Intent-driven Closed Loops for Autonomous Networks

Pedro Henrique Gomes*, Magnus Buhrgard, János Harmatos, Swarup Kumar Mohalik, Dinand Roeland and Jörg Niemöller

Ericsson Research, Brazil
E-mail: pedro.henrique.gomes@ericsson.com
**Corresponding Author*

Received 17 November 2020; Accepted 23 March 2021;
Publication 03 June 2021

Abstract

Closed loops are key enablers for automation that have been successfully used in many industries for long, and more recently for computing and networking applications. The Zero-touch network and service management (ZSM) framework introduced standardized components that allow the creation, execution, and governance of multiple closed loops, enabling zero-touch management of end-to-end services across different management domains. However, the coordinated and optimal instantiation and operation of multiple closed loops is an open question that is left for implementation by the ZSM specifications. In this paper, we propose a methodology that uses intents as a way of communicating requirements to be considered by autonomous management domains to coordinate hierarchies of closed loops. The intent-driven methodology facilitates hierarchical and peer interactions for delegation and escalation of intents. Furthermore, it extends the existing management capabilities of the ZSM framework and facilitates conflict-free integration of closed loops by setting optimal (and non-conflicting) goals that each closed loop in the hierarchy needs to account for. We show an example of the application of the proposed methodology in a network slicing assurance use case. The new capabilities introduced in this paper can be considered as an extension of the

Journal of ICT Standardization, Vol. 9_2, 257–290. *River Publishers*
doi: 10.13052/jicts2245-800X.929

ZSM framework to be used in scenarios where multiple intent-driven closed loops exist.

Keywords: Closed-loop automation, intent-driven management, zero-touch management.

List of Acronyms

3GPP	Third Generation Partnership
5G	Fifth Generation
AI	Artificial Intelligence
CL	Closed Loop
CLA	Closed-Loop Automation
CN	Core Network
E2E	End-to-End
ETSI	European Telecommunications Standards Institute
GSMA	GSM Association
IBN	Intent-Based Networking
IRI	Internationalized Resource Identifier
MD	Management Domain
ML	Machine Learning
MnS	Management Service
NEST	NEtwork Slice Type
OWL	Web Ontology Language
RAN	Radio Access Network
RDF	Resource Description Framework
RDFS	RDF Schema
SDN	Software-Defined Network
SDO	Standards Developing Organization
SLA	Service Level Agreement
SLO	Service Level Objetive
SLS	Service Level Specification
URLLC	Ultra-Realible Low Latency Communication
ZSM	Zero-touch network and Service Management

1 Introduction

The advances of virtualization and network softwarization create a flexible and dynamic infrastructure that will be leveraged by 5G networks to enable new and more flexible service offerings. Resource allocation is expected to change more frequently and the complexity of management tends to grow exponentially with the larger set of services. The new customer-facing services, such as online gaming, AR/VR, intelligent transport systems, smart cities, etc., create the need for optimizing network providers' operational efficiency to reduce costs while meeting the new requirements.

This high efficiency is only possible with the extensive adoption of network automation capabilities that minimize the need for human intervention. This is the scenario where zero-touch operations take full advantage of automation to create management processes that continuously optimize resource allocation to keep the whole network at the optimal state while meeting all (dynamic) functional and non-functional requirements.

One of the major differences in the provision of new (5G) services is the fact that the operating requirements are more and more dynamic. This means that the way requirements are translated into decisions and actions to be taken in the network need to change; new forms for the representation and communication of users' expectations that facilitate such translation is essential for the success of service providers. Instead of policies and other forms of imperative communication, high-level abstractions should be used to allow the translation of requirements across layers of management as well as across multiple domains that compose the end-to-end (E2E) service.

Intent-driven management has been proposed [4, 11] as a way of coping with the complexity of 5G services management. Intents are used to express all expectations to be fulfilled by an autonomous system. It can be used to convey the communication between management consumers and producers and to express the requirements (or parts of them) that need to be considered by a management (sub-)system.

Closed-loop automation (CLA) aims at removing human intervention and creating autonomous systems that can self-monitor, self-evaluate and self-heal and fulfill all specified requirements. It is, thus, a key tool to dynamically meet all requirements of 5G networks. However, the increased complexity in network operation and optimization (e.g., support slicing, virtualization, cloud-native operation, multi-domain, multi-layer ecosystem) makes traditional policy-based CLA solutions insufficient due to lack of flexibility. Similar to the concept of intent-driven management, where communication

between management services is conveyed by intents, the intents may become the basic tool to specify the goals for CLA and describe the interactions between the closed loops (CLs). This means that the different CLs that are employed in the management system can use intent objects as the (only) way of communicating the goals to be considered in their operation, making them intent-driven CLs. Therefore, intent-driven CLs can be used at different places for network management (at vertical layers and different domains) to create an E2E automation framework that is flexible to deal with dynamic requirements and provide operational efficiency with highly autonomous network management.

The ETSI ZSM [30] working group has proposed a future-proof reference architecture for E2E network and service automation that provides important tooling for the realization of intent-driven CLs. The ZSM framework is logically composed of management and data services that are distributed across multiple management domains (MDs) connected via integration fabrics, that enable management services consumption, communication, and integration with other management systems. In the ZSM framework, different CLs are running within the E2E service layer, as well as crossing multiple MDs, or within each of the involved MDs. The specification ZSM009-1 [18] has provided functionalities for the governance of CL instances within the ZSM framework, but aspects of communication between multiple CLs are not addressed.

The main contributions of this position paper are the following:

1. We propose a methodology to apply intent-driven management for the coordination of hierarchies of CLs. In the proposal, intent objects and intent handling functions are used for specification, communication, and processing of service requirements at different levels of the management system;
2. The proposal is applied to the ZSM framework to increase the autonomy levels of management domains;
3. New management capabilities are proposed to allow intent handling operations such as delegation, reporting and escalation, to implement intent-driven closed loops within the ZSM framework;
4. An example is described to illustrate the application of the proposed methodology on a service assurance use case, based on network service requirements specified by GSMA network slice templates.

This paper is organized as follows. Section 2 describes how CLA is realized within the ZSM framework. Section 3 describes how intent-driven

systems work and how this new approach is used in modern management systems. Section 4 defines intents and shows how intent management functions can be realized. Section 5 describes intent-driven CLs and how they can be coordinated. Section 6 summarizes the main related works with an emphasis on standardization and open source projects related to intent-driven management. Finally, Section 7 concludes this paper.

2 Closed-Loop Automation within the ZSM Framework

A CL is a control mechanism that uses feedback signals, i.e. outputs that are routed back as inputs and create a chain of cause-and-effect that forms a loop. CLs monitor and regulate themselves to achieve a specific goal, e.g. to minimize energy consumption, or to maximize service throughput. In management systems, CLA can be realized with the combination of management functions that provide data, analytics functionalities, policies, orchestration tools, etc. to create an autonomous system that constantly monitors and assesses the network and takes corrective actions to meet the specified goals [18]. CLA is one of the key principles of ZSM architecture framework. Within the ZSM framework, the CLA is realized by chaining the management services that are needed to autonomously collect data, make decisions and execute actions upon the managed entities.

A detailed description of how ZSM framework supports CL operations can be found in Annex C of ZSM002 [15] specification and is shown in Figure 1.

2.1 ZSM009-1 – Closed-loop Automation Work Item

Closed-loop automation is the central topic for the ZSM009 work item [3]. In general, the ZSM002 specification offers basic building blocks, e.g. management services, integration fabrics, data services, that are necessary to create the stages of a CL, while in ZSM009-1 specific functionalities that are needed to govern the execution of CLs, including some basic enablers for their coordination, are further detailed. ZSM009-1 splits the enablers into two groups of functionalities: governance and coordination.

