CHARTING THE FUTURE OF INNOVATION | #4·2017

ERICSSON TECHNOLOGY Review

COGNITIVE AUTOMATION AS AN IoT ENABLER

ERICSSON

Tackling IoT complexity WITH MACHINE INTELLIGENCE

Internet of Things (IoT) applications transcend traditional telecom to include enterprise verticals such as transportation, healthcare, agriculture, energy and utilities. Given the vast number of devices and heterogeneity of the applications, both ICT infrastructure and IoT application providers face unprecedented complexity challenges in terms of volume, privacy, interoperability and intelligence. Cognitive automation will be crucial to overcoming the intelligence challenge.

ANETA VULGARAKIS FELJAN, ATHANASIOS KARAPANTELAKIS, LEONID MOKRUSHIN, RAFIA INAM, ELENA FERSMAN, CARLOS R. B. AZEVEDO, KLAUS RAIZER, RICARDO S. SOUZA

The IoT is built on the concept of cross-domain interactions between machines that can communicate with each other without human involvement. These interactions generate a vast number of heterogeneous data streams full of information that must be analyzed, combined and acted on.

■ To be successful, providers of ICT infrastructure and IoT applications need to overcome interlinking challenges relating to volume, privacy, interoperability and intelligence. Doing so requires a multifaceted approach involving concepts and techniques drawn from many disciplines, as illustrated in *Figure 1*.

Research in 5G radio, network virtualization and distributed cloud computing primarily addresses the volume challenge, by evolving network infrastructure and application architecture design to increase the amount of resources available to applications (throughput, compute and store resources, for example). Meanwhile, the privacy and interoperability challenges are being addressed with a mix of R&D efforts and standardization activities that are leading to novel concepts and techniques, such as differential privacy, k-anonymity algorithms, secure multiparty computation, ontology matching, intelligent service discovery, and context-aware middleware layers.

Addressing the intelligence challenge requires the development and use of intelligent decision

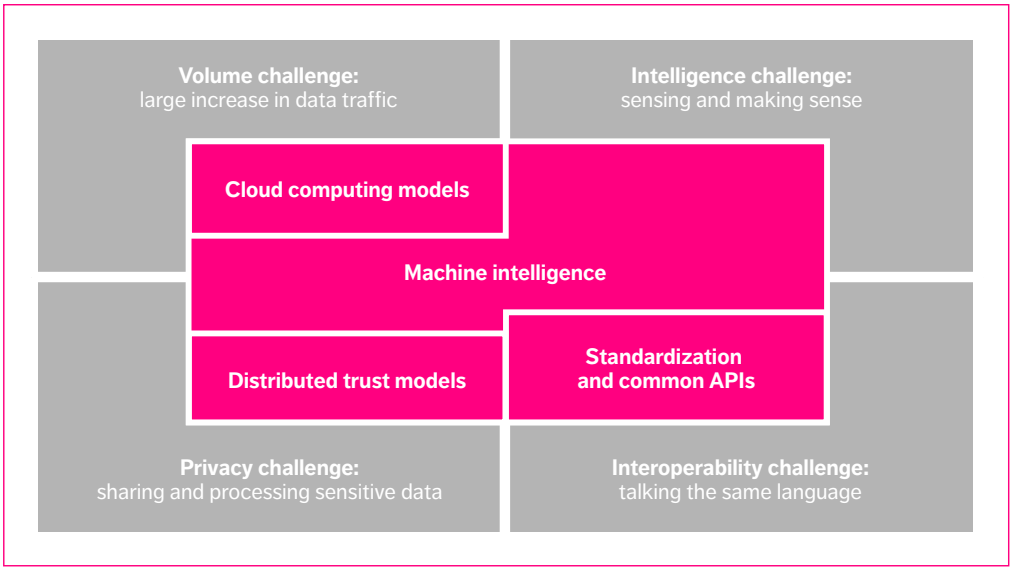


Figure 1
Key challenges (in gray) and approaches to addressing them (in pink)

support systems (DSSs) – interactive computer systems that use a combination of artificial intelligence and machine learning methods known as machine intelligence (MI) to assist and enhance human decision making. In response to this need, Ericsson has built a cognitive automation framework to support development and operation of intelligent DSSs for large-scale IoT-based systems. The use cases are not only the traditional telecom ones (such as operations support systems/business support systems automation), but new ones in domains such as transportation, robotics, energy and utilities, as well.

