TOWARDS AUTONOMOUS OPEN RADIO ACCESS NETWORKS

Adrian KLIKS[1], Marcin DRYJANSKI[2], Vishnu RAM[3], Leon WONG[4], Paul HARVEY[5]
[1]Poznan University of Technology, [2]Rimedo Labs, [3]Independent Expert, [4]Rakuten Mobile, [5]University of Glasgow

NOTE: Corresponding author: Adrian KLIKS, e-mail: adrian.kliks@put.poznan.pl

Abstract *– In this paper we give an overview of an open disaggregated network architecture based on an Open Radio Access Network (O-RAN), including the current work from standards bodies and industry bodies in this area. Based on this architecture, a framework for the automation of xApp development and deployment is proposed. This is then aligned with the key concepts described in ITU-T in terms of the evolution, experimentation, and adaptation of controllers. The various steps in such an aligned workflow, including design, validation, and deployment of xApps, are discussed, and use case examples are provided to illustrate further our position regarding the mechanisms needed to achieve automation.*

Keywords – 5G, 6G, network automation, Open RAN, O-RAN, virtualization, xApp design

1. INTRODUCTION

For the last few decades, telecommunication networks have been focused on connecting people. As the years have progressed, this connectivity has grown from human-orientated communication, such as web browsing, to include data-centric communication or machine-centric communication, as found in IoT device communication. To support this expanded role, telecommunication networks have continuously been innovating in order to meet current needs. Thanks to over 15 years of innovation, partial or full production deployments of virtualized end-to-end hardware infrastructure is now a reality. Most notably, this includes the virtualization of the Radio Access Network, known as (RAN).

In order to achieve interoperability for the use and development of management and operation software in the context of a virtualized RAN, the community is embracing the concept of an Open RAN (O-RAN) architecture, see Section 2. Here, control and monitoring functionality is encapsulated in either highly responsive software *xApps* or higher-level strategic software *rApps*. Each application is then used to drive an associated runtime referred to as the RAN Intelligent Controller (RIC). This way, engineers are presented with a clear way in which to design control and monitoring functionalities for RAN in software.

Despite the clear conceptual architecture for O-RAN using which xApps can be created, the design, implementation, testing, maintenance, and deployment of the xApp is the responsibility of the developer. Furthermore, there is currently no common RIC implementation, meaning that xApp developers must design for a given platform, limiting reuse and deployment on other RIC platforms and precluding mass deployment. Consequently, in order to achieve widespread adoption of xApp-driven control of the RAN, it is necessary to introduce automation to the adaptation of the xApp both at the RIC platform level that it operates upon as well as in the RAN that it is supposed to operate.

This work proposes an approach to achieve automation in the lifecycle of xApps. Thus, starting from the use case, requirements, design of the xApp, validation of the xApp, and finally, deployment in networks, the steps in the lifecycle are analyzed. The key considerations and challenges in achieving this automation are called out.

2. OPEN RAN OVERVIEW

The open RAN concept is based on the following principles: open interfaces, functional RAN disaggregation, hardware-software split, and native data-driven intelligence brought by the RIC concept [1]. The following section provides an overview of the O-RAN architecture, introduces RIC, xApps and rApps, and discusses standardization developments.

2.1 O-RAN architecture

Open RAN architecture, along with its building blocks with their functionalities and open interfaces, is standardized by the O-RAN ALLIANCE and is shown in Fig. 1 [2].

The O-RAN architecture adopts the 3GPP-based Higher-Level Split (HLS, or split 2) and Lower-Level Split (LLS, or split 7.2) building on the disaggregated base station structure that divides its functionality into a Central Unit (CU), a Distributed Unit (DU), and a Radio Unit (RU). A CU is further split into the control plane (CU-CP) and the user plane (CU-UP). In O-RAN language, those are prefixed with O-, namely O-CU-UP, O-CU-CP, O-DU, and O-RU, to refer to the fact that they are 3GPP-based functionalities adapted to O-RAN architecture.

Besides the functional split of a gNB, the O-RAN architecture is also defining a concept of the RAN Intel-

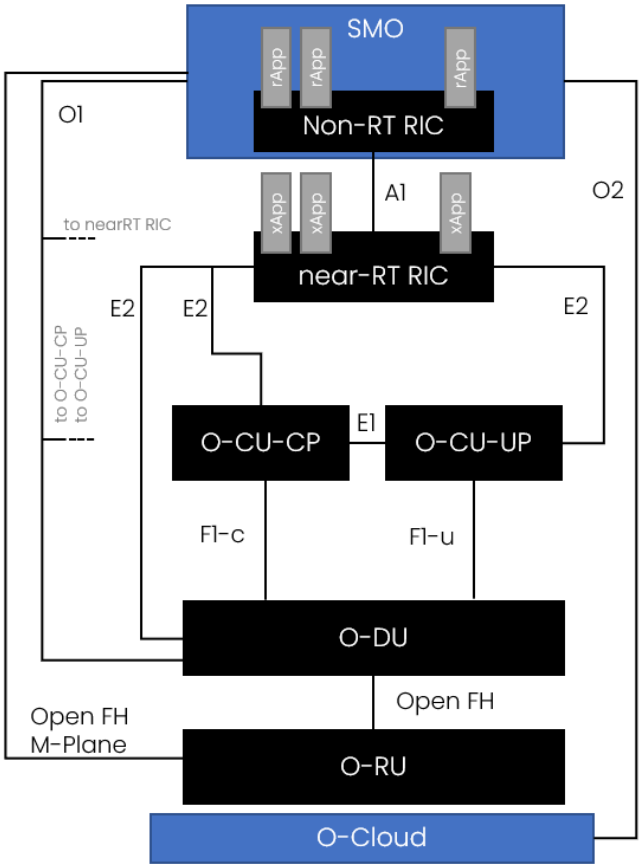

Fig. 1 – O-RAN architecture

ligent Controller (RIC), abstracting out RAN control and monitoring from a base station. RIC is further split into two logical entities, namely near-real-time (Near-RT) RIC and Non-RT RIC. Those serve as platforms for external applications aiming at radio network optimization, respectively, xApps and rApps, operating in different timescales and scopes, as discussed in the following sections.

To complete the picture, O-RAN architecture also covers: O-Cloud i.e., a cloud computing platform on which the virtual elements (like, O-CU, O-DU, or Near-RT RIC) can be deployed; and Service Management and Orchestration (SMO), the management platform, with one of the functions being Non-RT RIC.

The above-mentioned elements are connected via the O-RAN-specified interfaces. They could be split into three types:

- RAN-internal, namely: Open Front-Haul (OFH), an eCPRI-based interface between O-DU and O-RU to transfer I/Q samples; E1 and F1, 3GPP-based interfaces defined for HLS, between CU-CP and CU-UP and CUs and DU, respectively.
- Control interfaces, namely: A1, between Non-RT RIC and Near-RT RIC for policy management, enrichment information transfer, and ML models updates; and E2, between Near-RT RIC and RAN nodes serving as a control loop to execute commands and provide measurements from the O-CU and O-DU nodes.
- Management interfaces (also called Failure, Configuration, Accounting, Performance, Security (FCAPS)), namely: O1, a management interface for all RAN elements but O-RU; O2, interface for O-Cloud platform resources and workload management (e.g., scaling up/down resources); OFH M-Plane, the management interface for the O-RU.

2.2 Non-RT RIC vs. Near-RT RIC

As mentioned, the RIC is split into two following logical entities as shown in Fig. 2.

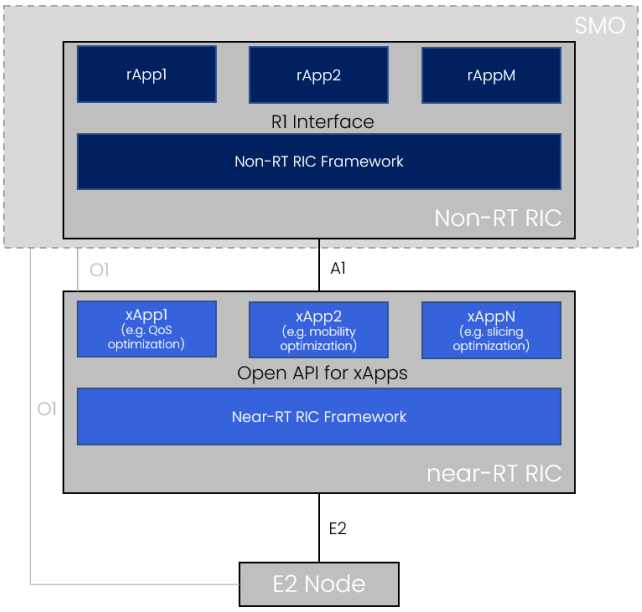

Fig. 2 – Near-RT RIC and Non-RT RIC

The **Non-RT RIC's** general task is to support non-real-time network and procedures optimization. The Non-RT RIC is composed of AI/ML model training, and service and policy management, which create the policies to be sent over the A1 interface and rApp management functions. As an input to the Non-RT RIC, besides the measurements and statistics, there is also so-called Enrichment Information (EI), i.e., additional information from network functions, and from external non-network functions, like user priority. The Non-RT RIC is responsible for configuration management, analytics, creation of the AI-based feeds, and provision of the recommendations to the Near-RT RIC [2].

The **Near-RT RIC**, on the contrary, serves as a software platform to allow the xApps to control the RAN. This is supported by the RAN and UE database storing the network state, along with xApp management, security, and conflict mitigation functions. It enables near real-time control optimization of the RAN elements (called E2 Nodes) via actions sent over the E2 interface [2].