The governance capabilities allow entities to manage the behaviour and life cycle of CLs. As part of the CL governance, we have capabilities for lifecycle management, i.e. design, instantiation, the configuration of CLs, together with capabilities for setting policies, priorities between different CLs, etc. Any authorized entity can retrieve information related to all CLs

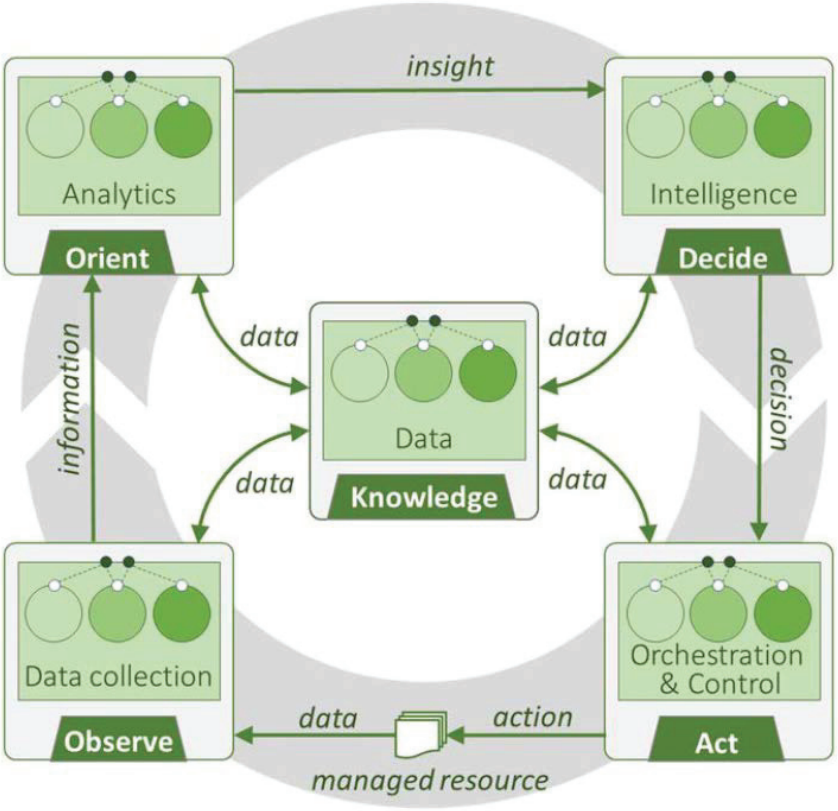

Figure 1 Functional view of CLs within the ZSM framework (source: [15]).

within the ZSM framework employing CL governance, which may include status as well as performance (e.g. health) information. CL governance allows interactions with the CL both at design-time and run-time. The coordination capabilities include mostly run-time functionalities that can be used to integrate and interoperate multiple CLs that need to work together to achieve global (E2E) goals.

At the time this paper was written there are not many details specified related to CL coordination capabilities. Therefore, this paper tries to build upon the current specifications of ZSM002 and ZSM009-1 and proposes a methodology for coordination of CLs based on the use of intents as a means for conveying dynamic requirements of new (5G) services and enabling higher autonomy of MDs.

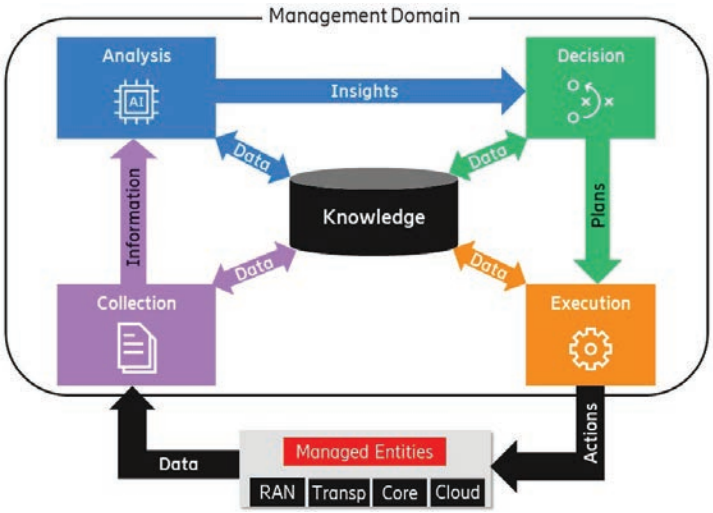

Figure 2 Functional view of CLs within the ZSM framework.

Based on the ZSM principles and the Management Services (MnS) defined in ZSM002, any type of CL can be realized, e.g. OODA [9], MAPE-K [13], COMPA [19], etc. Figure 2 shows an exemplary functional view of a CL considering 4 stages plus knowledge, following the functional view from ZSM009-1.

The primary data and control flow of the CL model shown in Figure 2 starts with the *collection* stage, which is responsible for collecting and pre-processing data coming from the managed entity. The data collected by *collection* stage is aggregated and sent to *analysis* stage, which is responsible for deriving the right insights about the current status of the environment under control. The insights consider the current and historical data, combined with the knowledge, and they try to assess if the goals of the CL are met. If the goals are not met, then some diagnosis on the reasons why this is happening should also be done. The insights from *analysis* stage are fed to the *decision* stage, which is responsible for finding the solution that makes the CL meet its goals and create plans for the necessary actions that need to be taken. The generated plans are technology-agnostic, usually in the form of workflows. Therefore, they are sent to *execution* stage to be translated into actions, according to the context and the technology employed in the managed entity. In hierarchical CLs, the actions are not only sent towards resources, but also to other CLs that are at different management layers. The

outcomes from *execution* are observed after new data are collected and the gap between the current and expected states is evaluated.

The *knowledge* in the CL has the main purpose of storing and retrieving data (including context and experiences) that can be shared between the different stages, as well as between different CLs. Knowledge can also be used as a means for feedback signalling between the CL stages.

2.2 Closed Loops Within the ZSM Framework

For most use cases that involve E2E services, more than one CL will be necessary. And the ZSM framework has a principle for the division of concerns that allows the integration between E2E management and management within multiple MDs, e.g. radio access network (RAN), transport, and core network (CN).

Figure 3 shows the integration between 6 different CLs in a ZSM deployment that can be applied for use cases such as E2E network slice assurance.

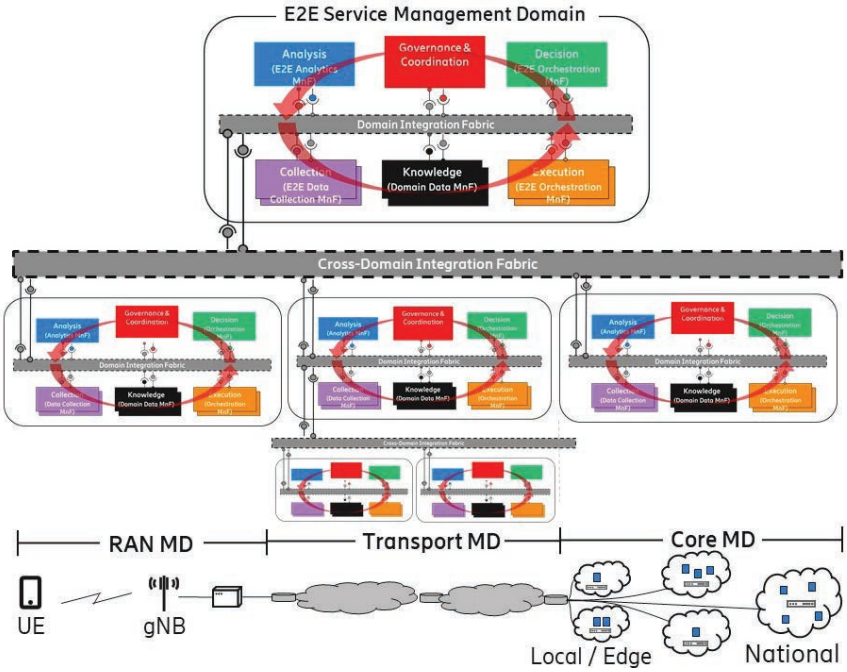

Figure 3 Deployment view of hierarchical CLs within the ZSM framework.

