Editor: **Amit Sheth,** Kno.e.sis—the Ohio Center of Excellence in Knowledge-Enabled Computing, amit@knoesis.org

Toward a Machine Intelligence Layer for Diverse Industrial IoT Use Cases

Jan Höller and Vlasios Tsiatsis, *Ericsson*
Catherine Mulligan, *Imperial College London*

The Internet of Things (IoT) has moved beyond the hype, and today we see promising applications materializing and industries transforming through well-known digitalization as well as servitization—that is, delivering a service as an integral part of a product. This is evident in the increasing number of physical industry assets represented and manipulated in both the digital and physical worlds and in the fact that the business models for physical and digital assets are converging toward service as opposed to product sales.

IoT can be used in many industry sectors with numerous benefits. Cost optimization and environmental efficiency are just two factors driving this expansion. Examples of IoT applications include predictive maintenance and condition-based monitoring, which are mainly used in industrial settings. However, the envisioned IoT applications are so diverse and include such a broad spectrum of technologies that system designers need design support tools and guidelines. This article provides a few of these design guidelines in the form of architecture and design patterns to enable scalable and replicable solutions rather than point solutions stemming from point problems. As a result, we attempt to structure both the problem (use case space) and the solution domain (architecture).

IoT involves the instrumentation of physical world assets or infrastructures (collectively called an *entity of interest*) with sensors, actuators, and identification devices to enable monitoring and control of these entities. The main high-level building blocks are devices, connectivity, and distributed machine intelligence. Although there is no formal definition of machine intelligence, for the purposes of this article we define it as the combination of machine learning and artificial intelligence technologies, which includes data analytics, symbolic reasoning, and action planning. It is expected that the value created by IoT lies mainly in the services provided by machine intelligence rather than devices and connectivity services. As a result, this article focuses on sharing our experience with the analysis of several IoT use cases and a machine intelligence framework that combines knowledge of solution design for them.

Approach

The main goal of building replicable solutions can be likened to the goal of formulating a reference architecture that encompasses and encodes the knowledge of numerous solution architectures. In turn, according to Nick Rozanski and Eóin Woods, a solution architecture can be described as a set of architecture views or blueprints, each addressing the concerns of a specific stakeholder.[1]

In generating a reference architecture one typically follows the design process for a single stakeholder concern that is typically expressed with one or more use cases. The process is then repeated for all possible stakeholder concerns. In the end, all solution architectures are combined in a union.

In this article, we follow the reference architecture process for a subset of stakeholders (end users); therefore, the union of different solution architectures is a partial version of a reference architecture called the *machine intelligence*