The RICs are accompanied by two control loops.
One is called a **non-real-time control loop** (linked to Non-RT RIC) with a time span larger than one second (» 1s). In this time frame, the policies are set, the RAN analytics are gathered, and the AI/ML models are trained based on long data sets. The time is used to deduct the trends in the network (e.g., traffic pattern over an hour, over a day, over a week, etc.) to optimize the overall RAN behavior.
The Near-RT RIC, in turn, closes the **near-real-time control loop**, which is between ten milliseconds and one second ($>$ 10ms, $<$ 1s). This is the time period for the operation of xApps, deciding on control actions, or producing policy updates, and gathering the key performance measurements. It is a time-scale related to aspects like connection management where e.g., an xApp decides to change the cell that UE is connected to.

2.3 rApps vs xApps

In the architecture we discussed in the previous sections, E2 Nodes (i.e., O-CUs, O-DU) expose parameters and functionalities towards the RIC, which can be used by xApps and rApps to tune the behavior of the radio network. The applications shall behave subject to operator goals, network state, and traffic conditions and may be equipped with Artificial Intelligence (AI) and Machine Learning (ML) algorithms. The main goal for the xApps and rApps is to autonomously adapt to the changes within the network, traffic, and channel and make sure that the Quality of Service (QoS) requirements and Service Level Agreement (SLA) are fulfilled.
Let's now compare the xApps and rApps.
xApps are hosted in the RAN domain. They are applications designed to run on Near-RT RIC, which are required to follow a specified API definition. Each xApp could be designed as one or more microservices. At the point of onboarding, an xApp needs to identify itself and provide information to the Near-RT RIC about the data types it wants to consume and the outputs it will produce. It is independent of the Near-RT RIC and may be provided by a third party. The individual xApp controls a particular RAN functionality exposed by the E2 Node [3]. Examples of xApps are mobility management, admission control, traffic steering, load balancing, etc. It is worth noting that depending on the performance targets selected by the operator (and, more generally, by the end user of the xApp/rApp), some of these examples may be implemented and achieved in various ways. Moreover, let us highlight that time-critical operations and algorithms will be realized in a very short time scale at the E2 Node.
rApps are modular applications designed to run on a Non-RT RIC and sit in the management plane. Their aim is to provide Value-Added Services (VAS) related to RAN optimization and procedure optimization through the Non-RT RIC. Examples of VAS include: policy-based guidance and enrichment information provisioning, performing data analytics, AI/ML training, and inference for RAN optimization or to be used by other rApps, providing recommendations on configuration management actions [4]. Examples of rApps are energy-saving management, capacity, and coverage optimization, or QoE prediction and assurance.
Similarities between xApps and rApps are that:

- they work as independent applications at the Near-RT RIC or Non-RT RIC, respectively;
- they need to fulfill the requirements for open API to be able to communicate with the other part of RIC.

The differences between xApps and rApps are the following:

- an xApp directly controls an actual function within the RAN element, while an rApp is used within the Non-RT RIC framework helping to create policies (i.e. indirectly influences the RAN behavior);
- xApps and rApps work in different time scales with respect to the RIC they work within (i.e. xApps with a control loop in order of tens or hundreds of milliseconds, and rApps with a control loop in order of seconds, minutes or even hours).

Recently, in the research community, another type of application has been considered within the O-RAN context, called dApp. Those, in turn, sit directly at O-CUs or O-DUs and receive real-time data from the RAN, along with E2 from Near-RT RIC, and execute inference and control of lower-layer functionalities, thus enabling stricter timing requirements than xApps and rApps, for such use cases, as beam management and user scheduling [5].[1]

2.4 Standardization and enablers

As already mentioned, the main standardization body for Open RAN is O-RAN ALLIANCE. It is responsible for defining individual parts of the O-RAN Architecture. This includes the use cases, architecture, interface specifications, reference designs, and protocols.
In addition, there has also been considerable work done in other organizations, including SDOs and industry bodies, regarding deployment options, development of xApps and integration (at the policy level), and management of near-real-time RICs.

[1] While a detailed study of Intellectual Property Rights (IPR) is out of the scope of this work, enabling the xApps/rApps delivered by the third-party requires, in general, careful consideration of the IPR issues. It is indeed possible that sophisticated IPR policies applied by contemporary IPR holders in the communications domain may have to be considered during software development and deployment. However, it is possible that the disaggregation of RAN functions as well as the assumed modularity paves the way for the creation of an innovation ecosystem for the third parties.

- The Telecom Infra Project's (TIP) OpenRAN program supports the development of disaggregated and interoperable 5G Radio Access Network (RAN) solutions based on service provider requirements. This aims to continuously improve the performance of the RAN by bringing innovation, automation, and competition. Reference architecture and design are output from this initiative for multiple deployment options of RU, CU and DU. In addition, the TIP OpenRAN RAN Intelligence & Automation (RIA) project focuses on the availability and use of data so that a centralized RAN controller can manage a disaggregated, virtualized, and multivendor RAN. Thus the primary aim is to provide practical solutions for OpenRAN with multiple deployment options, for various use cases, including the integration of AI/ML. While this could be a good starting point for automation, this does not currently address the problem of automation in the lifecycle of xApps.

- The Open Networking Foundation (ONF) works on building open source components for the mobile RAN space, complementing O-RAN's focus on architecture and interfaces by building and trialing O-RAN-compliant open source components as part of its SD-RAN project. Based on the O-RAN architecture, SD-RAN is developing a near-real-time RIC (near-RT-RIC) and a set of exemplar xApps for controlling the RAN. Thus, the primary aim is to accelerate the adoption of the O-RAN architecture and the availability of interoperable O-RAN components.

- The Open Network Automation Platform (ONAP) is a platform for the orchestration, management, and automation of network and edge computing services for network operators, cloud providers, and enterprises. It enables rapid automation of new services and complete lifecycle management for 5G and next-generation networks. ONAP has been enhanced to support A1 Policies. The A1 Policy functions are orchestration and automation functions for non-real-time intelligent management of RAN functions. The integration of A1 allows existing ONAP infrastructure to support non-real-time control of the RAN. Using the A1 interface will facilitate the provision of policies for individual UEs or groups of UEs; monitor and provide basic feedback on policy state from near-real-time RICs.

- The ITU-T Focus Group on Autonomous Networks (FG AN) studied the use cases for autonomous networks. This study centered around the three key concepts of *exploratory evolution*, *online experimentation*, and *dynamic adaptation* of "controllers" to networks. A controller is a workflow, open loop or closed loop composed of modules, integrated into a specific sequence, using interfaces exposed by the modules, which can be developed independently of the network implementations. Architecture components, subsystems, and their interactions which enable the key concepts, were described. This framework provides the fundamental building blocks for autonomy, and by representing xApps in the form of controllers and applying the concepts developed in FG AN, it is possible to achieve a high level of automation in their lifecycle.

2.5 Research in the O-RAN domain

Open RAN has become an attractive research topic, as RAN openness, modularity and disaggregation paved the way for new scientific challenges and opportunities. The functioning of RICs and their internal components (such as the subscription management module, conflict mitigation module, etc.), although partially standardized, constitute a very viable investigation domain. Moreover, the development of xApps/rApps is an interesting investigation topic, as new and sophisticated solutions can be provided that address specific aspects of RAN functions. For example, in [6], the authors discussed ways for efficient spectrum sharing through data-driven dynamic control in real time. Likewise, the paper [7] addresses how policy-based traffic steering can be implemented in future wireless networks. As the impact of artificial intelligence on Open RAN is foreseen to be crucial, in [8], the role of AI/ML tools in xApp design is widely discussed. In that context, it is worth mentioning the great overview of Open RAN and ongoing research activities presented in [9] In our context, it is important to concentrate also on the implementation and, in particular, on automation aspects. In [10], a dedicated framework called OrchestRAN has been discussed, which allows for network automation through orchestrated intelligence. In a similar spirit, in [11], the closed-loop automation in 5G Open RAN was presented, where the authors focus on enabling the optimization of 5G network resources and services in an automated and self-configured manner. Finally, it is worth mentioning the ongoing work in the field of tight cross-layer optimization through application-to-network communication, where one of the examples is the MPEG Server and Network Assisted DASH (SAND) solution [12]. In that context, let us notice that rApps (as entities operating on a longer time scale) of a different kind may be provided to cooperate with the SMO for better network-to-application support. Next, flexible policies down to the near-real-time RICs for better support of SAND-like applications to guarantee the assumed quality of service or experience. Finally, dedicated xApps may be deployed, which improve the network performance in such a way that SAND-line solutions are supported. An interesting approach for using SAND has been discussed in [13], where heterogeneous Quality of Service requirements of different applications in 5G systems are managed in the so-called xStream platform. The au-

thors have proposed the improvement of the SAND limitations by enabling communication between various applications and the 5G system. It paved the way for innovative solutions for various types of traffic, as proved by extensive simulations. Such an approach can be considered for xApp/rApp implementation and simulation. Next, in [14] the new architecture called ARBAT has been proposed. It uses the concept of the Universal Network Device and Unified Cellular Network; moreover, it utilizes the AirHYPE wireless hypervisor, together with the above-mentioned network-user application interaction through the xStream platform as well as the ServiceBRIDGE. Following this idea, in [15], the design of an optimization-based power allocation model to increase effective power efficiency was discussed in the context of guaranteed QoS. The same hypervisor AirHYPE has been considered for designing novel architectural solutions for the upcoming sixth generation of cellular in [16]. A very interesting roadmap paper has been published recently, i.e., [17], where the enabling techniques and recent advancements on 6G-related topics are highlighted, and open problems with possible solutions are discussed.