The CL on the top level (at the E2E service MD) has a broad view and is responsible for translating and decomposing the business inputs into domain-specific ones. The 5 CLs on the bottom level (at the MDs) have narrow views and are responsible for assuring parts of the E2E service. We show in Figure 3 that multiple levels of CLs can also exist within the MDs. In the example, transport has a hierarchy of 3 CLs forming 2 layers of management.

In the ZSM framework, the internal details of each MD are optionally exposed towards the E2E level. Therefore, the E2E CL may or may not be aware of all the existing CLs within the MDs. Each level in the hierarchy of CLs has to abstract its details and provide the necessary information to the CLs at the upper and lower layers. Optionally, CLs that are at the same layer can also engage in interactions to improve their performance.

Since each CL is responsible for parts of the network resources (horizontally), or has different views on the network service (vertically), they need to interact to work towards a (set of) common goal(s). Without proper interactions between the CLs, the gains achieved by individual CLs may be compromised if conflicting and/or contradicting actions are taken.

With a hierarchy of CLs, as depicted in Figure 3, different types of interactions can take place to enable autonomous management of E2E services. CLs located at different levels, such as the E2E CL and the MD CLs, or even the top MD CL and the two other bottom MD CLs at the transport domain, can engage in hierarchical interactions. Examples of such interactions are:

- delegation of domain-specific Service Level Specifciations (SLS) from the E2E CL to the lower CLs;
- escalation of issues such as SLS violations from the lower CLs to the E2E CL;
- performance and status measurements and reporting in both directions (downwards and upwards) to keep track of the execution of all CLs in the hierarchy;
- configuration of operational policies from the E2E CL to the lower CLs to steer its behaviour;
- request on fine or coarse-grained data and knowledge in both directions, to improve decision making done by individual CLs.

Finally, CLs located at the same level within the same MD, or at different MDs, such as the 3 CLs at the RAN, transport, and CN, can engage in peer interactions. Example of such interactions are:

- performance and status measurements to keep track of the execution of the peer CLs;

- joint optimization considering the knowledge and environment observability of different CLs;
- conflict detection and resolution of actions towards the same (set of) resource(s).

3 Intent-driven Systems and Management

Intents have been used in IT and networking mainly in the context of software-defined networks (SDN). The main objective was to create a high-level abstraction layer to facilitate the expression of commands and configurations and allow the selection of policies within the SDN controllers. The SDN north-bound interface was proposed to allow network administrators to define a desired outcome or business objective and leave the implementation up to the SDN controller. An intermediate context-aware translation layer is usually applied to convert the intents, which are in many cases based on natural language, into interpretable actions and/or configurations [25].

3.1 Intent-based Networking

Even though the concept of intent-based networking (IBN) is very generic, in practice it has been used exclusively for interactions between network administrators and network controllers. A similar application of intents has also happened in digital assistant devices, e.g. Amazon's Alexa, Apple's Siri, to which programmers can use intents to allow voice-enabled interactions between users and applications. In these systems, similarly to IBN, natural language input, e.g. user asking to launch an application, is interpreted and converted into executable actions, e.g. application startup or specific API invocation.

Besides IBN and software engineering, the intent is also considered for autonomous networking, and its definition has evolved from being declarative statements that are used to select appropriate policies into more generic artifacts that convey desired outcomes from business perspectives. RFC7575 [8], from 2015, defines intent as "an abstract, high-level policy used to operate the network", while more recent work from the same group has defined an intent as "a set of operational goals that a network should meet and outcomes that a network is supposed to deliver, defined in a declarative manner without specifying how to achieve or implement them." [11]. This shows how in the industry the definition of intent has shifted from a concept related to policies and rules towards a universal entity to be used for declaration and

communication of goals and requirements. Similarly, in ZSM intents are "an abstracted way to specify business or operational desired state of a system, without specifying how to achieve it." [18] In this sense, intents can as well be used to "specify a goal for a CL to accomplish", while "policies specify a behavioural pattern that a CL should follow" [18].

3.2 Properties of Intents

Intents hold important properties that make them useful for zero-touch automation [21]:

- Intents are **comprehensible** – They need to be understandable by humans, while still conveying formally and unambiguously all the necessary information to be used by an autonomous system;
- Intents are **declarative** – They express the expected behavior and do not specify how the result is supposed to be achieved;
- Intents are **infrastructure agnostic** – They only change according to the (dynamic) requirements, but they are decoupled from variations in the underlying infrastructure;
- Intents are **portable** – They are also independent of vendors and protocols;
- Intents are **composable** – Multiple intents are usually considered for any service, and they need to work together;
- Intents are **persistent** – The intents need to be fulfilled until they are removed from the system;
- Intents are **measurable** – The intents should use measurable and ideally use standardized metrics to define the desired state.

3.3 Intent-driven Management

Intent-driven management has been considered [6, 20, 17] as a way of taming the complexity of management systems for new services. In the new intent-driven management paradigm, an intent-driven MnS is a management service that allows its consumer to express an intent [6]. The intent-driven MnS producer translates the intent into executable actions, which may include performing network management tasks, identifying, formulating, and activating policies [6], or invocation of other intent-driven MnS.

Intents can be used as the abstraction between autonomous entities that need to work towards a common goal. Besides translating the intents into executable actions, the intent receiver (e.g. intent-driven MnS producer)

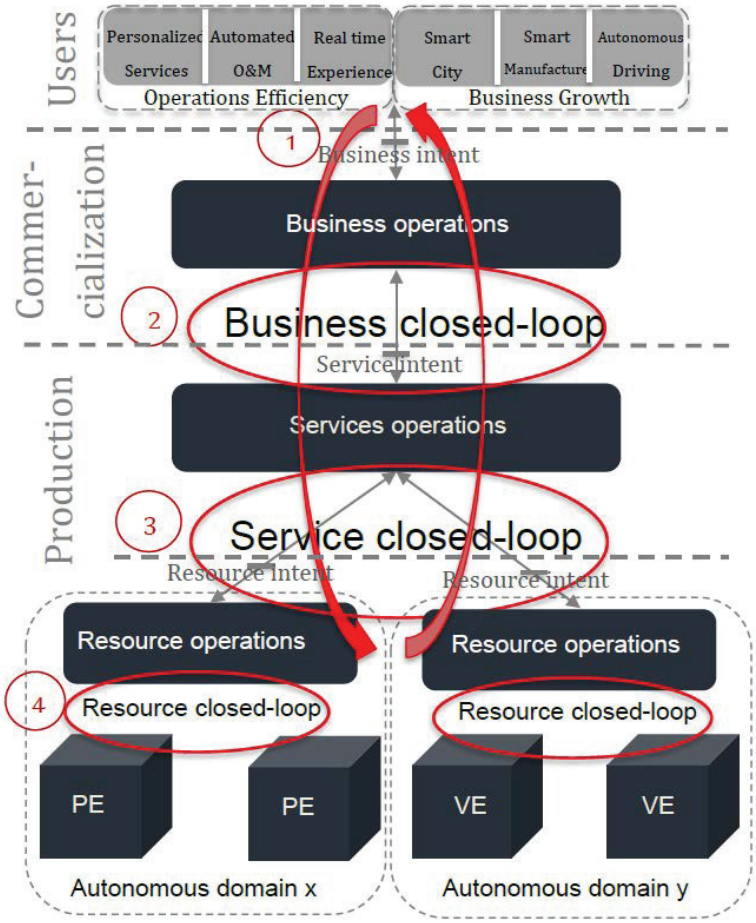

Figure 4 TM Forum's autonomous networks architecture (source: [20]).

needs to evaluate the intent fulfilment and report the results to the intent sender (e.g. intent-driven MnS consumer).