The main benefit of our cognitive automation framework is that it reduces DSS development and deployment time by reusing as much knowledge as possible (such as domain models, behaviors, architectural patterns, and reasoning mechanisms). It also cuts operational costs for IoT-based system management by making it possible during runtime for a DSS to decide automatically how to adapt to changes in its context and environment, with minimal or no human interaction.

While DSSs predominantly address the intelligence challenge, they can also be used to address privacy, volume and interoperability challenges. For example, IoT devices need intelligence to learn to trust each other. Management of network infrastructure may also benefit from intelligent systems, to handle the large volumes of transmitted data more efficiently. Finally, systems that use different standards and/or APIs can use MI to interface with each other, by using language games, for example.

Architectural components and structure of the knowledge base

A high-level conceptual view of the architecture of our framework is depicted in *Figure 2*. The main components are the observer, the knowledge base, the reasoner and the interpreter.

The observer is responsible for pushing knowledge from the environment to the knowledge base. This is done by first deriving a symbolic representation of data using model transformation

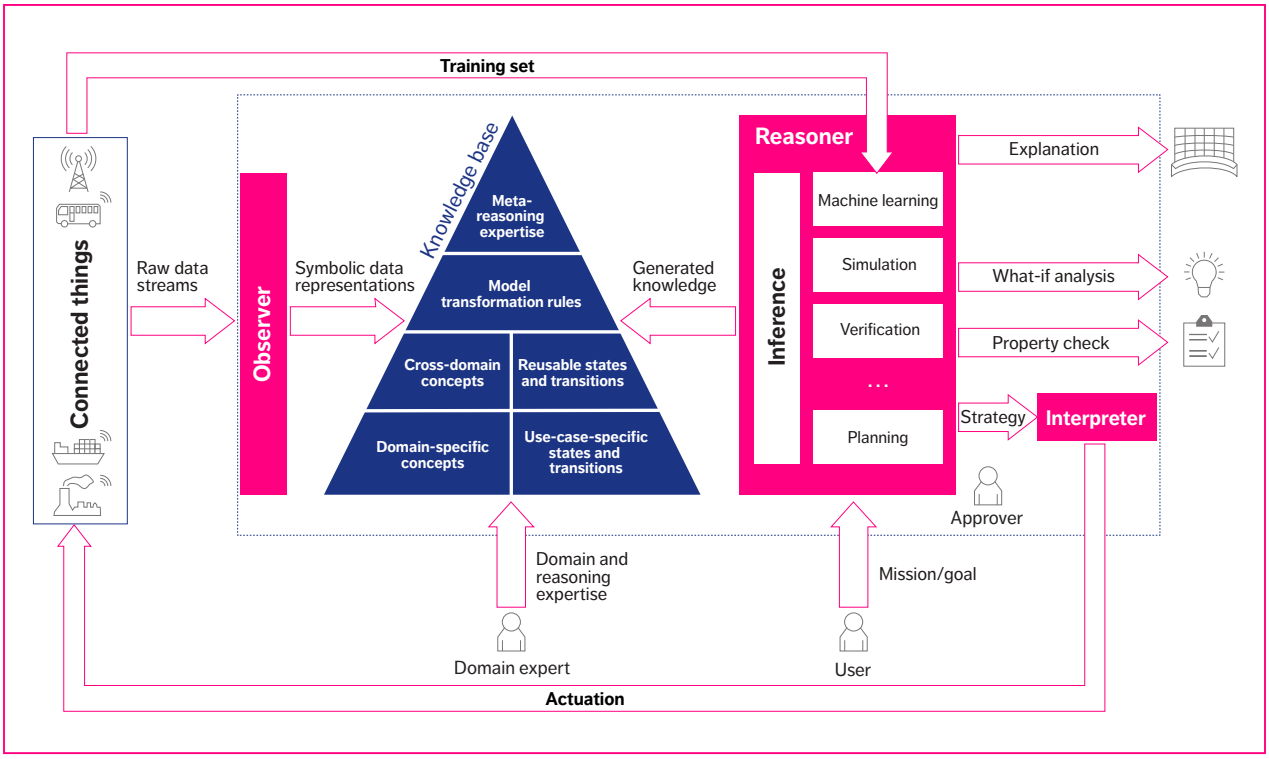


Figure 2
A high-level, conceptual view of the framework architecture

rules that identify the relation between raw data coming from the IoT-based system and a symbolic representation of the captured data. The next step is to perform ETL (extract, transform, load), followed by a semantic analysis of the obtained data.

The knowledge base contains a formalized description of general concepts that can be used across domains to facilitate interoperability, as well as domain-specific concepts (transportation, for example). In addition, the knowledge base contains the possible discrete states of the system, potential transitions between states, meta-reasoning expertise and model transformation rules. Ontologies in RDF/OWL semantic markup language are among the possible formats of the knowledge stored in the knowledge base [1].

Domain experts can enter domain and reasoning expertise directly into the knowledge base. Since the behavior of the environment is dynamic and unpredictable, the knowledge base is continuously updated with knowledge coming from the observer and the reasoner.

Most of the framework intelligence comes from the reasoner, which contains general purpose inference mechanisms that allow it to draw conclusions from information (propositions, rules, and so on) stored in the knowledge base, and problem-specific tools that implement various MI tasks such as machine learning, planning, verification, simulation, and so on. By relating a goal/mission query to meta-reasoning expertise, the inference engine selects an appropriate method or problem solver, and derives