In June 2022, O-RAN ALLIANCE founded a research task force called the next Generation Research Group (nGRG), which focuses on the research of open and intelligent RAN principles in 6G and future network standards. nGRG work is split into several research streams, including requirements, architecture, AI/ML, security, and research platforms. Supplementing the prior and ongoing research activities, in our paper, we focus on the definition of the automation of the xApp/rApp development and verification process.

3. THE NEED FOR XAPP DEVELOPMENT AND DEPLOYMENT AUTOMATION

As with any emerging technology, current x/rApp implementations are strongly coupled to the use case(s) for which they are created, as well as with vendor-specific RIC implementations. This requires engineers to manually adapt x/rApps from one use case to another and possibly re-implement them for use on different RIC implementations. Such effort prevents reuse, slows development, requires (programming) development skills from domain experts, and ultimately acts as a barrier to O-RAN adoption.

In this section, we discuss existing efforts that seek to achieve autonomy in O-RAN and then present our vision on how to achieve autonomous O-RAN taking xApps as a representative exemplar.

3.1 ITU-T related work

The ITU-T Focus Group on Autonomous Networks (FGAN)[2] has been working towards understanding and describing the elements required for autonomy in networks, especially in the context of the three key concepts of *exploratory evolution*, *online experimentation*, and *dynamic adaptation* [18].

The concept of exploratory evolution introduces the mechanisms and processes of exploration and evolution to adapt a controller in response to changes in the network. Generation of new controllers or update (evolve) of existing controllers to respond to such changes are part of evolution. Validation of controllers and their logic, using simulated and/or real data, may be done before the deployment of controllers in the network. This continuous process, based on monitoring and optimization of deployed controllers in the network, is called real-time responsive experimentation. The adapted and validated controllers are integrated at run time to underlay networks. Thus, dynamic adaptation is the final concept in equipping the network with autonomy and the ability to handle new and hitherto unseen changes in network scenarios.

The current use cases studied in FGAN in this context include analysis-driven evolution in virtualized RAN based on DevOps [19]. Such use cases take advantage of the programmability and interfaces exposed by RAN components in open RANs while allowing developers the opportunity to create applications (e.g., xApps) based on data from RAN. Provisioning and analysis of closed loops at near-real-time RIC allows operators to analyze the data and arrive at the new use case needs at the edge. This can then be used to drive the process of evolution of new xApps to enable the development, instantiation, and deployment of xApps based on the capabilities of various RAN nodes.

An initial proof of concept, done under the initiative of "Build-a-thon 2021" by FG AN produced a demonstration of a YAML-driven docker container-based instantiation of controllers. This was independently supported by an xApp implementation however, integrated automation of the xApp lifecycle was not achieved in FG AN Build-a-thon 2021.

Conceptually, the study of x/rApp lifecycle automation discussed in this work is well aligned with the concepts discussed in standards and industry bodies, including in ITU-T, and can even be considered as a realization of the concepts in the O-RAN context.

3.2 Towards xApp automation

Open RAN architecture creates opportunities for the Mobile Network Operators (MNO) for modular improvement of the access network domain through installation, uninstallation, or upgrade of specific functions,

[2]www.itu.int/en/ITU-T/focusgroups/an/Pages/default.aspx - Accessed 14/10/2022

mainly xApps and rApps. It means that the MNOs should be able to download, test, verify and, depending on the evaluation results, either keep the application or simply roll it back. Moreover, the immediate consequence of such an approach is the need for secure and mature space (domain) from where the software could be purchased by the MNOs.

In a classic model, the MNOs will directly order the implementation of the desired function by one of its collaborators, subcontractors or will outsource it via some software houses. However, another approach is also possible, which is widely known in the domain of applications for mobile phones and regular PC, where tested and certified application can be downloaded from the dedicated stores. Analogously, in the context of xApps and rApps, the usage of reputable and trusted application stores may be beneficial to the MNO.

The xApp/rApp store is highly dependent on the RIC platform deployed underneath, as each RIC may have different rules of coexistence of different applications, conflict mitigation, conformance reasoning, exception handling (and many more). One may see again the analogy to the mobile application stores, which are adjusted to the specific operating systems running on the mobile device. The xApp/rApp provider follows then the rules specified by the RIC provider and delivers the requested application to the store, which, after detailed evaluation and verification, makes it accessible from its online resources and exposes it to the network operators. The latter, being the end users of the xApp/rApp stores, select the most appropriate solution, analyze its performance benchmarks and metrics, and finally install and deploy them in the network. Thus, there are three players in the described model, as illustrated in Fig. 3, mainly: certified (trusted) xApp/rApp providers, xApp/rApp stores, and MNOs.

One may recognize two possible variants of cooperation between the software providers and the stores, which are directly related to how the RIC providers are positioned to the underlying base station software (i.e., O-RAN's O-CU/O-DU software). In the first approach, the RIC is fully associated and dependent on the RAN software and, in consequence, is highly optimized for it. For example, there could be some parameters or pre-standard service models available which are not compliant with the O-RAN ALLIANCE specifications, which would enable more insights and more granular optimization of the base station operation.

Following another way, the RIC is more generic and thus is independent of the base station software, which it will be controlling; in consequence, it is not optimized for it. In such a case, the E2 service models will be following the O-RAN ALLIANCE's specification, and the usability of the specific xApp may be limited to the parameters implemented by the software provider of the underlying O-CU/O-DU.

In the former case (indicated in Fig. 3 in blue), the xApp/rApp provider is also strictly bounded with the certain RIC platform and with the underlying software. Thus, it may deliver software only to the specific store as it is certified for it. Contrarily, in the second approach (marked in green in Fig. 3), the xApp/rApp providers may provide more generic (yet not that effectively optimized) solutions. Clearly, xApp/rApp providers may target both kinds of stores as well. In terms of security and reliability of the xApp/rApp providers, the stores may require dedicated certificates, which may be issued either by the dedicated Certificate Authority (CA) or by any third party yet trusted CA. Once the applications are deployed in the store, the MNO may check them and finally purchase if they fulfill all of their performance, security, and quality requirements.

However, one immediate conclusion is that there is a strong need for automation procedures at various phases of the above process, as it will be described in detail in the following sections. The store owner has to be fully convinced that the application delivered by the xApp/rApp provider fully matches all of the defined requirements of the store, follows all of its guidelines, and is written with good programming practices. Thus, the delivered software has to be checked for its integrity and completeness of all its internal components; it has to be verified from the perspective of potential security issues it could generate in the final system of the MNO. Finally, it has to be checked for its conformance, performance, and dependencies with other applications running on the desired RIC. All of these tests are necessary, yet many others may be specified. It is also important for the MNO to have the possibility to test the performance of the xApp/rApp in the context of its actual use case. This puts a requirement on the xApp/rApp store to have a tight connection to a Digital Twin (DT) [20, 21, 22], where the specific scenario can be emulated with the xApp/rApp under test. In consequence, it is impossible to perform an efficient verification loop (marked in Fig. 3 with dashed line) without automation and proper preparation of the whole xApp/rApp package. Finally, it is worth mentioning here that both the store and the MNO may provide evaluation and performance reports back to the xApp/rApp provider and the store, respectively. Such reports may be periodic or may be sent by request; nevertheless, the presence of the feedback loop has to be considered, as it allows for permanent application improvement based on the delivered reports. Such a procedure should also be automated, at least to some extent.

It is worth mentioning the importance of marketplace ownership in the above context. For the time being, various cases are possible from a business perspective. In the above discussion, we have considered two scenarios, a common marketplace (also called 3rd party xApp store) and a customized one (called a customized xApp store). Although it would be possible from the standardization perspective that the third-party xApp/rApp providers may deploy their software in all available marketplaces, the ownership aspects of the marketplace, if

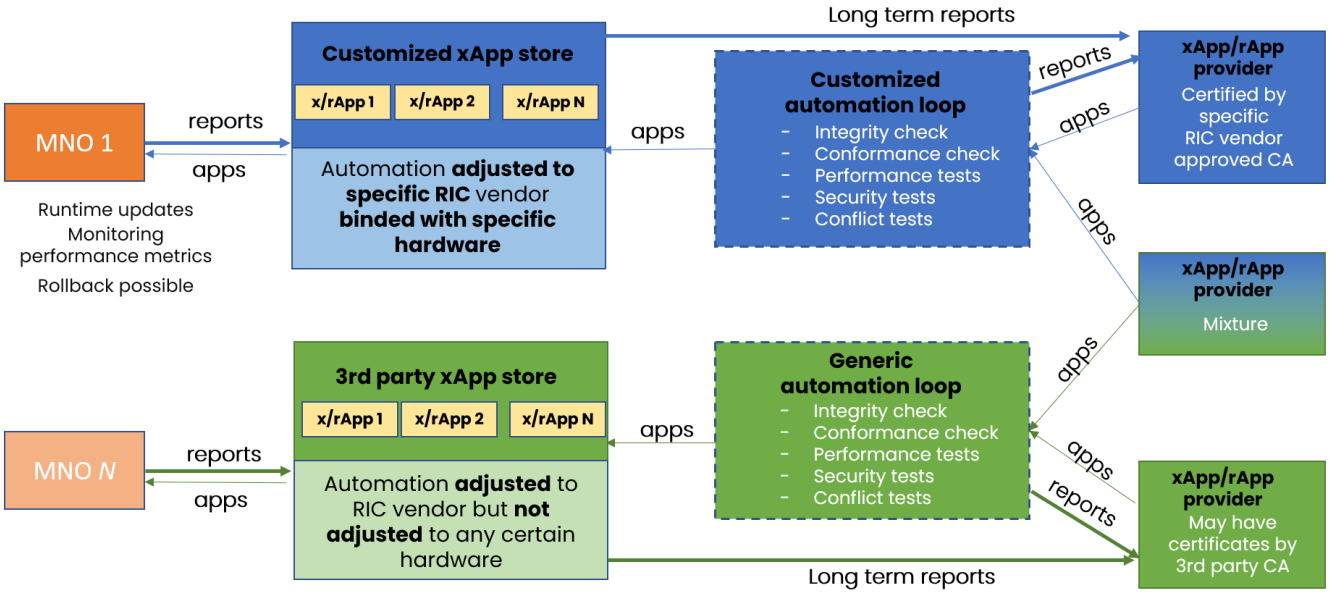

Fig. 3 – Relations between the xApp/rApp providers, store and the network operator; MNO - Mobile Network Operator, CA - Certificate Authority

not considered carefully, may somehow impact in various ways the performance of open RAN solutions.