If autonomous entities that are dealing with the management of parts of the E2E service employ CLs, intents can also be used to express the expectations of each CL in the management hierarchy. TM Forum proposes an architecture for autonomous management with 3 layers and 4 CLs [20], as shown in Figure 4. For each level, one type of CL exists and the different CLs interact based on intents. Therefore, the interactions between adjacent layers are technology/implementation independent, based in intent-driven CLs.

As explained before, intents are declarative, therefore, their translation will change in each system and will depend on the maturity level of the management system and algorithms employed in the translation. This non-deterministic behavior of autonomous systems is a necessary feature to allow them to deal with dynamic requirements that are part of new network services, but it may incur concerns related to the trust of the operator in their autonomy. Even though policies and rules can be used for the interpretation of intents and their translation into actions and/or other (sub)intents, the use of AI/ML technologies for this purpose should also be considered [18]. If AI/ML is applied, a supervision layer can be used to allow the verification of requirements and possibly incorporate inputs from domain experts while the autonomous translation of intents is improved and the zero-touch system gains operator confidence. New techniques that provide explainable AI can be leveraged to create trustworthy intent translation processes.

4 Intent Specification and Intent Handling Function

Given the evolution of the use of intents in the industry and how this concept is currently applied in standardization, as summarized in Section 3, we consider more generically that:

"an intent is a formal specification of expectations that need to be enforced in a system"

Different types of expectations can be included, but they generally express the (functional and non-functional) requirements, constraints, and goals of a system. Intents are usually applied at the business level to express the expectations coming from users, but more generally they are also applicable to different levels of the management system, aligned with the concept of intent-driven management.

4.1 Intent Objects

In a management system, an intent should be an information object that formally specifies all expectations to be fulfilled. The intents are the information that tells an autonomous system what it is expected to do. This means that for a truly zero-touch operation the management system needs to have capabilities to translate intents into decisions and actions. Intents should be the sole form of communicating the requirements between the zero-touch system and human operators, as well as between the different sub-systems

and layers of the management system. In the ZSM framework, this means that the service specification that is provided by the ZSM framework consumers should be conveyed by an intent object and the E2E service domain should be responsible for translating it into (sub)intents that specify domain-specific requirements that need to be fulfilled by each MD.

The communication based on intent objects is a generic mechanism that can be applied to any MD within the ZSM framework. With intents, the domain-specific semantics can be shifted into shared information models and the end-points that are based on intents can use a generic knowledge management service for life-cycle management of intent objects.

4.2 Types of intents

Intents can express expectations that are service-specific or general to all services. Service-specific intents are related to the fulfillment and assurance of a service instance that has been agreed upon between two entities within the management system. The agreement at the business level can be specified by service level agreements (SLAs), while at the service level, service level specifications (SLSs) together with service level objectives (SLOs) can be used. At the resource level, many types of operations may need to be included, and for each of them, the intents can be expressed by models and information objects. Non-service-specific intents are globally applicable to all service instances. They may involve aspects related to regulatory and/or legal compliance, security levels, and standardization conformance. General intents can also be used to express higher-level non-technical goals that allow the specification of common sense that usually is not necessary for human-driven operations, but is necessary for the completeness of intent-driven management. One technician knows, for instance, that resources need to be saved whenever possible, even though this non-functional requirement is not specified anywhere. Such common-sense type of intent can be applied for other purposes, such as financial gains, avoid unwanted bias in decision-making, allowed risk-taking levels, among others.

4.3 Intents Meta-models

Intents, as the formal specification of expectations, need to be properly modeled to be processed by the MDs. Intent has to be interpreted by the receiver without any ambiguity. This can be achieved by using a machine-readable declarative language, such as Resource Description Framework (RDF) [1].

The RDF is a standard specification for metadata modelling that was originally designed for data interchange on the web. It has features that facilitate merging data with different underlying syntax notations, supporting the evolution of data schemas over time. It is frequently used in knowledge management and is supported by many knowledge management and machine reasoning tools. RDF structures the data as a set of triples, consisting of a subject, predicate, and object. The subjects and objects can represent resources or literals. Resources can be anything, including a document, physical element, or abstract concept, and it is denoted by an internationalized resource identifier (IRI). RDF Schema (RDFS) is the RDF schema vocabulary that can be used for knowledge representation. RDFS is a set of classes with certain properties using the RDF data model, providing basic elements for the description of ontologies. Finally, Web Ontology Language (OWL) adds vocabulary to RDFS as well as semantics and allows execution of automatic reasoners for the knowledge process. An introduction to RDF, RDFS, and OWL can be found at [1].

The intent objects are knowledge objects that need to be stored and retrieved from knowledge bases. These knowledge bases should be machine-readable and easily used by machine reasoning algorithms. RDF ontologies can be represented by languages such as UML and OWL, can be processed by SPARQL query language, and serialized using JSON, XML, or Turtle. Among the different serialization languages, Terse RDF Triple Language (Turtle) has a more human-readable format and is commonly used in knowledge and ontology modelling. An introduction to Turtle can be found at [2].

4.4 Formal intent definition

We provide here a simplified ontology to represent services and intents, which are used later to exemplify the use of intent-driven closed loops for network slice assurance use case.

All intents should inherit from Intent class, which is an OWL class, as defined below.

```
:Intent rdf:type owl:Class ;
    rdfs:comment "This is the root class of all intent
    objects" .
```

Intents have a set of expectations, consisting of targets and parameters, as defined below.

```
:Expectation rdf:type owl:Class ;
    rdfs:comment "An expectation that is part of an
    intent" .

:hasExpectation rdf:type owl:ObjectProperty ;
    rdfs:domain :Intent ;
    rdfs:range  :Expectation .
```

Targets may be used to refer to different entities within the management system. They can refer to the managed entities that are part of a specific service type, a specific service instance, within a given MD, or any subset of managed entities to which the expectation should be applied to. Parameters specify the list of individual properties to assign goals and set targets. Targets and parameters are defined as below.

```
:target rdf:type owl:ObjectProperty ;
    rdfs:domain :Expectation ;
    rdfs:range  :ManagedEntities .

:parameters rdf:type owl:ObjectProperty ;
    rdfs:domain :Expectation .
```

Different types of expectations can be derived from class *Expectation* to assign different goals to intents and set different targets. Examples of expectations that can be created are to minimize or maximize a given metric, or more specific expectations for availability, slice QoS, UE density, etc.

4.5 Example of an Intent Object

We show below an example of a service level intent specification. The example specifies all the expectations for an ultra-reliable low latency communication (URLLC) service. The specification follows the recommended attributes and their values, according to NEST standard (Section 4.2 of NG.116-v3.0 [23]).

```
:urllcServiceIntent rdf:type owl:NamedIndividual ,
    :Intent ;
    rdfs:comment "Intent to assure an URLLC service" ;
    :hasExpectation :urllcAvailabilityExpectation ,
                    :urllcDeviceVelocityExpectation ,
                    :urllcMaxThroughputExpectation ,
```

```
                    :urllcSliceQoSExpectation ,
                    :urllcUeDensityExpectation .

:urllcAvailabilityExpectation rdf:type
owl:NamedIndividual ,
        :AvailabilityExpectation ;
    :target :urllcService ;
    :parameters [ :availability "99.999"^^xsd:float ] .

:urllcDeviceVelocityExpectation rdf:type
owl:NamedIndividual ,
        :DeviceVelocityExpectation ;
    :target :urllcService ;
    :parameters [ :deviceVelocity 2 ] .

:urllcMaxThroughputExpectation rdf:type
owl:NamedIndividual ,
        :MaxThroughputExpectation ;
    :target :urllcService ;
    :parameters [ :downlinkThrPerUE 100000 ;
                  :uplinkThrPerUE 100000 ] .

:urllcSliceQoSExpectation rdf:type owl:NamedIndividual ,
        :SliceQoSExpectation ;
    :target :urllcService ;
    :parameters [ :sliceQoSParameters 82 ] .

:urllcUeDensityExpectation rdf:type owl:NamedIndividual ,
        :UeDensityExpectation ;
    :target :urllcService ;
    :parameters [ :ueDensity 1000 ] .
```