 Published by the IEEE Computer Society

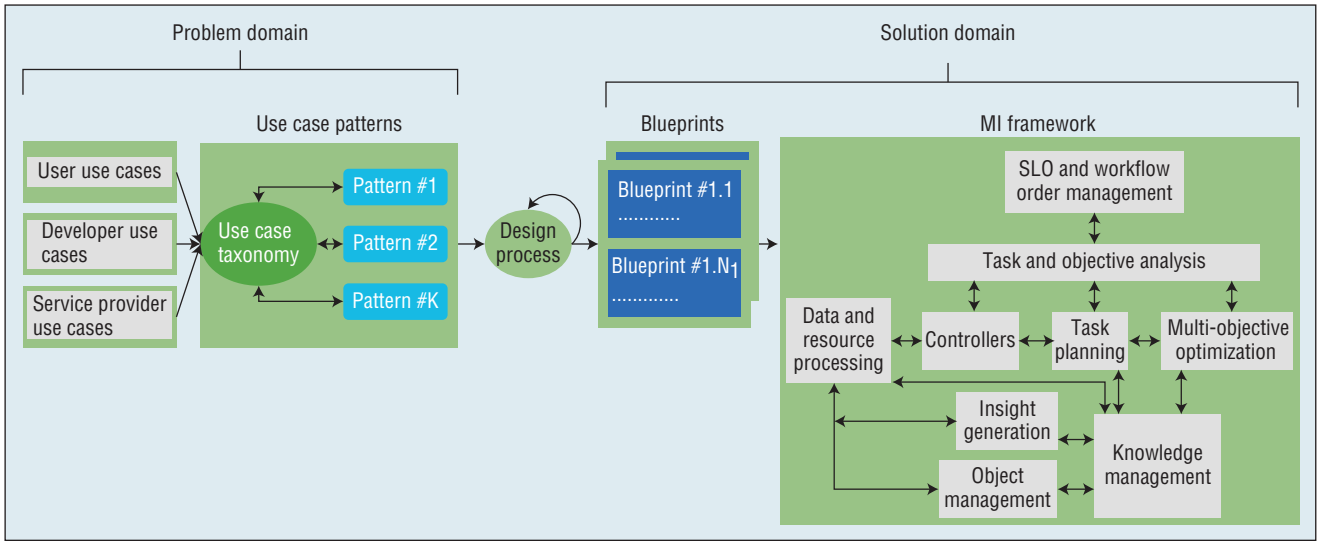

Figure 1. Structuring the problem and solutions domains (MI: machine intelligence, SLO: service-level objective).

framework. Figure 1 illustrates this process, referred to as *structuring the solution domain*. Other types of stakeholders are developers and service providers. The resulting machine intelligence framework with a different stakeholder will be somewhat different from that of an end user. However, following the same methodology outlined here, frameworks for other stakeholders can be generated.

Because of the huge number of available use cases, with new ones arising all the time, following the design process to generate a single solution architecture is not tractable. Therefore, we also structure the problem domain or the potential use cases. For the purposes of this exercise, we limited the studied use case sources to oneM2M,[2] the Industrial Internet Consortium (IIC) use cases and testbeds (www.iiconsortium.org), and National Institute of Standards and Technology big data.[3] In the context of this article, structuring the problem domain (Figure 1) means that individual uses cases are grouped into use case families or patterns. We have used the use cases' structure to limit the number of times we applied the design process for developing solution blueprints. As a result, instead of generating solution blueprints from each specific use case, we generated architecture blueprints for a representative use case from a family of use cases or a use case pattern. We then iterated the design process over all the identified use case patterns.

Structuring the Problem Domain: Use Case Patterns

We have studied about 100 use cases from different sources, which have varying degrees of detail and are classified or grouped according to market-related terms. The most typical of these are the (market) sector and (market) vertical that a use case belongs to. Although not well-defined in the literature, we have followed the IIC definition and taxonomy.[4] According to the IIC, a sector (such as healthcare) is a logical group of related verticals (for example, hospitals), and a vertical is a market in which vendors offer goods and services that meet a particular set of usage, technical, or regulatory requirements.

We focus our study on the end user as the main system stakeholder as stated earlier. The end user formulates concerns that are concretely described in use cases. These are grouped logically into (use case) applications apart from their market classification into sector and vertical. Our assumption is that a sector contains multiple verticals, each containing multiple applications and multiple use case instances of different use case types. We refer to a use case instance as simply a "use case" (see Figure 2). In turn, each use case expresses a stakeholder concern requiring a desired (technical) system functionality or set of characteristics.

Although this classification is done mainly from a market perspective, we aimed to reshuffle the same set of use cases into another set of groups that emphasize the technical characteristics. We call these groups (technical) *use case patterns*. We also identified a set of technical characteristics based on which the different (technical) use case patterns can be described; however, for brevity we omit the technical characteristics from the pattern definitions.

Through our research, we have identified seven use case patterns.