Finally, let us comment on the impact of the underlying hardware (i.e., E2 nodes in the O-RAN architecture) on the template-based xApp/rApp delivery. One has to notice that by standardization of all interfaces, protocols, messages, service models, and modules on the RIC, the whole concept is, to some extent, hardware-independent. The software, i.e., xApps/rApps, will receive all network performance parameters by subscribing to the appropriate services exposed by RICs. Thus, as long as the concept is implemented completely, the xApp/rApp implementation will be hardware-independent. However, every base station (E2 node) operating on specific hardware will be characterized by some performance metrics (such as the frequency of key performance indicators delivery from the base station to the RIC). In consequence, the underlying hardware may have an indirect impact on the performance of the xApp/rApp, as, for example, not all of the required data by the xApp/rApp will be available. In general, however, the xApp/rApp can be understood as strongly independent from the underlying hardware applied in the E2 nodes.

3.3 Three perspectives in xApp/rApp automation

To push the concept of xApps and rApps closer to practical implementation, each phase of the application lifecycle has to be fully automated. It includes not only the design and implementation stage but also verification and testing by the store and purchasing, downloading, and deployment by the MNO in the real network. Processing any of the above steps manually would be highly inefficient in terms of consumed time, utilized human and computing resources, and the total price at the end. Thus, one may identify three, to some extent, excluding perspectives on xApp/rApp design and deployment process.

First, from the perspective of the xApp/rApp developer, the main effort should be made the investigation of new technical and algorithmic solutions for specific problems in wireless networks. It will be mainly possible if the whole programming environment guarantees efficient tools for code verification, performance evaluation, and advanced debugging. It should be assumed that the xApp/rApp developer will be, in general, not an in-depth specialist in the software environment offered by the xApp/rApp store associated with the specific RIC provider. The process of uploading a new application to the xApp/rApp market shall be seamless for the developer, as the fundamental assumption is that the developer follows the well-established programming rules accurately specified by the xApp/rApp store and RIC provider.

Next, from the store's point of view, a similar observation can be made. The store owner should be in possession of such software tools (environment) that will allow smooth and easy uploading of new applications from external developers. The store owner does not necessarily need to be an algorithmic expert in the field of wireless communications, as its key role is to manage the entire application efficiently. On the contrary, the xApp/rApp store should be prepared not only to process new applications but also to permanently moni-

tor the status and dependencies with other, already existing applications available on the market. Based on detected conflicts or potential dangerous dependencies among various applications, the store should be able to automatically inform interested xApp/rApp developers about the encountered conflicts or possibly dangerous relations. Finally, the store shall be able to perform permanent and on-request effectiveness tests of any set of uploaded applications and react accordingly to the observed performance metrics (benchmarks).

In Fig. 4, the three typical phases of the development and deployment of new xApp / rApp are illustrated from a high-level perspective. When the external xApp/rApp provider decides to develop a new application, he needs to download the appropriate template of such an application and follow the store and RIC rule of application implementation, as will be discussed in Section 4. Then, applying any of the company-suitable programming principles (such as following the continuous integration continuous development paradigm, known as CI/CD), the main programming effort is made. In this phase, local verification, testing, and improvement have to be done based on the installed simulation environment. Once the application is ready and verified, it can be uploaded to the store, where the new application will first be validated and verified (for example, completeness), as defined in Section 5. After that, the performance of the newly uploaded application will be measured in various ways (such as following some predefined benchmark procedures), and its functionality performance metrics should be monitored to update the developer in case of any issues. Finally, if the application passes all of the verification tests, it will be exposed to the MNO, who can request it, and after downloading, verify it in its own environment. As will be analyzed in Section 6, such an environment could be realized in the form of a digital twin, which can mimic the architecture of the real network and the use case. Once the verification tests in the virtual environment pass positively, the MNO may decide to deploy it in the true network. At the same time, it may provide updates to the store (and to the developer) about the truly achieved key performance indicators.

4. TEMPLATE-BASED XAPP DESIGN

One of the immediate observations that can be made here is the need for dedicated software for the development of specific xApps/rApps. The presence of Software Development Kits (SDK) together with specified Application Programming Interfaces (API) is of utmost need to allow efficient xApp/rApp development by third-party companies. In practice, such a set of SDKs and APIs will be strongly related to the considered RIC offered by a certain provider. As it will support effective xApp/rApp development by the application providers, such an approach will entail portability issues; that is, the application adjusted to the specific RIC will probably not be compliant with other RIC platforms. Various solutions could be applied here, such as the implementation of xApps/rApps in native code; however, such discussion is out of the scope of this work.

As stated above, to upload any newly developed application to the xApp/rApp store, it has to fulfill various requirements specified by the store itself. To make this process efficient and maximally autonomous, the template-based approach could be applied, where the store releases technical guidelines on how to implement store-compliant software packages. These technical guidelines should contain not only the rules describing the particular steps for uploading the new xApp/rApp to the store or the pre-upload checklist but should also specify the whole format of the final package of files to be uploaded. It could include, for example, the detailed template for the project structure and all necessary project items. The developer will then know which files are obligatory and which play a supportive role. One can imagine that, depending on the applied software architecture, like model-view-controller or any other, some specific types of files may be stored in dedicated folders and forms. For example, various resources like images, icons, sets of strings or constant variables, language packages, or even trained models for the used AI tools could have dedicated folders in the project structure; the internal databases could be another example of the resource that require separate storage rules. Moreover, the developer shall follow the rules accepted by the store and/or RIC provider related to, e.g., applied ways for exception handling, usage of threads and processes, the ways for handling permissions, as well as the way for expressing dependencies with other xApps/rApps and the underlying network elements.

By default, the application uploaded to the store will be implemented by a third-party company (i.e., external to the store and the RIC provider). In that context, the final store client (i.e., the MNO) should be aware of the xApp/rApp authorship, the type of the application, its version, and the type of license. Moreover, the MNO would be interested in knowing what necessary permissions should be granted to the application to work; in other words, what are the required input arguments (such as number and location of active user equipment, UE, the receives signal strengths by the base stations, type of generated traffic by the UEs or applied modulation and coding schemes to certain UEs, etc.). The xApp/rApp provider shall also explicitly inform the MNO about the expected output parameters and list all potentially modified or affected features of the controlled network. Next, since the xApp/rApp will, in most cases, not run independently from other applications of that kind, all dependencies with other typical applications should be specified to detect and mitigate prospective conflicts. It seems obvious that most or at least some of the above parameters and pieces of information should