The intent based on GSMA NEST is only one example to be considered in an intent-driven management system. The same service expectations can be specified at a higher level, considering business goals, or even at lower level, considering specific technology domain aspects, e.g. transport latency and guaranteed bandwidth. We decided to adopt GSMA NEST as an information model for the service-level specification of intents as this has already been discussed in standards like 3GPP and ZSM [16] as a way

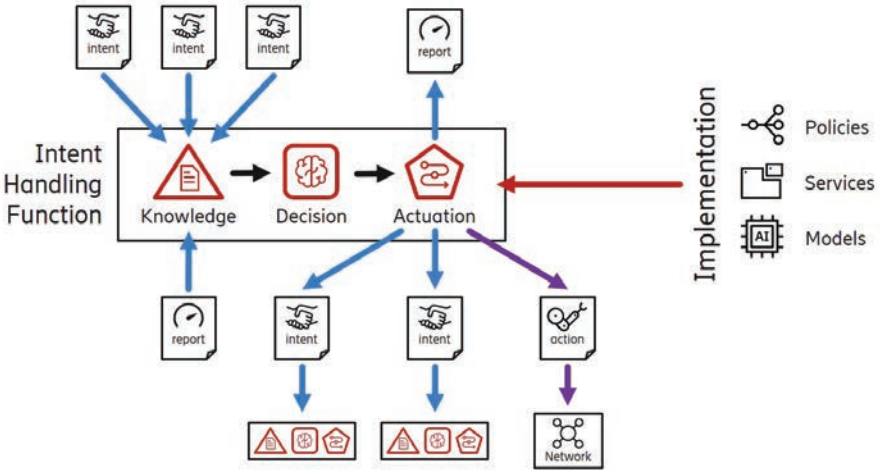

Figure 5 The intent handling function (source: [20, 27]).

of supporting the requirements for network slicing from different vertical consumers.

4.6 Intent Handling Function

Intents are the objects that represent the goals at different levels in an E2E autonomous system, e.g. business, service, or resource levels. They need to be handled accordingly in a distributed fashion throughout the different layers of management as well as across different MDs. The intents processing is done by intent handling functions that operate at each level and within each domain. Intent handling functions are the architectural building block to assemble intent-driven operations (Figure 5). They analyse the discrepancy between the observed network state and the state expressed by the intents [21].

The ultimate goal of the intent handling function is to close the gap between desired and actual states, which may involve decision-making, optimization processes, and even conflict resolution between multiple intents. Intent handling functions are also responsible for managing the life cycle of all intents.

A hierarchy of intent handling functions is expected in scenarios with multiple (autonomous) layers of management. In the ZSM framework, an E2E service intent is handled by the intent handling function located at the E2E service MD, which will send service intents to MDs that match the

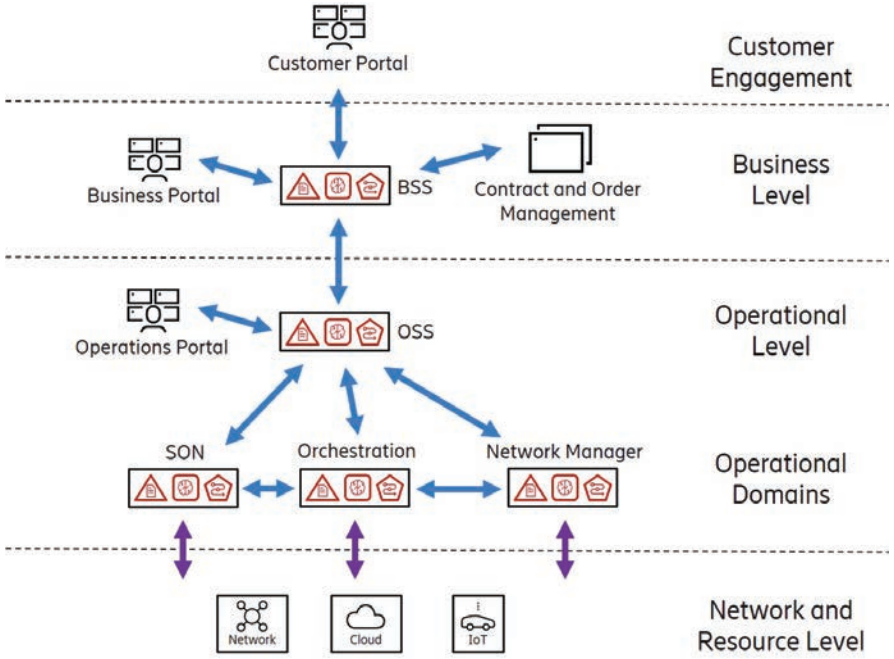

Figure 6 Network operation through distributed intent handling (source: [20, 27]).

concerns specified in the E2E service intent. Intents can originate directly from a user, or any ZSM framework consumer, and then additional intents are derived automatically and propagated downwards to specific MDs. Figure 6 shows the same concept of hierarchical intent handling specified in TM Forum standards [20].

Furthermore, for all intents that an intent handler receives, it is expected to report the progress and status back to the source of the intent, creating a feedback loop that is instrumental for adaptation to dynamic intents.

4.7 Intent Life Cycle and Operations

The intent objects are subject to life cycle management. Some aspects of intent life cycle management have already been covered by a 3GPP study [6]. In 3GPP, after an intent is received and instantiated, it can be in any one of the states *inactive, active or null*, and transition between these states.

Considering the use of RDF as a meta-model for intents, every intent object will have a globally unique identifier (IRI) that allows an intent

handling function to operate on these objects. After intents are created and activated they may need to be fulfilled by the management system. During this process that may involve multiple autonomous entities, e.g. CLs, new intent objects may be created and activated, or deleted. These new intent objects can be used for communicating the requirements between levels of management or between different MDs. We see the need for 3 other operations that are needed in the intent life cycle management for the realization of E2E management involving intent-driven CLs: delegation, reporting, and escalation.

4.7.1 Intent delegation

Intents are originally received from a human-facing portal or an automated service ordering system and it is expressed by an intent object as exemplified in Section 4.5. Considering the ZSM framework, the initial intent is received by the intent handling function at the E2E service management domain and it is used as inputs to the CL(s) at that level. In the case of E2E management or cases where the resulting actions from a CL have to be sent to another autonomous MD, intents can be used as a way of communication and one intent handling function can delegate an intent (or parts of it) to another intent handling function located at the target autonomous MD. In the ZSM framework, the delegation can be sent from the E2E service MD downwards to an individual MD, or even to different MDs.

In a delegation operation, the sending and receiving intent handlers need to agree on the semantics of the intent exchanged, including the information models used. The intents' expectations have to be decomposed and delegated to different MDs according to their scope of management, e.g. the E2E service MD may only delegate intent expectations related to UE density and UE velocity (as shown in the example of Section 4.5) to the Radio Access Network (RAN) MD, while expectations related to UE throughput, delay and availability may be delegation to RAN, transport and core network MDs. The receiving entity can accept or reject an intent delegation. If accepted, the intent is now considered in the MD and proper CLs need to be used (or instantiated if needed) to fulfill the new intent. Delegated intents can be merged with existing ones.

Finally, an intent delegation can be followed by new (sub)intents delegations if the intent fulfillment involves other MDs. The intent delegation is a continuous process that may also involve the modification and/or removal of previously delegated intents, as the autonomous MDs are able (or not) to fulfil the expectations.

4.7.2 Intent reporting

Whenever an intent is delegated to an MD, the intent handler is expected to report on its fulfilment. The reporting has to be sent to the originating intent handler and has to be done while the intent exists in the MD.