relevant input for it using model transformation rules. The result produced by the selected problem solver can be presented to the user, sent to the interpreter for actuation, or fed back into the inference engine to reinforce the knowledge base content.

A goal query can be initiated by the user or the inference engine itself based on a comparison between expected and actual system states. For instance, if the goal query is to verify a certain system property (safety, for example), the inference engine will look in the knowledge base and deduce that a verification method should be used to check that the system satisfies the required property. Similarly, if the goal query is to reach a certain goal, the inference engine will look up the knowledge base and deduce that a planner should be used to generate a strategy to reach that goal. Since most of the planners accept Planning Domain Definition Language (PDDL) [2] as their input, the inference engine looks up corresponding model transformation rules to translate the problem in PDDL. The interpreter takes the generated strategy and transforms it into actuation instructions for the connected things. When executing the strategy, the interpreter works closely with the reasoner. In the event of any changes in the expected state of the system, the reasoner sends a replanning request to the planner. The generated strategy can be presented to the user for approval before actuation, if required.

The reasoner can perform automatic knowledge acquisition using machine learning techniques that may extract insights (such as categorization, relations and weights) from training sets in the form of documentation, databases, conversation transcriptions or images. The reasoner can explain the reasons behind recommending and performing a given set of actions, trace back to the origins of the decisions, and inform the user (and other systems, if so desired) about actions planned for the future. The explanation can be made in a user-friendly manner by applying natural language processing, augmented reality and automatic speech synthesis and recognition, for example.

“THE REASONER CAN PERFORM AUTOMATIC KNOWLEDGE ACQUISITION USING MACHINE LEARNING TECHNIQUES SUCH AS CATEGORIZATION, RELATIONS AND WEIGHTS”

Application examples

Our research and experience show that there are a wide variety of ways in which our framework can be used to boost efficiency in different types of applications. Test Automation as a Service (TAaaS), automated scheduling of train logistics services, smart metering, ticket book systems and Everything as a Service (XaaS) are just a few examples.