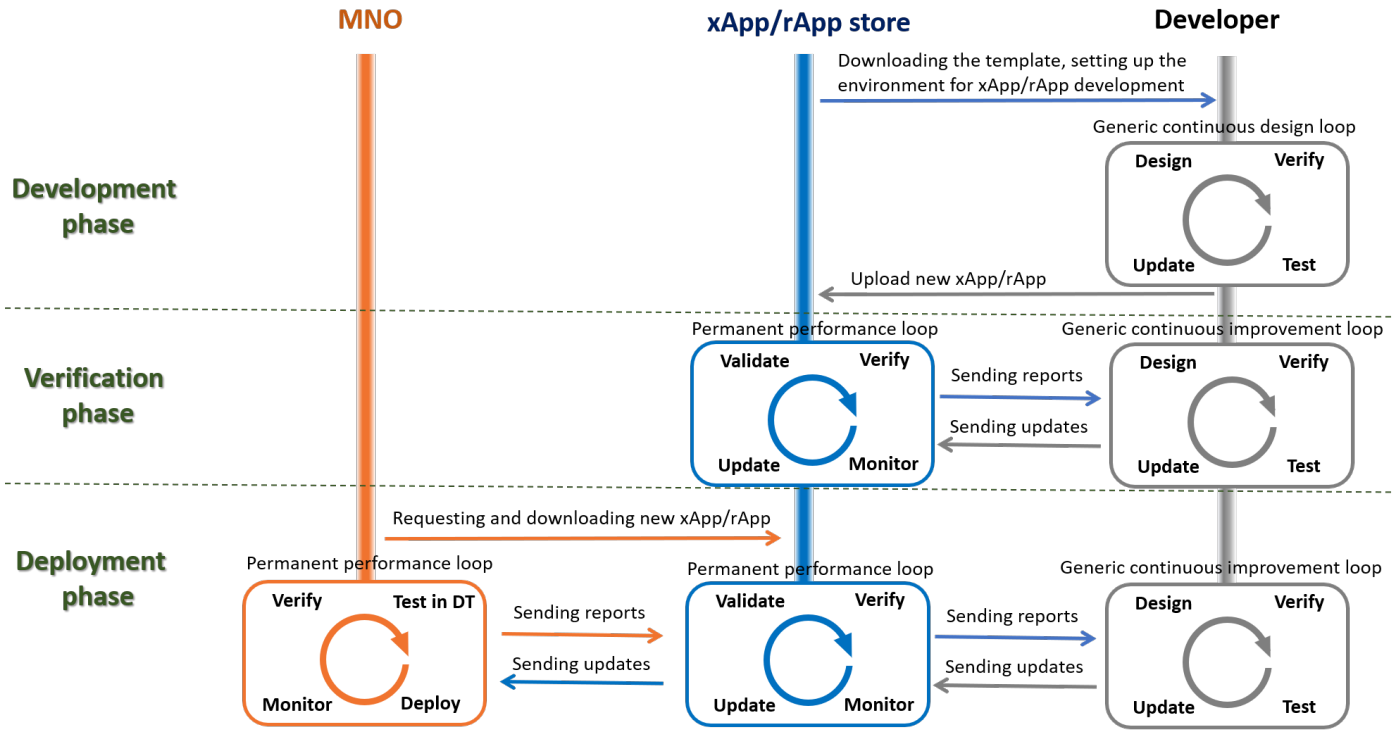

Fig. 4 – Key phases in automated xApp/rApp development, upload and deployment

be standardized by authorized standardization bodies. Nevertheless, such kind of information forms a collection of metadata, which could also be gathered in the dedicated file(s), such as in a manifest file when following the popular approach. In Listing 1 in A, we propose the content of the prospective manifest file that could be applied as a template for the delivery of xApp/rApp. Let us briefly summarize the key features of the xApp/rApp design from the perspective of effective automation of the whole process:

- the template-based application design can be used by both, the customized and the third-party xApp/rApp providers;
- there is a strong need for the standardized API/SDK, which should be secure, stable, upgradeable, etc.;
- the template-based xApp/rApp design should promote flexible, secure, and continuous implementation;
- there should be a specific project structure, which assumes the presence of, e.g., dedicated space for specific modules (resources), application of the common ways for exceptions handling, threads, and resource management, etc.;
- the process of template-based application design should be as much as possible independent of the target application, including, e.g., the topology of the controlled wireless networks, type of underlying hardware facilities, and ways of implementation (e.g., server-less or server-based);
- the application design should be prepared for intelligent coexistence of various xApps;
- the application design should be associated with various service models (E2SM); in other words, it should be compliant with various O-RAN ALLIANCE standard versions (guaranteeing backward compatibility and interoperability);
- similarly to the above, the application design should be associated with the A1-interface to consume standardized policies, which are O-RAN ALLIANCE compliant (again guaranteeing backward compatibility and interoperability between RIC platforms);
- following the *best practices of application design*, the template-based xApp/rApp implementation should allow automatic package generation and upload to the store.

5. XAPP EXPERIMENTATION AND VERIFICATION

Once the xApp/rApp is completely designed and tested locally, it should be delivered to the store with the aim of offering it for download by any interested stakeholder. As can be envisaged, the whole uploading process should

be automated, but it should guarantee precise and accurate verification of the new contribution by applying numerous tests. These tests include various conformance tests (e.g., conformance with the expected project structure, conformance with the applied subscription models to the events occurring in the network, etc.), and compliance tests (e.g., compliance with the O-RAN ALLIANCE standards), as well as performance tests (i.e., how various resources are utilized and consumed, such as processing power, memory, etc.; but also what is the performance of the application in various benchmarks). Finally, each application shall pass through advanced security tests and through dedicated AI-oriented tests, following, e.g., the ITU-T Y.3176 recommendations [23]. Such AI-focused tests should verify the robustness of the applied AI tools against various focused attacks on AI models and algorithms and the effectiveness of applied tools in critical situations.

A separate aspect from the store's point of view is to find a way to obtain fast, yet reliable, accurate, and precise verification of the true performance of the newly delivered xApp/rApp. As the application provider performs numerous internal tests locally, such a way of performance verification will not be detailed enough from the MNO point of view. The MNO will most likely search for numerous additional information about any new piece of software they are considering for installation. As some performance (benchmark) metrics will be available openly to every interested user, the store may offer advanced performance test results as one of its paid offers. To achieve this the store should have facilities and ways for an independent and, to some extent, standardized way for true performance evaluation of the selected xApp/rApp. The test could be performed in a dedicated sandbox, which will allow for the precise identification of any safety issues or symptoms of inefficiency. A sandbox can be interpreted as an isolated testing environment that will create opportunities for advanced testing and verification of the xApps/rApps without affecting the network and/or other applications or running systems. Moreover, following the concept of Digital Twins (DT) [20, 21, 22], the store can run xApp/rApp on the DT of an existing network to check the true impact of the application tested. Let us note that such a verification process may be multiphase by nature. First, the store may run the new xApp/rApp solely in the sandbox (or DT domain) to check its performance. Once it is done, mutual dependencies and relations with other xApps/rApps should be tested again within the dedicated sandbox or in the DT domain. Finally, short-term performance evaluation provides a view of the true functioning of the application. However, it is evident that long-term performance evaluation is also very important. Thus, the store should be able to run long-term simulations to provide detailed benchmark results of this kind.

Let us summarize the prospective steps of the automated application upload procedure:

- Step I
  - Execution of the completeness check (i.e., if all necessary files and modules are included in the package);
  - execution of the conformance check (format, type, etc.).
- Step II
  - Setup of the priorities and hierarchy of the new xApp/rApp in the generic network environment;
  - establishing relations between the applications (mandatory, optional) based on the exposed list of addressed events the application is responding to.
- Step III
  - Verification of the rules for accessing the databases
  - verification of the required permissions and of the impact of modified parameters;
  - verification of the APIs within the RIC (e.g., using the API Enablement element from the Near-RT RIC).
- Step IV
  - Identification of all prospective conflicts between applications.
  - classification of conflicts to various classes under the management of the conflict mitigation module in the RIC;
  - identification of any critical conflicts
- Step V
  - Verification of numerous security tests and performance tests based on data provided by the developer;
  - execution of independent security tests – generic tests and xApp/rApp specific tests.
- Step VI
  - Execution of the long-term performance tests (i.e., start monitoring how xApp/rApps behaves in the network in the longer time scale);
  - generation of generic reports delivered to the provider and for the customer;
  - generation of detailed reports available after payment.

6. XAPP CUSTOMIZATION AND DEPLOYMENT

Very similar observations (to those associated with the xApp/rApp store) can be made while analyzing the automation in the final deployment stage. The MNO shall be able to easily and safely buy the application of interest, test and verify it on its own environment, customize it to its needs, and, if all tests pass positively, finally deploy it in the network. Again, the application of the DT of the true underlying network of the MNO can be considered an effective and secure way for the true verification of the downloaded xApp/rApp in a real environment. However, as the installation of any new software may cause some unpredictable consequences, the MNO shall have the possibility to easily uninstall the application and roll back to the setup before its first run.

One very interesting option that appears to the MNOs, is the automated generation of new xApps/rApps, customized to the MNO's needs, based on the observations of the behavior of already installed applications. However, this item has been left for further study and is not a subject of this paper.

In Fig. 5, we have illustrated the concept of usage of the digital twin concept for testing purposes by the MNO. One may observe two layers, physical twin and digital twin, where the former reflects the true network physically deployed in a real environment, and the latter, its digital representation in the virtual world. One can notice that in the physical twin layer, the MNO performs typical monitoring activities for the network performance, it analyses the status of the network and also the impact of really installed applications (xApps, in this example, mobile handover xApp, MHO, and load balancing xApp, LB). Based on such analysis, it will make decisions on network configuration. In parallel, however, there is a digital twin running in the virtual domain. It is fed with information from the real world and performs short and long-term analysis of the behavior of the network operating in one or more different configurations. In Fig. 5, there are three parallel sublayers in the digital twin layer, where three sets of xApps are tested for the real data originating from the physical layer. Mainly, in configuration A, MHO xApp and LB xApp are tested together with the traffic steering xApp (TS xApp). In the second case, Quality-of-Service-based Resource Allocation (QRA) xApp is tested with MHO xApp and LB xApp, whereas in the third case, TS xApp without LB xApp is verified. Based on such analyses, the MNO may decide on the potential deployment of any new xApp in a real network. Please note that as discussed in Section 5, an analogous approach may be applied by the xApp/rApp store manager.

Similar to the steps described in the previous section related to the store, let us concisely analyze the possible steps in the deployment of any new application in the O-RAN network.

- Step I
  - Installation of the application in the DT domain;
  - execution of initial conformance, security, etc. tests specified by the MNO;
  - establishing relations between certain applications (mandatory, optional) already deployed, selection of permissions.
- Step II
  - Definition of the experimentation setup of the DT network (topology, structure, performance metrics);
  - execution of the advanced, yet short-term, performance tests.
- Step III
  - Initial performance results analysis;
  - statistics presentation and reasoning for simple performance tests.
- Step IV
  - Execution of the advanced and long-term performance tests;
  - active reasoning – observations of the results of specific xApps/rApps
- Step V
  - Performing the final evaluation;
  - deployment in the true network if all tests passed positively
- Step VI (continuous)
  - Performing a permanent evaluation to detect any long-term negative effects (both in the real world and in parallel in the DT domain);
  - performing long-term simulations to detect various dependencies between xApps;
  - based on the observations of the functioning of various applications, the intelligent creation of the new xApp benefiting from the already installed ones.