The reporting can be done by sending the (delegated) list of expectations with the current measured status. The reporting can include more specific metrics evaluated. The period and/or criteria for reporting can be set during the delegation operation or can be set independently for individual or set of active intents in an MD. Reports can be periodic, event-based, or polling-based.

The intent handler that is receiving the reports may decide to adjust the currently delegated intents according to the current reporting status. For instance, if delegated intents are not fulfilled properly, they may need to be updated or other MDs may be involved; if temporary intents are reported to be fulfilled, they may be removed from a given MD.

4.7.3 Intent escalation

It may happen that an MD is not capable of fulfilling a (set of) intent(s) that it has received, either originally from a service request or from another MD that has delegated intents. The inability can happen in two cases, (i) the MD has tried to fulfill a delegated intent either by itself or by re-delegating and has not succeeded, because of lack of resources and/or knowledge, (ii) there may be more than one solution available and the MD that received the intent needs help on the judgement of the best solution. Escalations, therefore, may involve an issue to be considered by the upper MD or a judgement request based on risk measurements

Different to the reporting operation, an intent escalation is expected to be followed by some decision-making in the receiving intent handler, and it usually involves adjustments to the currently delegated intents or the addition of new intents.

5 Intent-driven Closed Loops Within the ZSM Framework

Considering the current specification of closed-loop automation within the ZSM framework (Section 2), the new approach of intent-driven management (Section 3.3) and the formal specification of intents and intent handling functions (Section 4.6), we show how to use intents as a universal mechanism for communication between the different layer and MDs within a management system based on the ZSM framework.

Principle 10 of ZSM framework [15] says that it supports intent-driven interfaces aiming to "hide complexity, technology- and vendor-specific details from the user by exposing high-level abstractions". Even though the specification provides all MnS and capabilities in high-level abstraction that could, in principle, use intents as a means for communication between MnS providers and consumer, similarly to the intent-driven management approach specified in [6], the way intents objects are handled between the layers and different MDs within the ZSM framework is completely ignored.

Applying intents to set the goals and requirements for each MDs requires the employment of intent handling functions, as explained in Section 4.6. We propose to realize an intent handling function in the ZSM framework as an entity that plays the role of service producer and consumer of an intent handling MnS. This MnS should be available at the E2E service MD as well as at all the MDs. This new MnS is responsible for the life cycle management of intent objects and supports the capabilities for the 3 main intents operations listed in Section 4.7. The intents objects are stored and managed within each MD with the aid of the Data services.

The main purpose of intent handling functions is to automatically close the gap between network state and the state expressed by the intents. The use of CLs as a building block for automating this process, as discussed in Section 3.3, is a way that has been considered in different SDOs. The connection between the intent handling functions and the CLs can be made with the CL Governance MnS, as specified in ZSM009-1. The CL Governance MnS provides the capabilities to manage the CL models and the life cycle of CLs.

This way, for each intent that is received (e.g., delegated), the intent handling function can use existing CL instances or use the CL models to instantiate new ones that can fulfill the intent. As more than one intent may be delegated to an intent handling function, further operations in the intent life cycle management may be necessary, such as merging intents, detecting and resolving conflicts, decomposing or recomposing intents, etc. All these other operations are internally abstracted by the intent handling function and do not need to be exposed.

The translation from intents to CL lifecycle management operations may involve, besides instantiation of new CL instances, the assignment of their goal(s) for the intent fulfillment. This translation can use the Manage CL models capability from the CL Governance MnS to set the goal(s) of existing or new CLs. ZSM009-1 specifies that goal(s) can be stated in declarative or imperative forms; the former in a level of abstraction closer to intents and the

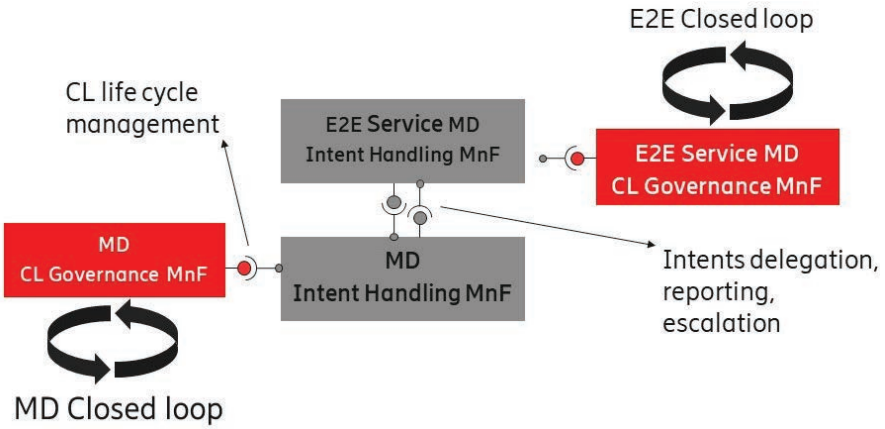

Figure 7 Intent handling interactions and the connection to closed loop instances.

latter with measurable service levels specifications. Depending on the type of CL employed at the MDs the translation process may involve converting intents into measurable KPIs to be monitored by the CL instance.

After a CL is associated with a (set of) intent(s), the intent handling function needs to report the fulfillment to the layer that originated that intent(s). The reporting information can be obtained using the Provide CL status information and/or Provide performance information capabilities from the CL Governance MnS. The status information has to be converted into intent-level information before it is sent back to the intent handling function that originated the intent(s).

During the fulfillment of the intent(s), issues or situations may occur where the upper layer needs to be involved in an escalation. The escalations may be processed by the intent handling functions, without necessarily involving the CLs, only based on the status information polled from the CL instances.

Figure 7 shows an illustration of the communication between two intent handlers at different management layers and the involvement of respective CLs.

We provide an example of how a service intent can be exchanged between the different intent handlers at different MDs. We consider the ZSM framework deployment as in Figure 2, where there are 3 MDs, namely RAN, transport, and CN, and the service intent as exemplified in Section 4.5. The

intent handler the E2E service MD decomposes the service intent into 3 resources intents, one for each MD, as shown below:

5.1 Resource Intent for RAN MD

```
:urllcServiceRANIntent rdf:type owl:NamedIndividual ,
    :Intent ;
    rdfs:comment "Intent to assure an URLLC service
    in the RAN MD" ;
    :hasExpectation :urllcAvailabilityExpectation ,
                    :urllcDeviceVelocityExpectation ,
                    :urllcMaxThroughputExpectation ,
                    :urllcSliceQoSExpectation ,
                    :urllcUeDensityExpectation .
```

5.2 Resource intent for CN MD

```
:urllcServiceCNIntent rdf:type owl:NamedIndividual ,
    :Intent ;
    rdfs:comment "Intent to assure an URLLC service
    in the CN MD" ;
    :hasExpectation :urllcAvailabilityExpectation ,
                    :urllcMaxThroughputExpectation ,
                    :urllcSliceQoSExpectation .
```

5.3 Resource intent for Transport MD

```
:urllcServiceTransportIntent rdf:type owl:NamedIndividual ,
    :Intent ;
    rdfs:comment "Intent to assure an URLLC service
    in the Transport MD" ;
    :hasExpectation :urllcAvailabilityExpectation ,
                    :urllcMaxThroughputExpectation ,
                    :urllcMaxDelayExpectation ,
                    :urllcMaxPERExpectation .

:urllcMaxDelayExpectation rdf:type owl:NamedIndividual ,
        :MaxDelayExpectation ;
    :target :urllcService ;
    :parameters [ :packetDelayBudget 5 ] .
```