TAaaS

Our TAaaS project addressed the product development life cycle and software-testing activities. Testing for products typically involves numerous tools for test design, test execution and test reporting, along with management software tools, put together in a toolchain. Toolchain configuration is an expensive process that often takes one to two days to prepare, depending on the product being tested. Reconfiguring a toolchain to test another product is an even more time-consuming activity. The TAaaS system automated the configuration of these toolchains based on user requirements and created virtual workspaces in a data center, eliminating the need for expensive, manual configuration. In this case, the framework contained a knowledge base of software tools and their dependencies.

Automated scheduling of train logistics services

In this project, we created a concept for a fully automated logistics transportation system. The system consisted of several actors (such as trains, loading cranes and railroad infrastructure elements) equipped with sensors and actuators, and coordinated by intelligent software. The main goal was to investigate the potential benefits of combining factual and behavioral knowledge to raise the level of abstraction in user interaction. A user-specified, high-level objective such as "deliver cargo A to point B in time C" is automatically broken down to subgoals by the intelligent management system, which then creates and executes an optimal strategy to fulfill that objective through its ability to control corresponding connected devices.

Smart metering

We used our framework to create an intelligent assistant to help Ericsson's field engineers manage Estonia's network of connected smart electricity meters. Its task was to automate troubleshooting and repair procedures by utilizing knowledge captured from the domain experts. By separating knowledge from the control logic, we made the system extendable so that new knowledge can be added to it over time. The representation comprises several decision trees and workflows that encode the root causes of various potential technical issues, processes for diagnosing and identifying them, and the steps in the corresponding repair procedure. Our project revealed that in this type of use case, the bulk of the work is in the manual acquisition and encoding of expert knowledge.

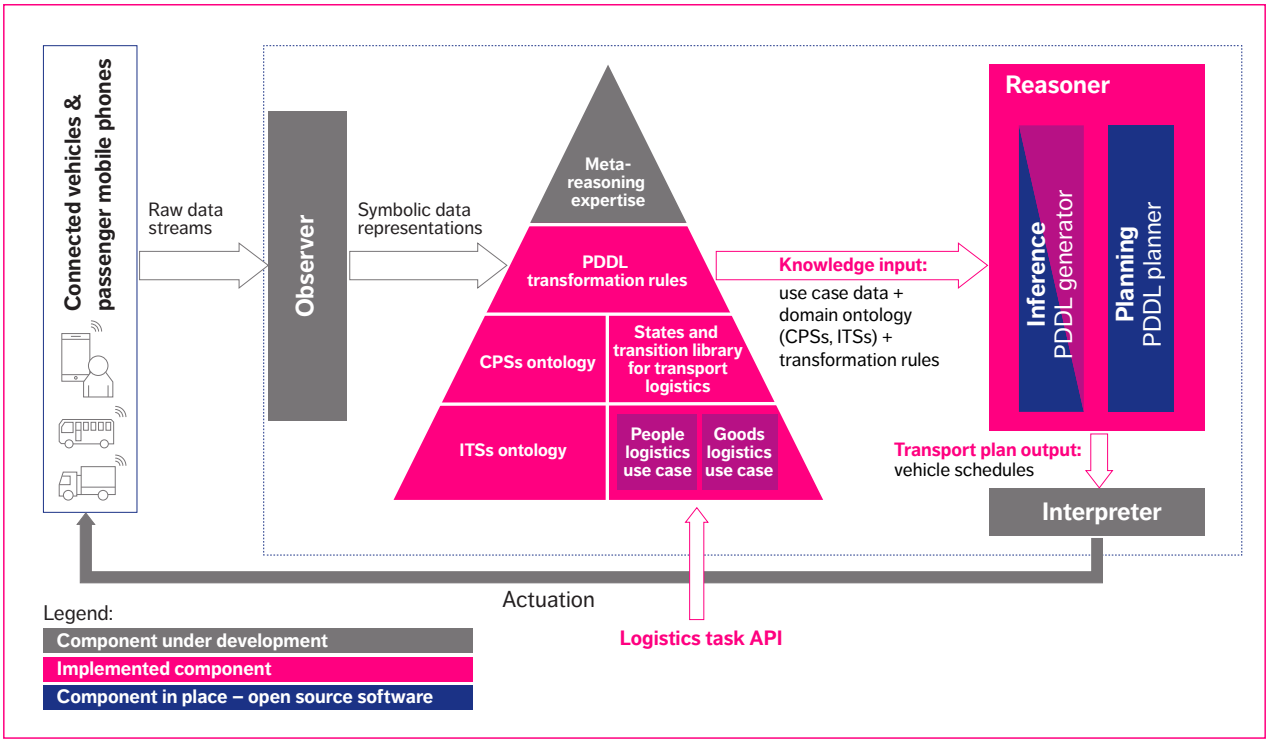
> "CONSUMER SERVICES CAN BE DELIVERED AUTOMATICALLY USING UNDERLYING MODULAR COMPONENTS KNOWN AS MICROSERVICES THAT CAN BE REUSED ACROSS DIFFERENT APPLICATION DOMAINS"

