security. In order to ensure trustworthiness regarding functional reliability, AI-empowered test systems ensure transparency, explainability and robustness for the full variety of test scenarios. With that background, testing systems should be resilient against accidental disturbances, social engineering and damage. Moreover, reliable algorithmic models used for testing procedures fulfil stringent requirements regarding accuracy, precision, recall and stability. Furthermore, standardized characteristic values foster the verifiability and enhance the interpretability of characteristic network components and functionalities (e.g. regarding path computation elements, network orchestration and network slicing).

In view of security, trustworthy AI-enabled test systems are protected against attack vectors and mitigate information risks. With that background, security protection goals include integrity, authenticity, and availability of training data for Machine Learning algorithms of testing systems.

9 Standardization landscape on the use of AI in testing

NOTE: This perspective on is a Subject for Further Study in the evolution of the present document. In the further study activities of other SDOs are worthy to reflect upon, as well as relating above clauses covered in the present document to other initiatives found elsewhere.

However, this clause gives insights into some of the aspects that should be considered in scope as discussed briefly below:

- [i.8] discusses European Commission (EC)'s Recommendations on Testing and Certification of AI and mapping with ETSI TC INT AI-Support System. As such, the ongoing work in ETSI is aligned with EC Recommendations on AI, in considering aspects such as Standardisable metrics for measurements and assessments in testing/certifying AI models; Methodologies and customizable frameworks for test system providers.
- Able frameworks for test system providers; Support for AI Test Centers and Certification Authorities.

- Clause 10 of the present document provides more insights that point to the Standardization landscape being pursued by the joint work ETSI TC INT and ETSI TC MTS, as the two TCs are the home for testing AI models and AI systems, in alignment with the EC.

In general, standardized risk management processes of AI-based testing applications are imposed to identify, analyse, assess and evaluate risks and carry out conformity assessment procedures. Regarding AI methods in AI empowered testing systems, additional requirements for conformity assessment procedures as well as monitoring should be taken into account, for example with regard to cybersecurity certification. In case of high risk applications, AI in AI-based testing systems should comply with European Harmonised Standards or undergo conformity assessment procedures.

10 Outlook on further work

AI/ML technologies are increasingly being incorporated in the design of various systems and ICT networks as discussed in various sources in literature and in emerging and ongoing standardization activities. The ETSI White Paper No.34 [i.57], presents a good outlook on standardization activities on AI in ETSI and outside of ETSI. Some of the insights are as follows:

- ETSI addresses key AI requirements including:
  - Leverage AI for ICT network optimization.
  - Ensure reliability through testing of systems using AI.
  - Manage and characterize data used by AI, e.g. in IoT.
- ETSI work is aligned with EC Recommendations on AI and seeks to continue addressing the following aspects (among the various topics of importance to the industry):
  - Standardisable metrics for measurements and assessments in testing/certifying AI models in Autonomic/Autonomous Networks (ANs) and their AMC components.
  - Methodologies and customisable frameworks for test system providers.
  - Support for AI test centers and certification authorities.

At this point in the trends on AI/ML incorporation in systems and ICT network infrastructures, notably in autonomic and autonomous systems and ICT networks, the topic of "Artificial Intelligence (AI) in Test Systems and Testing of AI Models" is increasingly a "hot" topic because test methodologies, frameworks and standards are critically required by the industry in order to address the question of testing and certification of AI Models, Components, Systems and AI-powered ICT Networks. The present document has focused on the subject of "Use and Benefits of AI Technologies in Testing". A new concept of Autonomic Test Systems has emerged in ETSI for testing complex systems that include AI-enabled systems and networks as described in the present document. As such there is a need for applying autonomics & AI in Test Systems due to rising complexity of the System and Network Under Test (SUT/NUT).

The present document is expected to be complemented by other documents targeted to be produced in the ongoing joint work of ETSI TC INT and ETSI TC MTS on the overall topic "Artificial Intelligence (AI) in Test Systems and Testing of AI Models".

The following are the aspects that belong to Further Work as the activities on this standardization area progress into the far future in ETSI:

1) Evolution of the present document:
   - Aspects marked with the NOTE that says "This perspective is a subject for further study in the evolution of the present document" need to be considered in the evolution of the present document in its next iteration.
2) Certification Methods and Approaches for AI Models, Components, Systems and AI-powered ICT Networks:
   - While there is now ongoing work on another TR document within the scope of the present document, it is expected that this topic will take quite some efforts into the far future and possibly separate documents in form of Technical Reports and Technical Specifications will be produced.

3) Types of Standardisable Metrics for Testing and Certification of AI Models:

   - While some work on defining Standardisable Metrics for Testing and Certification of AI Models of Autonomic Components/Systems (including Autonomic ICT Networks) has been started in the present document, is expected to continue.

   - And so, continued work is expected on Definition of Standardisable Metrics for Autonomic/Autonomous Networks (ANs) and other classes of AI systems that build the foundation for their certification. The same applies to the need for Standardisable Metrics for Testing and Certification of AI Models meant for other application domains (outside the scope of Autonomic Components/Systems), work in this area is expected to commence in ETSI on these aspects as well. Because Metrics form the Basis for Test and Certification of AI.