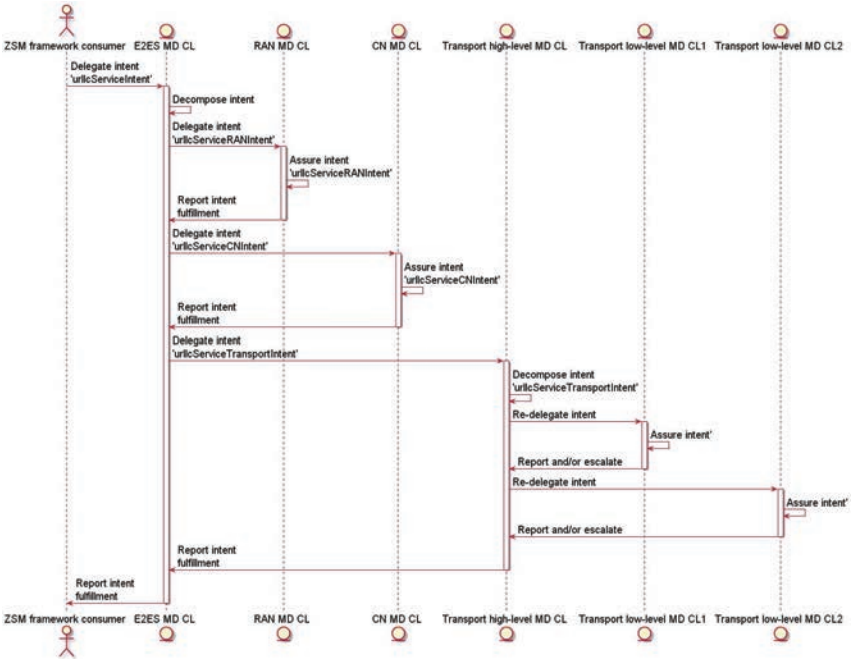

Figure 8 Example of a sequence diagram with intent operations involving different MD levels.

```
:urllcMaxPERExpectation rdf:type owl:NamedIndividual ,
        :MaxPERExpectation ;
    :target :urllcService ;
    :parameters [ :packetErrorRate "99.999"^^xsd:float ] .
```

The 3 different resource intents above are delegated for each corresponding MD and used to life cycle manage (i.e. instantiate, operate, etc.) the necessary CLs. Reporting should follow the intents delegation, and optionally new intents delegations may take place in MDs with subordinate levels, as in transport MD where 2 other lower-level CLs are used to assurance different segments. A sequence diagram with a (non-exhaustive) list of intent operations that may follow the initial intent delegation is shown in Figure 8.

We conjecture that higher levels of autonomy and flexibility can be obtained with the use of intents as the means of communicating expectations between MDs and their related closed loops that can be dynamically created with the use of intent handling functions. The 3 main intent life cycle operations, i.e. delegation, escalation, and reporting, are sufficient to provide

high-level interactions between the autonomous MDs and guarantee proper service operation. We show in this paper how to apply these concepts to an example of service assurance, connecting the E2E (slice) specification to domain-specific goals. The same concepts can also be applied to increase autonomy in the execution of other network management tasks, such as fulfilment, fault management, etc.

6 Related Work

CLA and intent-driven management are interrelated topics that have been addressed in different places in the telecommunications industry.

CLs have been commonly (and successfully) used in many different industries for decades. There is a multitude of related works in the literature that leverage CLs approach to optimize decision-making processes, mostly inspired by the military field where OODA (Observe, Orient, Decide, Act) CL model [9] was proposed. In the 2000s the MAPE-K (Monitor, Analysis, Planning, Execution, Knowledge) was an architecture proposed for autonomic computing, and it has changed the way autonomous functions are created and executed in distributed IT systems [13]. MAPE-K has inspired the way automation is done and how CLs are modeled. The stages of CLs within ZSM detailed in ZSM009-1 are also based on the MAPE-K model. Other types of CL have also been proposed for autonomous network management, where specific assets such as information models, policies, and reasoning capabilities are integrated; examples of these are FOCALE and COMPA loops [26]. Even though different types of CLs exist and have been applied to solve different use cases, all of them share some commonalities, which are the split of tasks between different stages (usually 4 or less) and a way of sharing and storing data that are relevant for the stages and the CL as a whole.

Even though some recent works have proposed reliable mechanisms for the integrating multiple CLs [29, 24] and the coordination of CLs has been addressed in other fields, e.g., supply chain [22, 14], the coordination of multiple CLs is still an open question for research in telecommunications. Telecommunications systems are highly stochastic and not well-bounded systems, which makes it harder to create models required for the application of control theory or learning techniques, for instance from the field of game theory. While trying to cope with this challenge and also aiming to increase the autonomy of different management services and functions, a new research and development trend has emerged for intent-driven management [10, 7].

In the intent-driven management approach, the communication between different parts of the management system is expressed by intents, which is responsible for setting all requirements expected by the entity that generated the intent.

The study 3GPP TR 28.312 [4] proposes the concept of intent-driven management services, with the goal of simplifying ways of managing the 5G network complexity. Management services (MnS) interactions are based on intents, where the MnS consumer expresses its desires and the MnS producer is responsible for translating them into executable actions, which may involve management tasks or activating specific service policies, as well as evaluating the result of these actions, i.e. checking if the intent is fulfilled or not. In principle, this new paradigm can be as a replacement for standardized reference interfaces or as additional to the existing ones. And the intent fulfillment could leverage CLA mechanisms to keep track and report the intent status. TM Forum's Autonomous Networks project has also worked on using and defining intent-driven management [20]. They have defined that the key needed capabilities for autonomous networks are: (i) simplified infrastructure, (ii) CLs, (iii) autonomous domains, and (iv) intent-driven interactions. Intent-driven interactions are the main mechanism in support of CLs across different layers in the management architecture. The TM Forum's framework is composed of 3 layers and 4 CLs. The layers are separated as business, service, and resources operations. There is one CL between each of the 3 layers, i.e. business CL (between business and services layers), service CL, and resources CL. Each of these 3 main CLs is responsible for integrating the different layers and their communication has to be simplified, business-driven, and technology-independent, which should be accomplished employing intents. The last CL is the vertical cross-layer user CL, which is the main streamline between business, service, and resources layers, and it is responsible to support service fulfilment.

The use of intents in networking can be traced back to 2015 with its application to SDN as the means for specifying north-bound interfaces of controllers [25]. The use of intent-driven interfaces aimed at removing the complexity of service requests and making the SDN systems easier to integrate with other systems. For the past 5 years, intents have been explored by academia, as well as players of different sizes in the networking and communications industries, open-source projects, and standardization organizations. The work [28] presents a survey on intent-driven networks and gives a very comprehensive overview of recent activities. Overall what we have witnessed in all different ways of applying intents is the importance of increasing the

flexibility and the abstraction in the communication between different entities (consumers and providers) in networking systems, whether these are service requesters and an SDN controller, end-points of management services or different CLs for service assurance.

7 Conclusions and Future Work

Closed-loop automation and intent-driven management are two important technologies for the successful realization of zero-touch operations and, consequently, the success of next-generation services with dynamic and stringent requirements. They have been used for some time in the industry and more recently have been the target of many standardization activities since automation is increasing in multi-vendor scenarios and flexible and vendor-neutral solutions still need to be developed. In this paper, we proposed the employment of intent-driven closed loops to allow higher autonomy levels of management within the MDs, while ensuring end-to-end service requirements. The proposal covers the formal specification of intents, intent meta-models, and the hierarchy of intent handling functions. We considered the ZSM framework the baseline for applying these concepts into a zero-touch service management system and provided some additional features and capabilities that would be needed for ZSM to realize the concept of intent-driven closed loops. An example of how intents could be used to convey service-level requirements is provided for a use case that considers URLLC service assurance in a deployment with 3 different MDs and 3 layers of CLs. The proposed new capabilities and the new approach for autonomous MDs based on intent management and closed loops can be considered for future standardization activities in ETSI ZSM or other standardization groups. Implementation of this solution and interoperability based on the intent meta models and standardized intent handling functions are also topics for further investigation.

References

[1] RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/rdf11-concepts/. Accessed: 2020-09-01.