Ticket book

When managing and operating mobile networks, support engineers normally create tickets (issue descriptions) for problems classified as non-trivial. Based on the assumption that the solution to an issue similar to another that has previously been dealt with will be similar, we created a system for semantic indexing and searching historical tickets. We used a vector space model to calculate a similarity score and sorted searched tickets according to their relevance to the current case. A similar technique was used to find technical documentation relevant to the case currently being solved. This use case relies on the availability of historical data (previous tickets) to automatically build a knowledge base that facilitates subsequent information retrieval and relevancy ranking.

XaaS

In our XaaS project, we developed an MI-assisted platform for service life cycle management. By leveraging technologies such as semantic web and automated planning, we presented how consumer services can be delivered automatically using underlying modular components known as microservices that can be reused across different application domains. This automation provides flexibility in service deployment and decommissioning, and reduces deployment time from months and weeks to minutes and seconds. The knowledge base comprised a specification of the domain used for service requirement formulation, predefined service definitions, the metadata of the service functional components, and the descriptions of the microservices that had been used to implement those components.



Prototype implementation – transportation planning

The automation of a transportation planning process is an excellent example of how our cognitive automation framework can be used to create an intelligent DSS. In this example, the DSS is used to automatically determine how to transport passengers or cargo at minimal cost (taking into account the distance traveled, the time needed to transport each passenger and piece of cargo, or the number of buses required for transportation) using connected vehicles (buses or trucks respectively). The answer for a particular task could be a plan that consists of a sequence of steps for the system to perform to reach the goal state. If the system determines that the task cannot be performed, the answer from the framework-based DSS could be the reason why, as well as a possible solution such as increasing the number of vehicles. The motivation behind using the framework for this particular use case was to demonstrate the decrease in the cost of design by building and reusing cross-domain knowledge specifically between people and cargo logistics.

Figure 3 provides an overview of the partially implemented prototype implementation of our framework-based DSS for transportation planning. (Work is still ongoing to implement missing components connecting the framework to the environment.) The knowledge base contains model transformation rules for PDDL and states and transition models that are based on a set of ontologies containing information for the particular transportation domain. The ontologies are organized in multiple layers of abstraction: one

Figure 3
Overview of partially implemented framework-based DSS for transportation planning

DOMAIN KNOWLEDGE	DESCRIPTION
Route	Specification of route(s) as graph(s), which includes vertices, edges and edge-traversal costs (such as travel time and fuel spent).
Transport agents	Number of vehicles, vehicle IDs and vehicle capacity.
Transportable entities	Number and ID of transportable entities (passengers, cargo).
Initial state	The starting location of the transport agents and transportable entities.
Transitions	The transitions that transport agents can perform. Current prototype implementation contains three transitions: pickup, drop, and move-to-next-coordinate.
Goal state	Which criteria need to remain constant for the transport service to complete the specified route (usually this means that all transportable entities are picked up and dropped off at specific points along the route).

Figure 4
Knowledge for transport plan generation

which is cross-domain (the cyber-physical systems ontology in Figure 3); and one which is specific to intelligent transportation systems (ITSs), used for reasoning about transportation problems.