4) Introducing the concept of an AI-Support System (AI-SS) in testing and certification life cycle:

   - It is expected that the concept of AI-Support System (AI-SS) should be further developed using a dedicated Technical Specification (TS), and also the associated AI-Support Framework should be used to drive PoCs (Proof-of-Concepts) meant to support the industry in applying the Framework.

5) Autonomic Test System (ATS) Concept and its Instantiation Cases for various Application Areas (i.e. types of SUTs or NUTs:

   - While the present document provides a draft detailed description of the concept of an Autonomic Test System (ATS), a much more detailed specification is planned for such that a Technical Specification (TS) should be developed in ETSI as follow up to this work reported in this present document.

   - Also, the various scenarios of the "Instantiation case of the ATS concept that considers Human-in-the-Loop in the ATS's Operations" need to be developed and elaborated for the benefit of the relevant stakeholders.

   - Also, the various scenarios of the "Instantiation case of the ATS concept that does not necessarily consider human involvement in the ATS's operations (i.e. the case of Self-Testing capability of a System)" need to be developed and elaborated for the benefit of the relevant stakeholders.

   - Then PoCs on the Instantiation Cases of the ATS concept should be launched by the industry.

6) Integration of AI-powered Autonomic Management and Control (AMC) Systems with an ATS as AI-powered Autonomic Online Test System:

   - Further work on Specification of the "The Desirable Framework for Integration of AI-powered Autonomic Management and Control (AMC) Systems and AI-powered Autonomic Online Test Systems during Network Operations" should be carried out and a dedicated Technical Specification or Technical Report can be developed to elaborate the Framework and its interfaces.

7) AI-powered Autonomic Test System that is meant to be used in a Test Lab to test an AMC System for a Scenario in which the targeted autonomics software to be tested by the AI-powered Test system is only in the AMC system:

   - Further work on Specification of the scenario should be carried out and a dedicated Technical Specification can be developed to elaborate the Framework and its interfaces.

8) AI-powered Autonomic Test System that is meant to be used in a Test Lab to test an AMC System for a Scenario in which the targeted autonomics software to be tested is in the AMC system and in certain Network Elements/Functions (NEs/NFs) of the underlying network infrastructure:

   - Further work on Specification of the scenario should be carried out and a dedicated Technical Specification can be developed to elaborate the Framework and its interfaces.

9) Building an Ecosystem for Certification Labs for AI models and AI systems: ETSI may play a key role in this aspect into the future.

10) The ongoing work on ETSI TR on "Testing of AI, with Definitions of Quality Metrics":

   - Further work on is still ongoing, and the result of that work is likely to further trigger the creation of various ETSI Technical Specifications dedicated to specific types of AI and Testing Requirements and Methods.

Annex A: Bibliography

- Böhme, M., Pham, V. T., Nguyen, M. D., & Roychoudhury, A. (2017, October): "Directed greybox fuzzing". In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 2329-2344).
- Böttinger, K., Godefroid, P. & Singh, R. (2018, May): "Deep reinforcement fuzzing". In 2018 IEEE Security and Privacy Workshops (SPW) (pp. 116-122). IEEE.
- Kuznetsov, A., Yeromin, Y., Shapoval, O., Chernov, K., Popova, M., & Serdukov, K. (2019, July): "Automated software vulnerability testing using deep learning methods". In 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON) (pp. 837-841). IEEE.
- Ali, N. bin et al. (2019): "On the search for industry-relevant regression testing research", Empirical Software Engineering, 24(4), pp. 2020-2055. doi: 10.1007/s10664-018-9670-1.
- de Castro-Cabrera, M. del C., García-Dominguez, A. and Medina-Bulo, I. (2020): "Trends in prioritization of test cases: 2017-2019", in Proceedings of the 35th Annual ACM Symposium on Applied Computing. New York, NY, USA: Association for Computing Machinery (SAC '20), pp. 2005-2011. doi: 10.1145/3341105.3374036.
- Elbaum, S., Rothermel, G. and Penix, J. (2014): "Techniques for improving regression testing in continuous integration development environments", in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York, NY, USA: Association for Computing Machinery (FSE 2014), pp. 235-245. doi: 10.1145/2635868.2635910.
- Khatibsyarbini, M. et al. (2018): "Test case prioritization approaches in regression testing: A systematic literature review", Information and Software Technology, 93, pp. 74-93. doi: 10.1016/j.infsof.2017.08.014.
- Marijan, D., Gotlieb, A. and Sen, S. (2013): "Test Case Prioritization for Continuous Regression Testing: An Industrial Case Study", in 2013 IEEE International Conference on Software Maintenance. 2013 IEEE International Conference on Software Maintenance, pp. 540-543. doi: 10.1109/ICSM.2013.91.
- Pradhan, D. et al. (2018): "REMAP: Using Rule Mining and Multi-objective Search for Dynamic Test Case Prioritization", in 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST). 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST), pp. 46-57. doi: 10.1109/ICST.2018.00015.
- ISO/IEC TR 29119-11 (2020-11): "Software and systems engineering -- Software testing -- Part 11: Guidelines on the testing of AI-based systems".

History

Document history		
V1.1.1	June 2022	Publication