Creating autonomous networks with intent-based closed loops

To create intelligent networks with the ability to adapt to new situations and changing needs, it will be necessary to decouple requirements from solutions. Intent is the expression of the requirements that an autonomous system needs to meet, which makes it a key concept in the creation of intelligent networks.

JÖRG NIEMÖLLER,
RÓBERT SZABÓ,
ANDRÁS ZAHEMSZKY,
DINAND ROELAND

Intent-based operation is a new paradigm for telecommunication systems that is essential to the creation of autonomous networks.

■ Intent is a declarative information object that defines the requirements that an autonomous system and infrastructure are expected to fulfill [1]. An intent is never imperative: the receiving system is not instructed to perform a particular action or process. On the contrary, the system is free to choose a solution strategy autonomously.

Intent allows the system to understand global utility and the value of its actions. Consequently, the autonomous system can evaluate situations and potential action strategies rather than being limited to following instructions that human developers have specified in policies. This means that intelligent decisions that were previously made exclusively by human policy developers can become automated.

While intent builds the foundation by providing information about requirements and utility, the implementation of autonomous capabilities remains a challenging task. We believe it can best be achieved by a combination of artificial intelligence techniques such as machine learning and machine reasoning using knowledge-centric architecture and processes

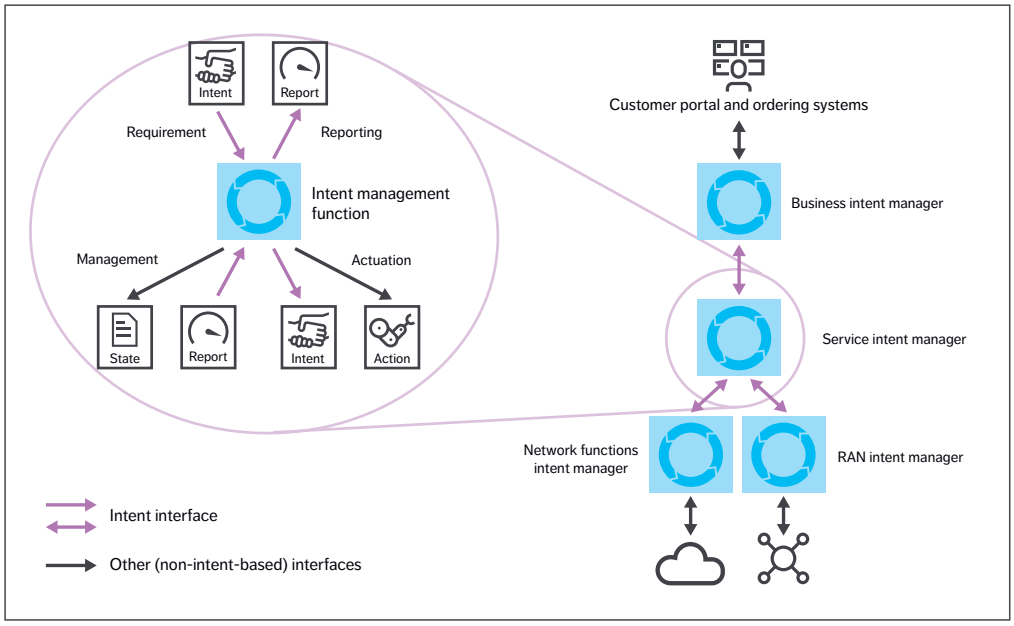

Figure 1 The intent management function and intent-based networking system stack

[2, 3, 4]. The result is a cognitive network based on the vision of 6G as an innovation platform for emerging use cases and applications [5].

Concepts of intent-based autonomous operation

The TM Forum Autonomous Network Project describes a reference architecture for network operation software that is able to act autonomously [6]. It defines a new function called the intent management function (also known as the intent manager) to coordinate intent-based operation [1].

Every autonomous domain contains an intent manager. This is the endpoint of the intent interface for managing the life cycle of the intent [7]. It also coordinates the operation within the autonomous domain to meet the requirements that the intent formulates. This implies that it is able to detect whether the intent has been fulfilled and, if not, find and execute corrective actions.

The left side of *Figure 1* shows the main interactions of the intent management function within a layered operation infrastructure. The intent management function receives all intents directed toward its autonomous domain. It also reports back to the intent origin regarding its success in fulfilling the intent, which closes an intent-based control loop.