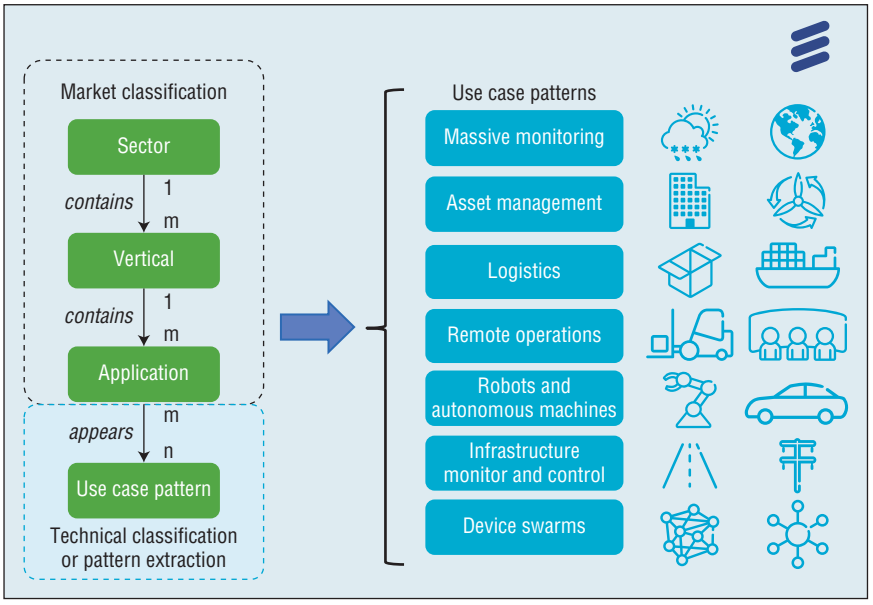

Figure 2. Structuring Internet of Things sectors, verticals, applications, and use cases toward recurring use case patterns.

Massive monitoring. This use case pattern involves numerous sensors deployed across a large geographical area. Data is collected over a period of time for bulk batch or stream analysis. Data analysis aims to find trends, detect anomalies or abnormal situations, or simply learn the behavior of the monitored asset or phenomena. Typical examples include environmental and climate monitoring and pollution monitoring.

Asset management. This pattern involves managing physical assets that are well-defined and confined. Example assets include a building, vehicle, piece of industrial machinery, turbine, or human patient. Typical management aspects are to optimize the asset's operation, perform diagnostics, or do predictive maintenance.

Logistics. A typical logistics scenario can be described as the process of coordination, management, and orchestration of a collection of tasks in a workflow to achieve a set of goals (such as time or cost optimization) by making efficient use of the available resources. Typical examples include fleet management, supply-chain optimization, and pickup-delivery services.

Remote operations. Remote operation generally refers to the control and operation of a system or equipment from a remote location either by humans or software. In this case, the remotely operated system or equipment cannot or is not designed to operate completely autonomously to accomplish the task. Typical examples include remote mining, remote-controlled vehicles and drones, and remote surgery.

Robots and autonomous machines. This use case pattern covers the operations and management of partially or fully autonomous systems such as robots, vehicles, and drones. Such systems are often described as cyberphysical systems (CPS), which integrate the dynamics of the physical processes with those of the software and networking. Typical examples include static or mobile factory floor robots and autonomous vehicles.

Infrastructure monitor and control. This use case pattern covers management of large-scale industrial or extended infrastructures that need to be monitored and controlled. Examples include a transportation infrastructure such as a national road network, a utility infrastructure such as the electric grid, oil and gas pipelines, and street lighting.

Device swarms. This use case pattern covers devices and systems that operate autonomously with a simple set of rules and no central intelligence, and form peer-to-peer groups to collectively reach a common goal. Typical examples include microgrid producers and consumers and zero-trust computing applications such as home automation.

Structuring the Solution Domain

The solution domain consists mainly of architectural blueprints that include a few main types of system components. This domain encompasses devices, connectivity, cloud and distributed computing, machine intelligence, and mechanisms for visualizing information and integrating applications to an enterprise environment. These blueprints typically concretely express the functional components of a resulting solution. However, a system also typically consists of a set of nonfunctional characteristics. Through our research we have identified a core set of functional and nonfunctional characteristics that can be grouped into different perspectives: data and information perspectives (multimodality of IoT data, the need for insight and analysis-driven functions, knowledge representation, cognition, and so on); control perspectives, meaning whether a control functionality exists (sensing and actuation in feedback loops, workflow/

process-driven); and general characteristics (locality, timing criticality, safety, and security).

A Framework for Distributed Machine Intelligence for Industrial Use Cases

Our approach is to aggregate solution blueprints for the identified recurring use case patterns, thus arriving at our desired framework. As mentioned, our focus in this article is distributed machine intelligence functionality supporting the diversity of IoT use cases we have identified.

For the sake of brevity, we have left out two important, but for the objective of this article, secondary considerations. The first is the general need for distributed processing of machine intelligence logic, and the second is lifecycle management of the solution.

IoT data processing and decision making is generally a highly distributed capability. The need for distributed processing in IoT comes from different requirements.[5] Data volumes, cost, performance and latency, autonomous local asset operation, robustness, and safety around IoT asset operation are the main requirements. IoT distributed processing extends beyond the datacenter to constrained IoT devices, and the resulting heterogeneity needs to be managed to meet service-level agreements for the applications.