This implementation assumes that the inference engine has already deduced that a planner should be used to solve the transportation planning problem using meta-reasoning expertise (from the knowledge base) and a user-specified goal. The other implemented component is the PDDL generator, which – given the PDDL transformation rules, states and transition files as input – generates files that are understandable for a PDDL planner. The source code of the implementation is available online [3].

As always, there is a direct relationship between the amount of knowledge to be formalized and the effort required to formalize it. Put simply, this means that the more non-formalized knowledge there is, the higher the operational costs will be in terms of time, human resource allocation and money. In the case of a transport schedule generation process, the benefit of using the framework to lower the cost of design is that it reduces the effort required to formalize the knowledge needed for the system to be automated.

Figure 4 shows the different types of knowledge used to generate a transportation plan for two use cases: transportation of passengers in buses and transportation of cargo in trucks. The transport network is viewed as a graph, with points of interest (POIs) as vertices and POI-connecting roads as edges. The "transportable entities" can be passengers or cargo, depending on the use case. Transitions are similar regardless of the route, number and type of transport agents (buses or trucks) and transportable entities. This means that reusing transitions across different transport planning use cases, and storing them as part of the "transition library" (see Figure 3), can reduce the cost of design.

> THE FRAMEWORK'S SELF-ADAPTATION IS SUPPORTED BY META-REASONING AND CONTINUOUS REINFORCEMENT OF THE INTERNAL KNOWLEDGE OVER TIME

We use a simple metric called the reusability index to measure reductions in the cost of design. The reusability index is the ratio of reused entities versus total entities in the knowledge input available to the PDDL generator to generate the plan. The ratio was 0.364 for the bus case and 0.251 for the truck use case. This means that out of the total number of entities created for each case, 36.4 percent were already available in the library for the bus case and 25.1 percent for the truck case. The contrast between the two can mainly be attributed to the difference of the route specification entities, as agents on both routes followed different paths. As the knowledge base becomes more populated, the reusability index will increase, which will in turn lead to a decrease in the development time of further custom ITS solutions. Our measurements also indicate that the reusability index can rise if the route specification is part of the PDDL generator, which can automatically generate the specification using data from a mapping service in conjunction with a routing library. The knowledge engineer could then only specify the desired waypoints (bus stops, for example), and the routes and graphs would be generated automatically by the PDDL generator.

Conclusion

IoT-based systems require a large number of decisions to be made in a short time, which in turn requires automation – not only in terms of infrastructure management, but also within the logic of the IoT applications themselves. A DSS is an essential tool in this context, owing to its ability to enhance human decision-making processes with MI based on behavioral models and information contained in the relevant data streams.

Any system that requires automation in the decision-support process could be enhanced by our cognitive automation framework. It has a domain-agnostic, fixed architecture in which the only variable components are the knowledge base and the set of reasoning methods. By separating cross-domain knowledge from use-case-specific knowledge, the framework makes it possible to maximize reuse, enabling substantial savings in terms of design cost, and shortening time to market for custom-developed solutions. The set of reasoning methods is extensible and can be populated with the methods and tools that are best suited to specific tasks. The framework's self-adaptation is supported by meta-reasoning and continuous reinforcement of the internal knowledge over time. In the future, we plan to extend the framework with analysis of hypothetical situations and an intuitive user interface for both experts and less experienced users. ✱

Terms and abbreviations

API – application programming interface | **CPS** – cyber-physical system | **DSS** – decision support system | **ETL** – extract, transform, load | **IoT** – Internet of Things | **ITS** – intelligent transportation system | **MI** – machine intelligence | **OWL** – Web Ontology Language | **PDDL** – Planning Domain Definition Language | **POI** – point of interest | **RDF** – Resource Description Framework | **TAaaS** – Test Automation as a Service | **XaaS** – Everything as a Service

THE AUTHORS

Aneta Vulgarakis Feljan

◆ has been a senior researcher in the area of MI at Ericsson Research in Sweden since 2014. Her Ph.D. in computer science

from Mälardalen University in Västerås, Sweden, focused on component-based modeling and formal analysis of real-time embedded systems. Feljan has coauthored more than 30 refereed publications on software engineering topics, and served as an organizer and reviewer of many journals, conferences and workshops.