The intent manager is aware of the system state through measurements, analytics and other information systems such as inventories. Comparing the state to the intent shows if and where the system is not meeting the requirements and indicates whether or not corrective action is needed. As soon as the intent management function finds a preferential action strategy, it executes on it. It can act through conventional management interfaces or, if the targeted autonomous domain is intent-aware, it can act by defining its requirements through an intent.

At the highest level, intent-based operation could

Terms and abbreviations

SMO – Service Management and Orchestration
UE – User Equipment

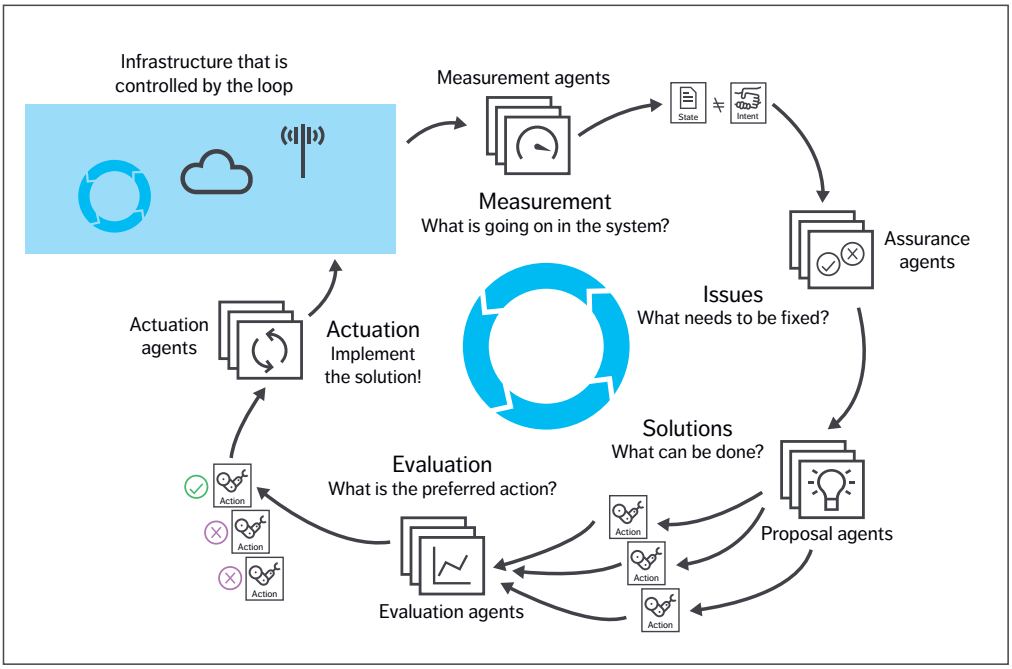

Figure 2 The cognitive intent management loop driven by machine reasoning

start with customer and operator requirements. Autonomous domains coordinated by their intent management function would then make partial decisions resulting in a more specific intent being issued to the next level of operation. This process ultimately arrives at a detailed distributed configuration of the network capable of meeting the high-level requirements that initiated the process.

The right side of Figure 1 provides an example of three levels of intent-based operation. The intent manager at the service level is part of a service management and orchestration (SMO) system. In this example, the solution at the service level has impacts on the network function management as well as on the RAN.

Cognitive intent management loop

The intent control loop between autonomous domains is covered by standardization [1, 7]. The remaining challenge is how the intent manager interacts with the specific business logic within an autonomous domain, so that they can together translate an intent that expresses business needs into detailed technical configurations.

Figure 2 illustrates how we have implemented an intent management function. We have established a generic cognitive loop that consists of five components: measurement, issues, solutions, evaluation and actuation.

Measurement of the system state is a continuous activity used both to set goals (expressed as intents) and to evaluate fulfillment of intent. The intent manager needs to know which resource instances are used to fulfill the intent as well as their operational state. Knowledge of the current performance of service components at their resource instances makes it possible to determine if the system is in breach of its intent and needs corrective

action. However, as the intent can change, so can the required measurements. The intent manager must adapt its measurements accordingly by employing measurement agents (specialist implementations of measurement tasks) as required.

The issues stage focuses on identifying what needs to be fixed. Issues are defined as requirements (expressed by intents) that are not met by the system. Examples include a required QoE metric that is not reached or a required resource that is not available. Assurance agents implement continuous monitoring of state versus requirements and raise issues when required. This can include a root cause analysis for a more precise description of the issue, prioritization of issues depending on their severity or waiting for ongoing actions showing results.