Lifecycle management includes operational aspects such as the type of logic to deploy and location of the deployment, ensuring necessary robustness, and trust and security. In addition, the system should be adaptive and cognitive to handle changing external requirements or changing contexts. For instance, the system should ideally detect the need to change controller parameters that might be wrongly set or continuously do model training.

Machine Intelligence Functional Domains

The machine intelligence framework provides the functionality needed to realize the use case applications relevant to end users. As such, it processes IoT data from various sources, derives and executes control operations to manipulate the asset or infrastructure via actuators, and maintains a cognitive knowledge base related to the assets. An objective of the framework is to partition functionality into application-independent building blocks. These can then be interconnected to realize the use case applications according to a service-oriented paradigm, lending themselves to microservices implementations.[6]

> The machine intelligence framework provides the functionality needed to realize the use case applications relevant to end users.

The main functional domains, as shown in Figure 3, include the capability to manage data and relevant IoT resources (sensing, actuation, and identification) and process data and extract information (analytics or machine learning); various types of control and execution (controllers or planning tools); and capabilities to manage and represent knowledge relating to the assets and the system. Figure 3 shows the functional domains as well as some of the main interfaces between them.

Data and Resource Processing

By data and resources we mean sensor data, actuator services, and their representations. Individual sensor data and actuator control is the raw fabric for interacting with the physical world. Sensor data includes individual data items and events and datastreams from a single sensor. Resources are abstractions of sensors and actuators in the system.

Massive-scale IoT deployments require some key considerations in the collection phase. Data can be received as an event stream with varying speed, volume, and dynamicity over time. Information creation, attribute validation, and verification are necessary steps in the data collection. Data can be received asynchronously or synchronously, depending on the type of application at hand.

Data management, curation, and resource management are crucial in IoT systems and comprise the following steps. First, data is collected and distributed. Then, data and resources are modeled to capture heterogeneity (structured, unstructured), high distribution, large size (number of data sources, streams, and actuator end points), and semantic annotation describing meaning of the endpoint capabilities. Resources need the proper annotation, describing such attributes as meaning, origin, and quality. Also, transmitted data needs to be filtered to the application needs. IoT data, when needed, will require distributed storage, taking into account factors such as cost and storage capacity.

Real resources require appropriate abstract representations and management in an IoT system, for example, as a representation of a datastream or as an aggregation of sensor data. Resource abstraction implies a level