7. CONCLUSION

This position paper describes the need for autonomous operation in O-RAN; based on current O-RAN implementations, engineers are focused on creating solutions that are tightly-coupled to esoteric platform implementations, rather than reusable, platform-portable innovative solutions in their expert domains. Influenced by active efforts in various standards bodies, as well as hyperscaler app marketplaces, we propose a straightforward approach to the assisted design and automated

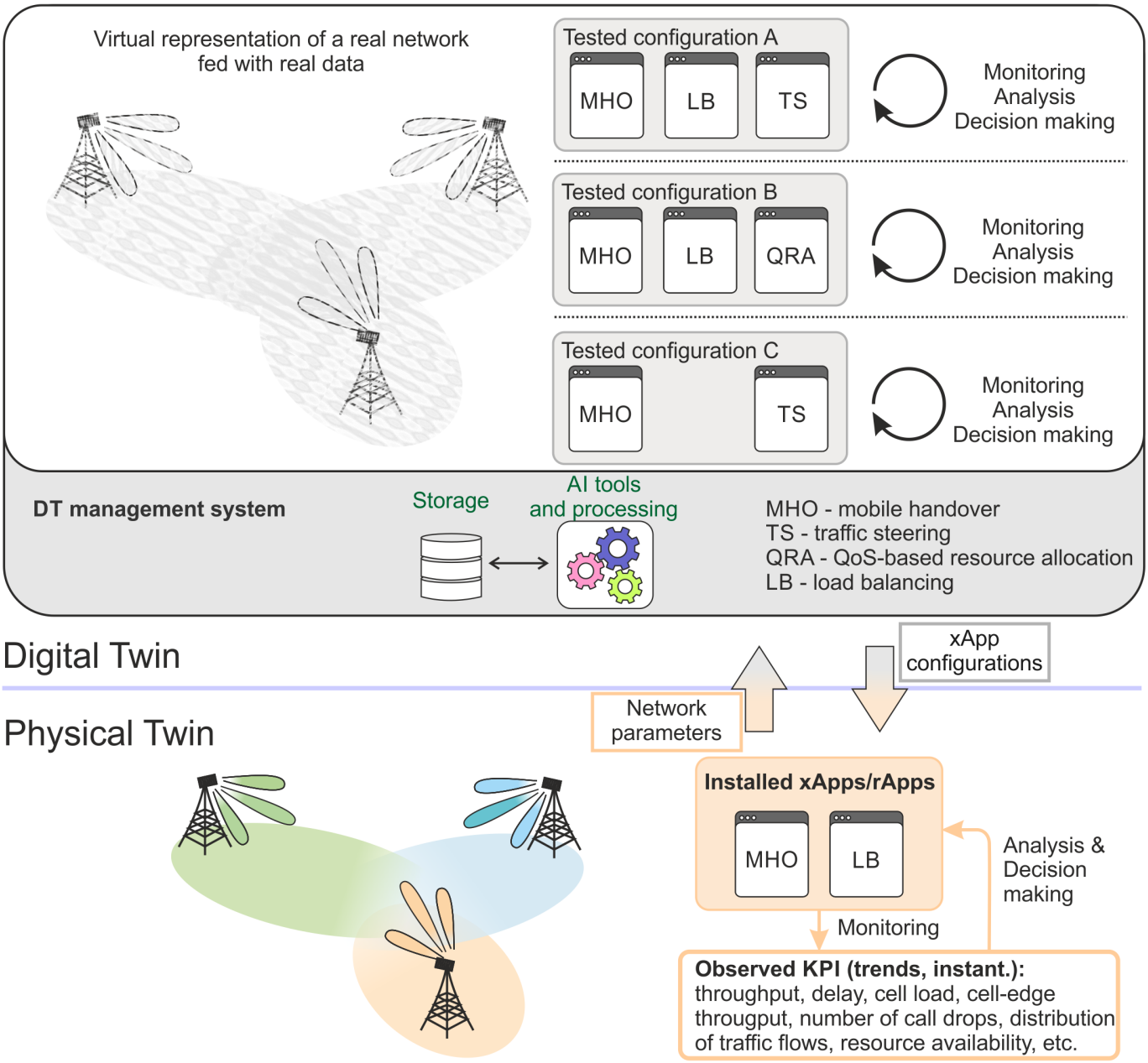

Fig. 5 – Illustration of the usage of digital twin concept for xApp/rApp automated testing

testing in O-RAN. Via a concrete example, we describe our platform (RIC)-independent approach, in which an xApp may be described, verified, tested, and deployed by operators and third parties in an autonomous manner. We believe that our approach will enable engineers to more easily focus on improving RAN operations and providing high-quality service and experience for users.

Looking forward, we plan to develop our platform-independent RIC to demonstrate autonomous development and deployment for various xApp use cases, as well as explore the same approach to automatically configure testing sandboxes and use them to perform experimental validation of xApps. Additionally, we will explore a common marketplace for xApp and testing based on our template system. Together, we see this as a pathway to platform-independent *autonomous devops* for open radio access networks.

APPENDICES

A. THE XAPP/RAPP TEMPLATE

In this appendix, we present the proposed xApp template (1), followed by a discussion of its components. Let us briefly describe the main blocks of the proposed xApp/rApp manifest file. First, in the <metadata> section, all necessary information about the author and the version of the application could be provided. In addition, the version of the O-RAN ALLIANCE standard to which the application is compliant should be exposed.

Next, the second key block in the manifest could define all input and output arguments <IOParameters>. These could be further categorized into subcategories depending on the type of interface, as specified in Fig. 1. In Listing 1, one can see some examples of such parameters depending on the associated interface; for example, one can find the type of base station under the control of certain RICs, available bandwidths and reported radio signal strengths observed as E2 parameters, and policy ID and enrichment information as A1 parameters.

Those are examples, while possibly exact aspects shall be defined, e.g., if the xApp refers to a certain E2 Node type, i.e. O-CU-CP, O-CU-UP, O-DU, or O-eNB, such that the MNO has a specific knowledge, which network elements are affected. Therefore, the list of input and output parameters should be possibly standardized, and it could be used by the conflict detection and mitigation modules to manage the co-functioning of various xApps/rApps.

Once the input and output parameters are specified, xApp/rApp may define the dependencies between other applications from the same provider and other applications offered by other reputable providers.

Next, xApp/rApp may require access to some sensitive data, such as UE location or UE capabilities. It may happen that access to some kind of such data will be mandatory (for the proper functioning of the application) and, to some, only supportive. The interested MNO may filter and select the xApp/rApp also based on the type of information shared with the third-party application. Please note that the relation between the input-output parameters and the permission is not unique.

A separate section in the proposed manifest file deals with the applied AI tool within the xApp/rApp. The application of any AI tool is associated with decisions on applied training models, ways of verification, etc. The MNO may have a choice to select between non-trained AI models (following the description of the provider and rules defined by the RIC provider) and initially trained model (i.e., to choose the model trained by reputable xApp/rApp providers). However, such a decision can be difficult to make, so the application provider should include a description (following the predefined template) of the applied AI tools, models, and the ways in which to use them. Moreover, the application of the selected AI tool entails specific security threats; various advanced and focused attacks against specific AI models are possible nowadays; thus, the MNO shall be informed in detail about the proposed AI tool.

The last section of the proposed manifest file is related to the events to which the xApp/rApp should react and have access. If the considered application deals with improving the mobile handover, it should be informed by the controller about any event of interest that could trigger the start of the application. Similar to the permission section, the event one could be further split into obligatory events and supportive events. The presence of such a section would also support the effective functioning of the RIC. In particular, by exposing the list of events to which the xApp/rApp should be subscribed, the application can be correctly classified by the RIC, proper priorities and hierarchy levels can be assigned to it.

```
\label{lst:manifest}
<metadata>
    AuthorList: Author Name
    AuthorAffil: Author affiliation
    AuthorEmail: email address
    xAppName: ExamplexApp
    xAppVersion: app version
    xApp subversion: app subversion
    Standard_version: version
    …

    <BriefDescription>
        The goal of this xApp...
    </BriefDescription>
</metadata>
<IOParameters>
    <E2In>
        E2InParamList:
            VecBS_PCI: {List of IDs}
            //for each entry in the VecBS_PCI
            PCI_BS_type: {small, macro}
            {required/additional}{standard
    version}
            PCI_freq:{bandwidth}{}{}
            PCI_UE_assigned:{}
            PCI_UE_RSSI: {}
            Other
    </E2In>
    <A1In>
        A1InParamLIst:
        A1PoliciesApplied: {ListOfId}{}{}
        PolicyID: {title, rules}{}{}
        PolicyIDApplicationRange:{}{}{}
        EnrichmentInfo:{ListOfEI}{}{}
    </A1In>

    <E2Out>
        E2OutParamList:
        VecBS_PCI: {List of IDs}
```

```
        //for each entry in the VecBS_PCI
        PCI_BS_type: {small, macro}
        {modified/available}{standard version}
        PCI_freq:{bandwidth} {}{}
        PCI_UE_assigned:{}
        PCI_UE_RSSI: {}}
        Other
    </E2Out>
    <A1Out>
        A1OutParamLIst:
        A1PoliciesApplied: {ListOfId} {}{}
        PolicyID: {title, rules} {}{}
        PolicyIDApplicationRange:{} {}{}
        Other
    </A1Out>
<IOParameters>

<Dependencies>
  <ownApps>
      <appName>
        xAppName:{Name}
        xAppVersion:{ver}
        List of jointly modified parameters:{}
        Required_data_from_xAppName:{}
        Modified_data_in_xAppName:{}
        Action:{list_of_predefined_actions}
        </appName>
        <otherName>
          …
      </otherName>
  </ownApps>
  <otherApps>
      <appName>
        xAppName:{Name}
        xAppVersion:{ver}
        List of jointly modified parameters:{}
        Required_data_from_xAppName:{}
        Modified_data_in_xAppName:{}
        Action:{list_of_predefined_actions}
        </appName>
        <otherName>
      …
      </otherName>
  </otherApps>
</Dependencies>

<Permissions>
  <perm-obligatory>
        Access-to-UE-location
        Processing-of-UE-locations
        Access-to-UE-capabilities
        Modification-of-handover parameters
    </perm-obligatory >
    < perm-preferred>
        List-of-neighboring-cells
        List-of-UE-from-neighboring-cells
        Historical-data
  </perm-preferred >
</ Permissions >

<AITools>
  <type>
        AIType:{supervised/unsupervised/
    reinforced}
    </type>
    <alg>
        algType:{LTSM}
        algDesc:{brief description}
    </alg>
</AITools >

<AITools_tests>
  <input_data>
        In_data:{filesInThexAppPackage}
        In_data_desc:{brief description of the
     input data}
        Initial_AI_setup:{Setup of the AI
    Models}
        Initial_network_setup:{Setup of the
    wireless network model - used for digital
    twin testing}
    </input_data>
    <output_data>
        OutData:{results_of_training}
    </output_data>
</AITools_tests>

<Events>
  <events-obligatory>
        Handover - start
        Cell reassociation - start
        Cell reassociation - finished
  </events-obligatory>
  <events>
        New user detected
        5QI reported/changed
  </events>
</Events>
```