In the solutions stage, proposal agents react to issues and determine the available corrective action strategies. Any number of differently implemented proposal agents can be used at this stage, including human designed policies and machine learned policies that derive solutions from evidence data. Multiple action strategies may be proposed, each of them representing what the system can do to address the issue.

In the evaluation stage, evaluation agents determine the expected impact of an action on the system state to determine which of the proposed solutions can deliver the most positive effect on the fulfillment of all currently valid intents. Predictive models and digital twins will play a major role in this step, as they make it possible to virtually explore actions and their expected outcomes. The preferred action proposal is the one expected to maximize global utility by best fulfilling all intents. The evaluation stage can also include detection of conflict from actions that fulfill one intent at the expense of degrading another. Evaluation can therefore be a sanity mechanism with the potential to save the network from risky actions and degradations.

A solution with an overall preferential evaluation proceeds to the actuation stage. Intent-based systems can act by using an intent to define requirements on the autonomous subordinate system layer. Alternatively, they may act through traditional interfaces by changing configurations of invoking processes. Specialized actuation agents are available to implement a different type of action-taking. For example, in service intent management, a proposal may be expressed by a TOSCA (Topology and Orchestration Specification for Cloud Applications) model and the actuation would therefore be orchestration.

> “THE ENTIRE INTENT MANAGEMENT LOOP IS BASED ON AGENTS CREATING AND CONSUMING KNOWLEDGE”

The entire intent management loop is based on agents creating and consuming knowledge. Agents both react to knowledge changes and create knowledge. For example, proposal generation is initiated by the appearance of the type of issue that the proposal agent was implemented to handle. It delivers a proposal for the issue through the creation of a proposal description in the knowledge database.

Intents are introduced in standards [8, 9] as ontology graphs. To enable a complete knowledge-centric implementation of the intent management loop, Ericsson has defined additional ontologies for knowledge about state, issues, proposals and predictions. This allows agent coordination through a generic logical-reasoning-based implementation. This loop is highly self-adaptive to changing intents. As long as respective agents are available, the machine-reasoning-based intent manager will autonomously utilize them by composing a custom, ad hoc workflow.

Within an SMO system, apps would typically be used to implement domain-specific business logic. They constitute agents and as such participate in intent management.

The cognitive intent management loop in Figure 2 is not specific to any particular autonomous domain;

> IN OUR EXAMPLE, WE FOCUS ON INTENT-BASED MANAGEMENT OF A CLOUD-NATIVE FUNCTION

it is, rather, a template for implementing any intent manager. The creation of an intent manager for a particular autonomous domain is primarily an implementation task of domain-specific agents.

Identifying solutions through resolution

Based on the concept presented in Figure 1 and the work of the TM Forum Autonomous Network Project, we have designed an autonomous network with a multilayered architecture. Operation starts with the intent expressing abstract business requirements – a cost-efficient video service shall be delivered to the silver-level user group, for example. This specifies the subject of a video service, high-level preference, such as cost efficiency and the targeted user group as context.

A business intent manager would receive the intent with business requirements. It would need to translate this into an intent for a service intent management function. This translation, or resolution, would include selecting eligible service blueprints from onboarded service definitions. They would be further contextualized by interpretation and transformation of the abstract requirements into more concrete goals. Policies and apps would typically implement the transformation business logic. In our simple example, the customer group would imply certain locations where the service would need to be available. It would further translate the business requirements of the service into an expected user experience with goals about video quality and availability.

An intent management function associated with service management would receive these requirements about the needed service within an intent from the business intent manager. The main objective of the service intent manager would be the delivery of a service instance that fulfills this intent. A key aspect of this process would be the distribution of application and network function components over the network and cloud infrastructure. A solution proposal within the intent management loop on the service layer would describe the full topology of these artifacts.

Operator concerns, such as resource utilization, energy consumption and cost, would be considered by the policies that determine the solution. These concerns may also be expressed by an intent and constitute additional operator requirements.

In the solution-finding process, the requirements on the service would be further translated into lower-level requirements per component. For example, a network function would be required in a particular location with goals on latency and bandwidth. This would contribute to delivering the required user experience.

Executing the solution at service level typically involves a service orchestrator that coordinates the execution of distributed actions. Some of these actions may involve sending further intents into autonomous domains that are intent-aware and contain an intent management function. In our example, we focus on intent-based management of a cloud-native function. A cognitive intent management loop such as the one in Figure 2 also coordinates the operation at this intent management level.