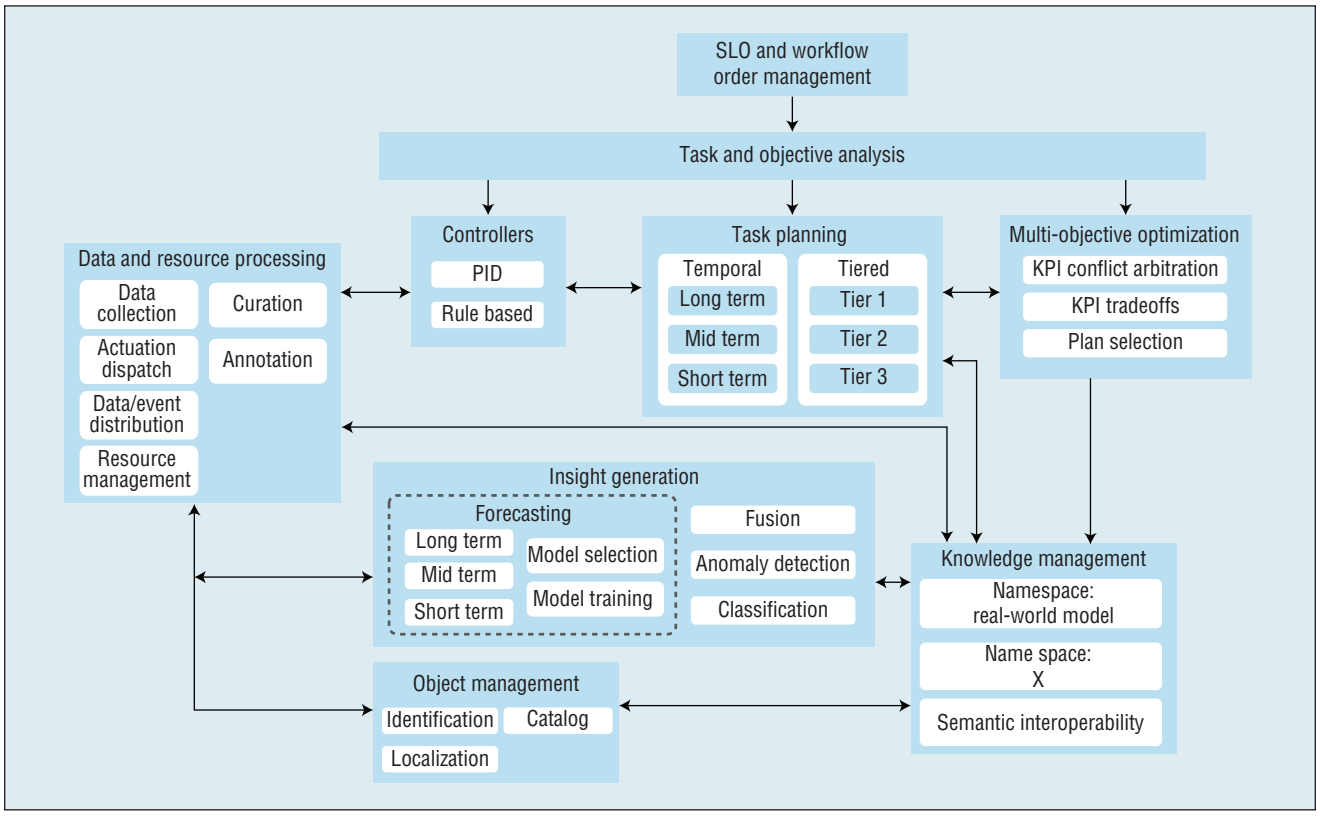

Figure 3. Framework for machine intelligence supporting a diversity of IoT use cases (KPI: key performance indicator, PID: proportional, integral, and derivative; SLO: service-level objective).

of indirection requiring a resolution function that dynamically maps between the resource representation and the real resource.

Insight Generation

Forecasting is a key issue in the prominent IoT use case of predictive maintenance, which is used to determine the health of a piece of machinery and understand when any maintenance might be needed.

Forecasting involves predicting new outcomes based on previously known results. Depending on the IoT use case, different forecasting timeframes apply. For example, trajectory forecasting of moving objects can be real time, whereas machine degradation is more long term. Forecasting can be data driven or model driven depending on the problem requirements. Model training is a necessity and can be based on training sets or via reinforcement learning. Typical forecasting models can be statistical or neural networks-based, Bayesian or non-Bayesian, linear or nonlinear, parametric or nonparametric, univariate or multivariate.

Sensor fusion is another technique. In general, fusion concerns combining data and information from diverse sources so that the resulting information is more accurate than if one had relied on a single source. An example is the localization of an object that can rely on a combination of ultra-wide band (UWB) transponders, camera detection, and contextual information sensed by the object itself, and when fused provide a much higher degree of location accuracy.

Knowledge Management

Knowledge management involves representing, modeling, structuring, and sharing knowledge about a physical asset or infrastructure. Knowledge is the collected set of data, inferred knowledge and insights, and control capabilities of the asset. The knowledge can further be structured so the different data, insights, and control capabilities can be directly mapped to the asset's real-world structure, thus becoming a proper digital representation of the asset.

Knowledge is generally of two types: declarative knowledge (also referred to as propositional knowledge) and procedural knowledge (also referred to as imperative knowledge). Declarative knowledge describes what an entity is and how it is structured and formally expressed using ontologies. Procedural knowledge describes how an entity behaves, for example, in response to stimuli; and the formal description format is typically via state machines.

Knowledge is captured and made available in what can be referred to as a knowledge base, which is manifested by a set of ontologies. Typical ontologies in IoT are not only for the actual real-world model of the asset and expert knowledge but also knowledge about system and application objectives, such as key performance indicators (KPIs), a work order, task plans, and constraints of the IoT system itself. The real-world model is typically a hierarchical or graph structure.

Across domains in IoT, semantic interoperability is essential for achieving many business applications.[7] Semantic interoperability enables data and information to be shared across domains and understood by systems without needing manual interpretations on top of technical details or protocol and syntactic interoperability. Semantic interoperability requires mapping methods that can be predefined or self-learning. The latter requires algorithms that consider structural, terminological, and semantic differences and similarities.