[2] RDF 1.1 Turtle – Terse RDF Triple Language. https://www.w3.org/TR/turtle/. Accessed: 2020-09-01.

[3] Zero touch network & Service Management (ZSM). https://www.etsi.org/technologies/zero-touch-network-service-management. Accessed: 2020-09-01.

[4] 3GPP. Management and orchestration; Intent driven management services for mobile networks (Release 17). Technical Specification (TS) 28.312, 3rd Generation Partnership Project (3GPP), 09 2020. Version 0.1.0.

[5] 3GPP. Study on concept, requirements and solutions for levels of autonomous network; (Release 16). Technical Report (TR) 28.810, 3rd Generation Partnership Project (3GPP), 08 2020. Version 1.1.0.

[6] 3GPP. Study on scenarios for Intent driven management services for mobile networks (Release 17). Technical Report (TR) 28.812, 3rd Generation Partnership Project (3GPP), 09 2020. Version 17.0.0.

[7] Fred Aklamanu, Sabine Randriamasy, Eric Renault, Imran Latif, and Abdelkrim Hebbar. Intent-based real-time 5g cloud service provisioning. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2018.

[8] M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter, S. Jiang, and L. Ciavaglia. Autonomic networking: Definitions and design goals. RFC 7575, RFC Editor, 06 2015.

[9] John R Boyd. The essence of winning and losing, 1996.

[10] Walter Cerroni, Chiara Buratti, Simone Cerboni, Gianluca Davoli, Chiara Contoli, Francesco Foresta, Franco Callegati, and Roberto Verdone. Intent-based management and orchestration of heterogeneous openflow/iot sdn domains. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–9. IEEE, 2017.

[11] Alexander Clemm, Laurent Ciavaglia, Lisandro Granville, and Jeff Tantsura. Intent-based networking – concepts and definitions. Internet-Draft draft-irtf-nmrg-ibn-concepts-definitions-02, IETF Secretariat, 07 2020.

[12] Alexander Clemm, Mohamed Faten Zhani, and Raouf Boutaba. Network management 2030: Operations and control of network 2030 services. *Journal of Network and Systems Management*, pages 1–30, 2020.

[13] Autonomic Computing et al. An architectural blueprint for autonomic computing. *IBM White Paper*, 31(2006):1–6, 2006.

[14] Pietro De Giovanni. Closed-loop supply chain coordination through incentives with asymmetric information. *Annals of Operations Research*, 253(1):133–167, 2017.

[15] ETSI. GS ZSM 002 V1.1.1: Zero-touch network and Service Management (ZSM); Reference Architecture. Group specification (gs), 08 2019.
[16] ETSI. GS ZSM 002 V0.19.2: Zero-touch network and Service Management (ZSM); End to end management and orchestration of network slicing. Group specification (gs), 09 2020.
[17] ETSI. GS ZSM 005 V1.1.1: Zero-touch network and Service Management (ZSM); Means of Automation. Group specification (gs), 05 2020.
[18] ETSI. GS ZSM 009-1 V0.9.1: Zero-touch network and Service Management (ZSM); Closed-loop automation; Enablers. Group specification (gs), 09 2020.
[19] Liam Fallon, John Keeney, Mark McFadden, John Quilty, and Sven van der Meer. Using the COMPA autonomous architecture for mobile network security. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 747–753. IEEE, 2017.
[20] TM Forum. Autonomous Networks – Business requirements & architecture. Technical Report IG 1218, TM Forum, 07 2020. Version 1.0.0.
[21] TM Forum. White Paper – Autonomous Networks: Empowering digital transformation for smart societies and industries. Technical report, TM Forum, 10 2020. Release 2.
[22] JY Ge, PQ Huang, and ZP Wang. Closed-loop supply chain coordination research based on game theory. *Journal of Systems & Management*, 5:016, 2007.
[23] GSMA. NG.116 – Generic Network Slice Template – Version 3.0. Technical report, 05 2020.
[24] Sharmin Jahan, Ian Riley, Charles Walter, Rose F Gamble, Matt Pasco, Philip K McKinley, and Betty HC Cheng. Mape-k/mape-sac: An interaction framework for adaptive systems with security assurance cases. *Future Generation Computer Systems*, 2020.
[25] C Janz, N Davis, D Hood, M Lemay, D Lenrow, L Fengkai, F Schneider, J Strassner, and A Veitch. Intent nbi–definition and principles. Technical report, 2015.
[26] Liam, Fallon and John, Keeney and Ram Krishna, Verma. Autonomic Closed Control Loops for Management, an idea whose time has come? In *International Conference on Network and Service Management*, 2019.

[27] Jörg Nïemöller, Leonid Mokrushin, Swarup Kumar Mohalik, Martha Vlachou-Konchylaki, and George Sarmonikas. Cognitive processes for adaptive intent-based networking. *Ericsson Technology Review*, 2020.

[28] Lei Pang, Chungang Yang, Danyang Chen, Yanbo Song, and Mohsen Guizani. A survey on intent-driven networks. *IEEE Access*, 8:22862–22873, 2020.

[29] Adja Ndeye Sylla, Maxime Louvel, Eric Rutten, and Gwenaël Delaval. Design framework for reliable multiple autonomic loops in smart environments. In *2017 International conference on cloud and autonomic computing (ICCAC)*, pages 131–142. IEEE, 2017.

[30] ETSI ZSM. Zero-touch Network and Service Management – Introductory White Paper. Technical report, ETSI, 12 2017.

Biographies

Pedro Henrique Gomes is a researcher at Ericsson Research, Brazil, engaged in orchestration and automation of 5G networks and services. He is a delegate in the ETSI Zero-Touch Network & Service Management working group, contributing to the architecture definition especially with AI and ML concepts, and driving the specification of enablers for closed-loop automation in end-to-end network services. Received Ph.D. (2019) and M.Sc. (2015) in electrical engineering from the University of Southern California, Los Angeles, USA, also M.Sc. (2011) in computer science and B.Sc. (2007) in computer engineering from the University of Campinas, Brazil.

Magnus Buhrgard is Open Source and Standardization Manager at Ericsson. He is engaged in ONAP, where he is responsible for coordination with SDOs regarding network management. He is also Ericsson's primary delegate to ETSI ISG ZSM. With 30 years of experience in the telecommunication industry, his work has spanned a multitude of architectures and technologies, reaching from fiberoptic research to network and service automation. He has a long-term engagement in carrier-grade network architecture. Magnus is also engaged in the challenge of merging standardization, open source, and R&D ways of working. He holds an M.Sc. in Engineering Physics from Lund University, Sweden.

János Harmatos is a senior researcher at Ericsson Research. Currently he works on real-time cloud framework and edge-cloud integration in industry networks. He received his PhD from Budapest University of Technology and Economics.

Swarup Kumar Mohalik is a principal researcher at Ericsson Research who joined the company in 2015. His expertise is in the areas of AI and formal methods, and his work primarily focuses on applying them to service automatization and the Internet of Things (IoT). He has research experience in the areas of formal specification and verification of real-time embedded software and AI planning techniques. Mohalik holds a Ph.D. in computer science from the Institute of Mathematical Sciences, Chennai, India, and a postdoctoral fellowship at LaBRI, University of Bordeaux, France.

Dinand Roeland is a Principal Researcher at Ericsson Research. His current research interests are on introducing AI technologies in the end-to-end network architecture. He received an M.Sc. cum laude in Computer Architectures and Intelligent Systems from the University of Groningen, the Netherlands.

Jörg Niemöller is an expert in analytics and customer experience. He joined Ericsson in 1998 and has since held multiple positions in research as well

as in system management for core network and digital services. His current focus is innovation in OSS through architecture and solutions for autonomous network and service operation. Niemöller holds a Ph.D. in computer science from Tilburg University, the Netherlands, and a diploma degree in electrical engineering from the TU Dortmund University, Germany