The proposal agent used to determine a solution strategy is mainly concerned with identifying all atomic deployment function artifacts and allocating them in the data center. Deployment function candidates are still technology agnostic, but we assume that there is a direct mapping of each atomic function to some deployment artifacts that are understood by the orchestrator or virtualization manager of the administrative domain.

The deployment function candidates may contain other functional dependencies such as the existence of an operations and maintenance function. These dependencies would be recursively resolved until the process reaches a set of atomic functions that satisfies all dependencies and their connectivity

requirements. The process may also involve functional decomposition, when a logical function is substituted with a set of atomic components. For example, a gateway function is decomposed into user plane and control plane components, each adhering to some resiliency and scaling requirements.

Ultimately, the multiple layers of resolutions for a new business intent provide a service instance design that is broken down to locations (such as data centers and transport paths), deployment artifacts (represented by Helm charts, for example) and initial resource allocation with respect to bandwidth, QoS class, virtual central processing units, memory, storage and so on.

Solution proposals in each layer may come from a distinct set of different resolution agents, or a resolution agent may be able to propose multiple solution alternatives. In the next step of the cognitive intent management loop, the most preferential solution may be chosen by an evaluation logic in each layer.

Once the service is instantiated according to the preferred solution, information becomes available that enables service monitoring. This means the intent management loop becomes an assurance loop that takes corrective actions if the intent requirements are not matched by the measured system state.

Conflict detection and resolution

Typically, the solution that a proposal agent proposes aims to improve one metric, which can be problematic in cases with multiple requirements in a single intent or multiple simultaneous intents. As network resources are shared and limited, a proposal that aims to solve an issue related to one metric (when a requirement is not fulfilled) may have a negative side effect on another metric, resulting in unstable network behavior. Overcoming this challenge requires the system to consider the effect of any given action on all the metrics under evaluation, as opposed to just the one that caused the issue. In situations where multiple solutions to a particular issue are proposed, the intent manager needs to decide which action to approve.

Our implementation includes a conflict detection and a conflict resolution component. Conflict detection examines the outcome of a given action in terms of its effect on all the metrics in the system. If any of the metrics that were met before the action are likely to be violated if the action is taken, those potential negative side effects are considered to be conflicts. In the conflict resolution step (part of the evaluation in Figure 2), the intent manager evaluates whether it should approve or reject the action.

In the prediction step (also part of the evaluation in Figure 2), prediction agents estimate the effect of the given action on all the metrics in the system. In one option, the prediction is achieved by machine reasoning rules inserted into the system by domain experts. The rules are formulated through experience and highlight key correlations between action and measured metrics. For example, changing user plane priority for some user equipment (UE) means that the bandwidth will be shared differently among all the UEs. Consequently, a change in bandwidth means a change in QoE for a certain application. From this, it can be concluded that changing the user plane priority in the network will lead to a change in the QoE. Based on historical measurements, it is also possible to predict how much the QoE is expected to change. Another option for predictions is to utilize machine learning and train models that predict the outcome of the actions.

Next, an evaluation agent examines the proposed actions and decides which one to take. It uses the intent manager penalty to determine whether to execute the action. The intent manager penalty is calculated as the sum of the penalties of all the

> “OUR IMPLEMENTATION INCLUDES A CONFLICT DETECTION AND A CONFLICT RESOLUTION COMPONENT”