Listing 1 – Generic xApp/rApp manifest structure

B. USE CASE EXAMPLE

Let us now discuss the above observations in the specific use case, the creation and deployment of the Traffic Steering (TS) use case, as specified in [24]. The considered TS xApp has been, in practice, implemented and tested using the open-source environment by Rimedo Labs, i.e., the SD-RAN by Open Networking Foundation. The TS application should be deployed at the Near-RT RIC as its goal is to provide network optimization within the control loop of between 10 ms and 1 s. The following set of mandatory parameters (from the TS xApp design perspective) can be identified as defined in the standard: Distribution of Synchronization Signal Reference Signal Received Power (SS-RSRP),

User Equipment Identity (UE ID), Cell Global Identity (CGI), Single – Network Slice Selection Assistance Information (S-NSSAI), 5G QoS Identifier (5QI), Mean number of RRC Connections, and max number of RRC Connections. Next, besides the mandatory items, optional (supportive) parameters may be used, such as Distribution of DL/UL UE throughput in gNB, Radio Resource Utilization (including DL/UL total available PRB, Mean, and Peak DL/UL PRB used for data traffic), and mobility management-related measurements (including the number of inter-gNB handovers, intra-gNB handovers, intra/inter-frequency handover related measurements). Finally, the following output parameters can be identified, such as User Equipment Identity (UE ID), Primary Cell ID, Target Primary Cell ID, List of PDU sessions for handover, List of Data Radio Bearers (DRB) for handover, List of Secondary cells to be set up (optional).
The TS xApp assumes that it will follow the policies specified by the MNO, as defined in [25]. In our implementation, four policies are considered, named as SHALL, FORBID, PREFER and AVOID. In addition, the vanilla AI tool applied (logistic regression) was used to select the best policy for a given user deployment. Finally, the TS xApp should respond to the following events:

- Registration of the new UE in the network;
- change in network status due to the mobile handover;
- start and end of the mobile handover;
- deregistration of a UE in the network;
- change of the TS policy (addition of new policy, selection of new policy, change of the policy setup);
- change the 5QI, as a result of the changed service;
- switching on/off of base station or its elements (like spectrum bands, antennas/rf-chains, carriers, etc.).

B.1 Template example

Based on the above xApp characterization, the following example of the template manifest file could be specified, as shown in Listing 2. One may notice the presence of the metadata block that specifies the generic description of the Traffic Steering Application (TS xApp). Next, all input and output parameters associated with the E2 and A1 interfaces are discussed. It is followed by the definition of the dependencies with the other xApps delivered by xApp provider, namely Quality-of-Service-based Resource Allocation (QRA) and Load Balancing (LB). As in the case of TS and QRA one may state that they are fully complementary, it is envisaged that TS and LB have an impact on a similar set of parameters. As a proposed policy, it is suggested that TS xApp overwrites all decisions of the LB xApp. For proper functioning, the TS application requires some specific permissions, such as access to the power of the signal from each UE in the network. As mentioned above, the application performs the logistic regression algorithm, being an example of supervised learning. Finally, the set of events to which the application should be subscribed is provided. Note, the description of the xApp is taken from [26] as developed by Rimedo Labs.

```
<metadata>
    AuthorList: Author1, Author 2
    AuthorAffil: xApp Provider
    AuthorEmail: email@address.com
    xAppName: Traffic Steering xApp
    xAppVersion: 1.0
    xApp subversion: 1.0
    Standard_version: 03.2022
    <BriefDescription>
       The goal of the xApp is to
    intelligently and flexibly associate users
    -to-cells based on predefined policies.
    Controls cell preferences and mobile
    handovers related to individual users,
    users within the same QoS flow, or users
    associated with a given network slice, to
    improve the utilization of radio resources
     within the network and meet the QoS
    demands of users.
    </BriefDescription>
</metadata>
<IOParameters>
    <E2In>
        E2InParamList:
            User Equipment Identity (UE ID)
            Cell Global Identity (CGI)
            Single - Network Slice Selection
    Assistance
Information (S-NSSAI)
            5G QoS Identifier (5QI)
            Mean number of RRC Connections
            Max number of RRC Connections
    </E2In>
    <A1In>
        A1InParamLIst:
        AVOID, SHALL, FORBID, PREFER
        PolicyID: {AVOID, rules}{SHALL, rules
    }{FORBID, rules}{PREFER, rules}
        </A1In>

    <E2Out>
        E2OutParamList:
            User Equipment Identity (UE ID)
            Primary Cell ID
            Target Primary Cell ID
            List of PDU sessions for handover
            List of Data Radio Bearers (DRB)
    for handover
    </E2Out>
```

```
<IOParameters>

<Dependencies>
  <ownApps>
      <appName>
        xAppName:QRA
        xAppVersion:1.0
        List of jointly modified parameters:
            None
        Required_data_from_QRA:
            Radio Resource Utilization
            DL/UL throughput distribution
        Modified_data_in_QRA:
            Number of active PDU sessions
            Number of active UEs
        Action: {NONE}
        </appName>
        <appName>
        xAppName:LB
        xAppVersion:2.0
        List of jointly modified parameters:
            List of DRBs for handover
            List of PDU sessions for handover
            UE ID
            Target Primary Cell ID
        Required_data_from_LB:
            None
        Modified_data_in_LB:
            None
        Action: {OVERWRITE TS DECISIONS}
        </appName>
  </ownApps>
</Dependencies>

<Permissions>
  <perm-obligatory>
        Access-to-UE-RSSI reports
    </perm-obligatory >
    < perm-preferred>
        List-of-neighboring-cells
        List-of-UE-from-neighboring-cells
  </perm-preferred >
</ Permissions >

<AITools>
  <type>
        AIType:{supervised}
    </type>
    <alg>
        algType:{Logistic Regression}
        algDesc:{The algorithm decides on the
    best policy for a given distribution of
    UEs}
    </alg>
</AITools >

<AITools_tests>
  <input_data>
        In_data:{TRAINED_MODELS}
        In_data_desc:{The model is trained
    over million of user deployments in the 5
    base station scenario}
    </input_data>
    <output_data>
        OutData:{TRAINED_MODEL}
    </output_data>
</AITools_tests>

<Events>
  <events-obligatory>
        Registration of the new UE in the
    network
        Change in network status due to the
    mobile handover
        Start and end of the mobile handover
        Unregistration of a UE in the network
        Change of the TS policy (addition of
    new policy, selection of new policy,
    change of the policy setup)
        Change of the 5QI parameters of give
    users
        Switching on/off of base station, its
    elements (like active bearers, spectrum
    bands, antennas, etc.)
  </events-obligatory>
</Events>
```

Listing 2 – TS manifest example

B.2 Verification and deployment procedure

Based on the information provided in the manifest file, the store and the MNO shall perform numerous tests to verify the true performance of the application. In addition to various typical tests for performance evaluation, dedicated tests could be performed in the scenario described in the manifest. It was mentioned there that logistic regression models had been trained and verified in the network consisting of five base stations over one million UE deployments. Short and long-term tests should be run to assess the performance of xApp in various network configurations. Next, it is mentioned that the TS xApp has identified dependency with the LB application of the same provided, thus test for the mutual impact of these applications shall be performed at least by the store (and by the MNO, if it already has the LB application installed). Finally, the set of events to which the TS should be registered is provided. In that context, advanced testing scenarios shall be defined and tests should be carried out to check possible conflicts between other applications also being dependent on and influencing the same network parameters.

REFERENCES

[1] "O-RAN ALLIANCE website". In: (July 2022). URL: https://www.o-ran.org.

[2] WG1 O-RAN ALLIANCE. "O-RAN Architecture Description, v.6.0". In: (Mar. 2022). URL: https://orandownloadsweb.azurewebsites.net/specifications.