Object Management

Object management involves identifying, localizing, and cataloging physical assets that are handled by the IoT system. This is important for some types of use cases, such as logistics involving transported goods or localization of tools on a factory floor.

Object identification is possible using various techniques, such as tags based on optical or radio technologies (for example, QR codes or RFID tags). The purpose is to uniquely identify and name objects. A resolution infrastructure is usually in place to find information about the object. A prominent example is electronic product code information services (EPCIS).[8]

Object localization needs to be tailored to the IoT needs and deployment scenarios (such as indoor or outdoor environments). Typical indoor localization technologies include video or image processing, Bluetooth beacons, use of Wi-Fi access points, or UWB ranging. Outdoors, GPS-based localization is typically relied on. For any localization solution, the required accuracy, size of area covered, and real timeliness of location must be considered.

Semantic interoperability enables data and information to be shared across domains and understood by systems without needing manual interpretations on top of technical details or protocol and syntactic interoperability.

A catalog function can also be required. This function works as a repository of all assets of interest and includes other properties of the asset. EPCIS is an example.[8]

Controllers

Control is a core automation point in any IoT system involving actuators. Control software commands the assets' desired behavior. Common to all controllers is the deterministic behavior of controlling operations based on input from an a priori desired and defined operational behavior. The use of different controller types is based on functional and nonfunctional characteristics meeting application needs.

Whereas many control systems in robotics and other real-world continuous and industrial systems use proportional, integral, and derivative (PID) controls, other IoT use cases, such as home automation, often use rule-based systems for event-driven control.

A PID controller is a control loop feedback mechanism using a mathematical function that takes the deviation between the desired state and the measure state as input for control. *Proportional* control means that proportional feedback of the deviation is provided to determine the control value. A *derivative* part of the deviation dampens the error. An *integral* part of the deviation provides errors to be removed over time. Examples include inverse kinematics for robot control and temperature control of a fluid system. This requires knowledge about the physical behavior and properties of the asset controlled.

Rule-based controllers are based on a set of predefined rules that are trigger-action pairs, where a trigger is a condition and an action is a predefined workflow typically containing commands to the devices or related services—for example, following the simple logic of "if this, then that."

In sufficiently complex, dynamic, and nondeterministic situations one can enhance the usability and maintainability of both PID and rule-based control systems by making them use task planning technologies to help infer the actions to be taken.

Task Planning

Task planning can be defined as the process of generating a sequence of actions with certain objectives. Planning

can be applied to a variety of problems such as route planning of autonomous vehicles, optimization of a logistics flows, and automation of field personnel.

The planning problem is normally represented by three key elements—states, actions, and goals. State identifies the model of the world, actions represent different operations that affect the system's state, and goals are states to achieve or maintain. Deriving the task plan is to take the current state, the desired state and the possible actions and from that generate a plan as a sequence of possible or proposed actions. A plan can also be a partially ordered list of tasks. One possible way to perform task planning is using AI planners, where the world and the problem are modeled using a planning domain definition language (PDDL).

Multiobjective Optimization

Automating complex system operations by leveraging data-driven strategies designed to analyze alternatives under multiple conflicting views or KPIs is challenging. First, KPI evaluations are not always reliable and might be subject to changes over time; second, the costs incurred in adapting solutions under operation must be accounted for. In such cases, it is difficult to track how the underlying tradeoffs (such as return versus risk or throughput versus cost) will evolve over time, and decision-making preferences are hard to elicit and represent computationally. In the absence of clear preferences and priorities over the KPIs, general problem-solving strategies and architectures must be designed for automating general data-driven multiobjective optimization (MOO) systems under uncertainty.

MOO can play a key role in applications where conflict resolution is expected. For instance, in supply-chain control applications, the proposed system can monitor the profitability for the whole chain as well as the overall product shortage risk. Those two KPIs are clearly in conflict as optimization at an extreme for one results in a risk for the other.

A key difference between task planning and optimization is that in the latter does not assume that desired goal states will be input by the system stakeholders. This stems from the fact that it can be impossible for humans to cope with the underlying complexity of explicitly specifying goal states while simultaneously fulfilling all service-level objectives (SLOs). In such cases, it is possible to leverage simulation-based MOO to automatically explore the space of all candidate goal states that not only fulfill all SLOs but actually surpass them and deliver outstanding performance.