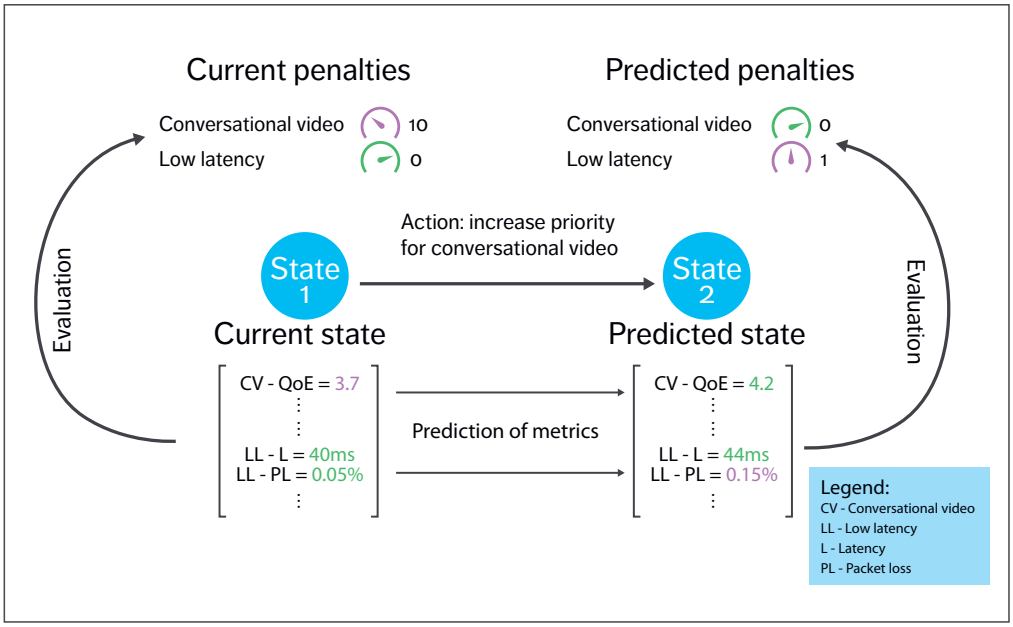

Figure 3 Conflict detection and resolution example

intents handled by the given intent manager. The penalty formula, which was submitted alongside the requirements of an intent, is used to calculate the penalty of the intent. The penalty formula incorporates the cost of not fulfilling the intent.

In its simplest form, it is a constant value, if at least one requirement is not fulfilled, and zero otherwise. In its more sophisticated form, the formula gives a higher value if the gap between the measured value(s) and the target value(s) of the metric(s) is high, and a lower one if the gap is low. As in the prediction step, the impact of each action on all the metrics is determined, so that the evaluation agent can calculate the expected intent manager penalty for each action. An action is approved if the intent manager penalty is not expected to increase after executing the action. In an extreme case, when all the proposed actions are predicted to increase the intent manager penalty, none of the actions will be approved.

The evaluation agent can give feedback to the proposal agent on the outcome of the action. For example, if an action was rejected, the proposal agent can incorporate this information to come to a better proposal in the next round. However, if after multiple attempts, the target of the corresponding metric is still not reached, the intent manager can utilize the escalation process to signal to the submitter of the intent that it cannot be fulfilled.

Figure 3 illustrates an example use case that features conversational video and low-latency services. The first intent expresses that a conversational video service is required and for 80 percent of the users the QoE should be higher than 4.0 (on a scale of 1 to 5). The second intent states that a low-latency service is required, where for 99 percent of the UEs using the service the packet loss should be less than 0.1 percent and the latency below 50 milliseconds. For the first intent, the penalty is 10, while for the second, it is 1. In other words, it is more

important to be able to fulfill the intent for the conversational video than the intent for the low-latency service, in cases of resource shortage situations.

Based on the assumption that only the conversational video service is active at the start, and all requirements are fulfilled, the intent manager penalty in the system will be zero. Then, at a later point, the intent for the low-latency service is added. After the required functions are deployed, and the UEs start to use the service, the target of the metrics for this service on packet loss and latency are met. However, as a consequence of sharing the limited network resources, the QoE for the conversational video users drops and the metric's current value is below the target. At this point, the intent manager penalty is 10 because one requirement of the intent for the conversational video is unmet. This situation is shown on the left side of Figure 3.

The fact that the QoE requirement is no longer being met invokes a proposal agent that is capable of proposing solutions to QoE issues. After considering the alternatives, it proposes to increase the user plane priority for the conversational video service.

Next, in the conflict detection step, a prediction agent runs the rules to see if the proposed action will have any side effects, and it discovers it will no longer be possible to meet the packet loss target for the other service if the proposed action is taken. The predicted state is shown on the right side of Figure 3.

The evaluation agent is invoked to resolve the conflict. It concludes that taking the action would change the intent manager penalty from 10 to 1, and therefore it decides to approve the action.

Conclusion

Our research indicates that a knowledge-centric and machine-reasoning-based implementation of intent management functions in networks offers considerable advantages with respect to dynamic self-adaptation to new situations and new requirements as they arise. This approach has the potential to achieve advanced levels of autonomy, far surpassing what would practically be possible with policy-driven automation. Agents have the ability to find solutions in complex network topologies and make evaluations based on digital-twin-style prediction models. The proof of concept that we have successfully implemented at Ericsson clearly demonstrates that the technologies needed to create highly capable autonomous networks are already available today.