[3] O-RAN ALLIANCE WG3. "O-RAN Working Group 3, Near-Real-time RAN Intelligent Controller Near-RT RIC Architecture, v.2.1". In: (Mar. 2022). URL: https://orandownloadsweb.azurewebsites.net/specifications.

[4] O-RAN ALLIANCE WG2. "O-RAN Working Group 2, Non-RT RIC Architecture, v.2.0". In: (July 2022). URL: https://orandownloadsweb.azurewebsites.net/specifications.

[5] Salvatore D'Oro, Michele Polese, Leonardo Bonati, Hai Cheng, and Tommaso Melodia. "dApps: Distributed Applications for Real-Time Inference and Control in O-RAN". In: *IEEE Communications Magazine* 60.11 (2022), pp. 52–58. DOI: 10.1109/MCOM.002.2200079.

[6] Luca Baldesi, Francesco Restuccia, and Tommaso Melodia. "ChARM: NextG Spectrum Sharing Through Data-Driven Real-Time O-RAN Dynamic Control". In: *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 2022, pp. 240–249. DOI: 10.1109/INFOCOM48880.2022.9796985.

[7] Marcin Dryjański, Łukasz Kułacz, and Adrian Kliks. "Toward Modular and Flexible Open RAN Implementations in 6G Networks: Traffic Steering Use Case and O-RAN xApps". In: *Sensors* 21.24 (2021). ISSN: 1424-8220. DOI: 10.3390/s21248173. URL: https://www.mdpi.com/1424-8220/21/24/8173.

[8] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. "ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms". In: *IEEE Transactions on Mobile Computing* (2022), pp. 1–14. DOI: 10.1109/TMC.2022.3188013.

[9] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges". In: *IEEE Communications Surveys & Tutorials* (2023), pp. 1–1. DOI: 10.1109/COMST.2023.3239220.

[10] Salvatore D'Oro, Leonardo Bonati, Michele Polese, and Tommaso Melodia. "OrchestRAN: Network Automation through Orchestrated Intelligence in the Open RAN". In: *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 2022, pp. 270–279. DOI: 10.1109/INFOCOM48880.2022.9796744.

[11] Theofanis Karamplias, Sotirios T. Spantideas, Anastasios E. Giannopoulos, Panagiotis Gkonis, Nikolaos Kapsalis, and Panagiotis Trakadas. "Towards Closed-loop Automation in 5G Open RAN: Coupling an Open-Source Simulator with xApps". In: *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. 2022, pp. 232–237. DOI: 10.1109/EuCNC/6GSummit54941.2022.9815658.

[12] DASH Industry Forum. "DASH-IF Position Paper: Server and Network Assisted DASH (SAND)". In: *position paper* (2016). URL: https://dashif.org/docs/SAND-Whitepaper-Dec13-final.pdf.

[13] I. F. Akyildiz, E. Khorov, A. Kiryanov, D. Kovkov, A. Krasilov, M. Liubogoshchev, D. Shmelkin, and S. Tang. "xStream: A New Platform Enabling Communication Between Applications and the 5G Network". In: *2018 IEEE Globecom Workshops (GC Wkshps)*. 2018, pp. 1–6. DOI: 10.1109/GLOCOMW.2018.8644183.

[14] I.F. Akyildiz, A. Kak, E. Khorov, A. Krasilov, and A. Kureev. "ARBAT: A flexible network architecture for QoE-aware communications in 5G systems". In: *Computer Networks* 147 (2018), pp. 262–279. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.10.016. URL: https://www.sciencedirect.com/science/article/pii/S1389128618311228.

[15] Shriganesh Yadav and Sameer Nanivadekar. "Hybrid Optimization Assisted Green Power Allocation Model for QoS-Driven Energy-Efficiency in 5G Networks". In: *Cybernetics and Systems* 0.0 (2023), pp. 1–16. DOI: 10.1080/01969722.2023.2175147. eprint: https://doi.org/10.1080/01969722.2023.2175147. URL: https://doi.org/10.1080/01969722.2023.2175147.

[16] Ahan Kak. "Towards 6G Through SDN and NFV-Based Solutions for Terrestrial and Non-Terrestrial Networks". PhD thesis. 266 4th Street NW, Atlanta, GA 30332, 2023. DOI: http://hdl.handle.net/1853/64746.

[17] Ian F. Akyildiz, Ahan Kak, and Shuai Nie. "6G and Beyond: The Future of Wireless Communications Systems". In: *IEEE Access* 8 (2020), pp. 133995–134030. DOI: 10.1109/ACCESS.2020.3010896.

[18] ITU-T Focus Group on Autonomous Networks. *Architecture Networks framework for Autonomous*. ITU-T, Sept. 2022, pp. 1–55. URL: https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Architecture-AN.pdf.

[19] ITU-T Focus Group on Autonomous Networks. *Use cases for Autonomous Networks*. United Nation's ITU-T, Sept. 2022, pp. 1–56. URL: https://www.itu.int/rec/T-REC-Y.Sup71-202207-P/en.

[20] Fei Tao, He Zhang, Ang Liu, and A. Y. C. Nee. "Digital Twin in Industry: State-of-the-Art". In: *IEEE Transactions on Industrial Informatics* 15.4 (2019), pp. 2405–2415. DOI: 10.1109/TII.2018.2873186.

[21] Qinglin Qi and Fei Tao. "Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison". In: *IEEE Access* 6 (2018), pp. 3585–3593. DOI: 10.1109/ACCESS.2018.2793265.

[22] Adil Rasheed, Omer San, and Trond Kvamsdal. "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective". In: *IEEE Access* 8 (2020), pp. 21980–22012. DOI: 10.1109/ACCESS.2020.2970143.

[23] ITU-T. *Recommendation ITU-T Y.3176: Machine learning marketplace integration in future networks including IMT-2020*. Sept. 2020, pp. 1–32. URL: https://www.itu.int/rec/T-REC-Y.3176-202009-I/en.

[24] O-RAN ALLIANCE WG1. "O-RAN Working Group 1, Use Cases Detailed Specification, v.7.0". In: (Mar. 2022). URL: https://orandownloadsweb.azurewebsites.net/specifications.

[25] O-RAN ALLIANCE WG2. "O-RAN Working Group 2, A1 interface: Type Definitions, v.2.0". In: (Mar. 2022). URL: https://orandownloadsweb.azurewebsites.net/specifications.

[26] Rimedo Labs. "O-RAN Traffic Steering xApp - Technical Specification". In: (Mar. 2022). URL: https://mailchi.mp/0eec92853d8f/o-ran-ts-xapp-techspec.

AUTHORS

Adrian KLIKS received his postdoctoral degree in technical sciences, discipline: technical computer Science and telecommunications in February 2019. He works as a university orofessor at the Institute of Radiocommunications of the Poznan University of Technology. He took part in numerous international research projects such as URANUS, NEWCOM++, ACROPOLIS, COGEU, NEWCOM#, COHERENT. He also participated in Cost Actions IC0902, COST-Terra (IC 0905), and is currently active in CA20120 INTERACT action. Adrian has been an IEEE Senior Member since 2013; between 2012-2017 he participated in the work of the IEEE P1900.x standardization group. In the years 2014-2016, he was the Membership Development / Web Visibility Chair in the IEEE for the EMEA area. Since 2019 he is the editor-in-chief of the Journal of Telecommunications and Information Technology of the Institute of Communications. He is a co-founder of the PUT spin-off company Rimedo Labs.

Marcin DRYJANSKI received his Ph.D. (with distinction) from the Poznan University of Technology in September 2019. Over the past 15 years, Marcin has served as an R&D engineer and consultant, technical trainer, technical leader, advisor, and board member. Marcin has been involved in 5G design since 2012 when he was a work-package leader in the FP7 5GNOW project. Since 2018, he is a Senior IEEE Member. He is a co-author of many articles on 5G and LTE-Advanced Pro and a co-author of the book "From LTE to LTE-Advanced Pro and 5G" (M. Rahnema, M. Dryjanski, Artech House 2017). Currently, he serves as CEO and principal consultant at RIMEDO Labs.

Vishnu RAM holds a masters degree in computer science and engineering from Indian Institute of Technology, Delhi. He has hands-on experience in the field of the telecommunications industry for more than two decades, developing and implementing standards, and holds 15 international granted patents. He was nominated to Scientific Advisory Board Associate (SABA) member of Motorola Networks and was a senior specialist in the Advanced Technologies group of Nokia Networks. Currently, he works as an independent expert, coordinating standards initiatives, liaisons with other SDOs, industry bodies, open source and academia, mentoring student projects and coordinating the ITU "AI/ML in 5G" Challenge around the globe. Vishnu is a senior member of IEEE and a vice chair of the ITU-T Focus Group on Autonomous Networks and co-convener of ITU-T Correspondence Group on datasets.

Leon WONG Leon Wong is the research collaboration and engineering lead for Research & Innovation Lab in Rakuten Mobile. He is also currently serving as chairman of ITU-T Focus Group of Autonomous Networks (FG-AN), established under ITU-T Study Group 13 -

Future networks and emerging network technologies.

.

Paul Harvey gained his PhD from the University of Glasgow in 2015 in the area of *distributed adaptive systems*. Since then he has worked on runtimes for high performance computing as a work-package leader in the ECOSCALE H2020 project and trustworthy adaptive systems as a JSPS fellow. From 2018, Paul joined Rakuten in Japan and became co-founder and research lead of the Rakuten Mobile Innovation Studio. Paul is now a lecturer in autonomous systems at the University of Glasgow, visiting scholar, Focus Group on Autonomous Networks working group co-chair, and a board member. (paul-harvey.org)