Athanasios Karapantelakis

◆ joined Ericsson in 2007 and currently works as a senior research engineer in the area of MI at Ericsson Research in Sweden. He holds an M.Sc. and Licentiate of Engineering in communication systems from KTH Royal Institute of Technology in Stockholm, Sweden. His background is in software engineering.

Leonid Mokrushin

◆ is a senior researcher in the area of MI at Ericsson Research in Sweden. His current focus is on creating and prototyping innovative concepts for telco and industrial use cases. He joined Ericsson in 2007 after starting

postgraduate studies at Uppsala University focused on the modeling and analysis of real-time systems. He holds an M.Sc. in software engineering from St. Petersburg State Polytechnical University in Russia.

Rafia Inam

◆ has been a researcher at Ericsson Research in Sweden since 2015. Her research interests include 5G cellular networks, service modeling and virtualization of resources, reusability of real-time software and ITS. She received her Ph.D. from Mälardalen University in Västerås, Sweden, in 2014. Her paper, Towards automated service-oriented

lifecycle management for 5G networks, won her the best paper award at the IEEE's 9th International Workshop on Service Oriented Cyber-Physical Systems in Converging Networked Environments (SOCNE) in 2015.

Elena Fersman

◆ is a research leader in the area of MI at Ericsson Research and an adjunct professor in CPSs at the KTH Royal Institute of Technology in Stockholm, Sweden. She received a Ph.D. in computer science from Uppsala University, Sweden, in 2003. Her current research interests are in the areas of modeling, analysis and management of software-intensive intelligent systems applied to 5G and industry and society.

THE AUTHORS

Carlos R. B. Azevedo

◆ joined Ericsson Research's Brazilian team in 2015. He is a researcher in the area of MI whose work is devoted to supporting and automating sequential decision processes by anticipating and resolving conflicts. He holds a Ph.D. in electrical engineering from the University of Campinas in Brazil, and his doctoral thesis was awarded the Brazilian Ministry of Education's 2014 Thesis National Prize in Computation and Automation Engineering.

Klaus Raizer

◆ is a researcher in the area of MI at Ericsson Research in Brazil. He joined the company in 2015. Raizer holds a Ph.D. in electrical and computer engineering from the University of Campinas in Brazil, where he is a founding member of the IEEE Computational Intelligence Chapter. His research interests include computational intelligence, machine learning, cognitive architectures, CPSs, robotics and automation.

Ricardo S. Souza

◆ joined Ericsson Research in Brazil in 2016, where he is a researcher in the area of MI. He holds an M.Sc. in electrical and computer engineering from the University of Campinas in Brazil, and he is currently finalizing his Ph.D. at

the same institution. His research interests include distributed systems, networked and cloud robotics, shared control and computational intelligence.

Further reading

» **KMARF: A Framework for Knowledge Management and Automated Reasoning presented at a Development aspects of Intelligent Adaptive Systems (DIAS) workshop (February 2017), available at:** *https://arxiv.org/abs/1701.03000*

» **The Networked Society will not work without an automation framework, Ericsson Networked Society Blog (July 2015), available at:** *https://www.ericsson.com/thinkingahead/the-networked-society-blog/2015/07/09/the-networked-society-will-not-work-without-an-automation-framework/*

References

1. **Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider and Sebastian Rudolph (December 2012) OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, available at:** *https://www.w3.org/TR/owl2-primer*

2. **Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld and David Wilkins (1998) PDDL – The Planning Domain Definition Language. Tech Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, available at:** *http://homepages.inf.ed.ac.uk/mfourman/tools/propplan/pddl.pdf*

3. **PDDL Generator for Transportation Logistics Scenarios, based on CPS and ITS ontologies, available at:** *https://github.com/SSCIPaperSubmitter/ssciPDDLPlanner*

ISSN 0014-0171
284 23-3299 | Uen

© Ericsson AB 2017 Ericsson
SE-164 83 Stockholm, Sweden
Phone: +46 10 719 0000