> THE PROOF OF CONCEPT CLEARLY DEMONSTRATES THAT THE TECHNOLOGIES NEEDED TO CREATE HIGHLY CAPABLE AUTONOMOUS NETWORKS ARE ALREADY AVAILABLE TODAY

Further reading

- **Ericsson, Intent based networks, available at:** *https://www.ericsson.com/en/ai/intent-based-networks*
- **Ericsson Technology Review, Adaptive intent based networking, available at:** *https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/adaptive-intent-based-networking*
- **TM Forum Autonomous Network Project:** *https://www.tmforum.org/collaboration/autonomous-networks-project/*
- **TM Forum IG 1253:** *https://www.tmforum.org/resources/standard/ig1253-intent-in-autonomous-networks-v1-1-0/*

References

1. **TM Forum Autonomous Network Project, IG1253 Intent in Autonomous Networks v1.1.0, January 2022, available at:** *https://www.tmforum.org/resources/how-to-guide/ig1253-intent-in-autonomous-networks-v1-1-0/*
2. **Ericsson Technology Review, Cognitive technologies in network and business automation, June 28, 2018, Niemöller, J; Mokrushin, L, available at:** *https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/cognitive-technologies-in-network-and-business-automation*
3. **Ericsson Technology Review, Cognitive processes for adaptive intent-based networking, November 11, 2020, Niemöller, J; Mokrushin, L; Mohalik, S.K; Vlachou-Konchylaki, M; Sarmonikas, G, available at**: *https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/network-compute-fabric*
4. **Journal of ICT Standardization, Intent-driven Closed Loops for Autonomous Networks, June 8, 2021, Gomes, P.H; Buhrgard, M; Harmatos, J; Mohalik, S.K; Roeland D; Niemöller, J, available at:** *https://journals.riverpublishers.com/index.php/JICTS/article/view/5829*
5. **Ericsson Technology Review, The network compute fabric – advancing digital transformation with ever-present service continuity, June 30, 2021, Sefidcon, A; John, W; Opsenica, M; Skubic, B, available at:** *https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/network-compute-fabric*
6. **TM Forum Autonomous Network Project, IG 1230 Autonomous Networks Technical Architecture v1.1.0, July 2021, available at:** *https://www.tmforum.org/resources/how-to-guide/ig1230-autonomous-networks-technical-architecture-v1-1-0/*
7. **TM Forum Autonomous Network Project, IG 1253C Intent Life Cycle Management and Interface v1.1.0, January 2022, available at:** *https://www.tmforum.org/resources/standard/ig1253c-intent-life-cycle-management-and-interface-v1-1-0/*
8. **TM Forum Autonomous Network Project, IG1253A Intent Common Model v1.1.0, January 2022, available at:** *https://www.tmforum.org/resources/standard/ig1253a-intent-common-model-v1-1-0/*
9. **TM Forum Autonomous Network Project, IG 1253B Intent Extension Models v1.0.0, January 2022, available at:** *https://www.tmforum.org/resources/how-to-guide/ig1253b-intent-extension-models-v1-0-0/*

THE AUTHORS

Jörg Niemöller

◆ is an analytics and customer experience expert in Solution Area OSS. He joined Ericsson in 1998 and spent several years at Ericsson Research, where he gained experience working with machine-reasoning technologies and developed an understanding of their business relevance. He is currently driving the introduction of these technologies into Ericsson's portfolio of Operations Support Systems/Business Support Systems solutions. 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.

Róbert Szabó

◆ joined Ericsson in 2013 and currently works as a principal researcher at Research Area Cloud Systems and Platform, where he focuses on distributed cloud, zero-touch automation and Network Functions Virtualization. Before joining Ericsson, he worked as an associate professor at Budapest University of Technology and Economics (BME) in Hungary. Szabó holds both a Ph.D. in electrical engineering and an MBA from BME.

András Zahemszky

◆ is a master researcher at Ericsson Research whose work focuses on core networks and network automation. He is currently leading research prototype efforts in the area of end-to-end network automation. He joined Ericsson in 2007 after completing an M.Sc. in computer science at BME in Hungary.

Dinand Roeland

◆ is a principal researcher at Ericsson Research who joined the company in 2000. His current research interests involve introducing artificial intelligence technologies into end-to-end network architecture with the goal of achieving an autonomous cognitive network. He has worked in a variety of technical leadership roles including concept development, prototyping, standardization, system management and project management. Roeland holds more than 75 patents and an M.Sc. (cum laude) in computer architectures and intelligent systems from the University of Groningen in the Netherlands.