Service Level Objectives and Workflow Management

The end user's interests in the system can be specified as a set of high-level, quantifiable performance metrics by SLOs and workflow orders. SLOs are translated into KPIs, which are deemed critical for verifying service execution and detecting deviations from SLOs. KPIs can further be broken down to needed insights and, together with workflow orders, the intentions or actions of the system. The insights and actions can then be used to define the needed sensor data and actuator controls. For instance, in a logistics use case, a workflow order can request that a number of products be delivered to a certain subset of retailers within a specified deadline to keep shortage risk under the agreed levels.

The KPIs and workflow orders encapsulate information that allows the extraction of inputs to task planning, controllers, and MOO, which also includes the necessary information from the insight generation functional domain. For task planning, the extracted inputs should correspond to goal states that can be used to compute an appropriate plan.

For controllers, workflow orders might specify new set levels of parameters or rules. For multiobjective optimizers, workflow orders should specify a set of KPIs to be balanced by automated tradeoff analysis to comply to overall service objectives as well as mitigating conflicts.

IoT is about the digital representation of the physical world to enable the digitalization and servitization of physical assets or entities of interest. Since the application spread in today's IoT is wide and is typically structured in market-oriented groups, a system designer needs IoT system design patterns to assist in designing for scalable and replicable solutions. The work presented here provides a generic blueprint for designers to jumpstart the design process of an unknown use case. ■

References

1. N. Rozanski and E. Woods, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*, 2nd ed., Addison-Wesley, 2011.
2. ETSI, *oneM2M Use Case Collection*, ETSI TR118501v1.0.0, tech. report, 2015; www.etsi.org/deliver/etsi_tr/118500_118599/118501/01.00.00_60/tr_118501v010000p.pdf.
3. Nat'l Inst. of Standards and Technology, *Big Data Interoperability Framework: Volume 3, Use Cases and General Requirements*, NIST Special Publication 1500-3, 2015; http://dx.doi.org/10.6028/NIST.SP.1500-3.
4. R.A. Martin and A. Soellinger, "The Emerging IIC Verticals Taxonomy Landscape," *IIC J. Innovation*, June 2016;

www.iiconsortium.org/news/joi-articles/2016-June-The-Emerging-IIC-Verticals-Taxonomy-Landscape.pdf.
5. OpenFog Consortium, *OpenFog Reference Architecture for Fog Computing*, tech. report OPFRA001.020817, 2017; www.openfogconsortium.org/ra.
6. J. Lewis and M. Fowler, "Microservices: A Definition of This New Architectural Term," MartinFowler.com, 25 Mar. 2014; https://martinfowler.com/articles/microservices.html.
7. M. Serrano et. al., *IoT Semantic Interoperability: Research Challenges, Best Practices, Recommendations and Next Steps*," European Research Cluster on the Internet of Things (IERC), 2015; www.internet-of-things-research.eu/pdf/IERC_Position_Paper_IoT_Semantic_Interoperability_Final.pdf.
8. *EPC Information Services (EPCIS) Standard*, release 1.2, GS1, 2016; http://www.gs1.org/sites/default/files/docs/epc/EPCIS-Standard-1.2-r-2016-09-29.pdf.

Jan Höller is a research fellow at Ericsson Research, Sweden. His research interests include industrial Internet of Things systems, digital transformation, and machine intelligence in autonomous systems. Höller has an MSc in engineering physics from Lund Institute of Technology. He serves on the Board of Directors of the IP for Smart Objects Alliance. Contact him at jan.holler@ericsson.com.

Vlasios Tsiatsis is a senior researcher at Ericsson Research, Sweden, and an Internet of Things architect. His research interests include IoT, cloud, and analytics security. Tsiatsis has a PhD in electrical engineering from the University of California, Los Angeles. He is a member of IEEE and ACM. Contact him at vlasios.tsiatsis@ericsson.com.

Cathy Mulligan is a research fellow at Imperial College London and vice chair for ETSI Industry Specification Group for Context Information Management. Her research interests include digital technologies and its impact on industrial structures. Mulligan has a PhD in engineering from the University of Cambridge. She is a member of IEEE, IET, and ACM. Contact her at c.mulligan@imperial.ac.uk.

This article originally appeared in IEEE Intelligent Systems, *vol. 32, no. 4, 2017.*