IoT Service Platform Enhancement through 'In-Situ' Machine Learning of Real-World Knowledge

Marc Roelands
Bell Labs
Alcatel-Lucent
Antwerp, Belgium
marc.roelands@alcatel-lucent.com

***Abstract*— With Machine-to-Machine and Internet of Things getting beyond hype, including an ever wider range of connected device types in ever more value-added services, a new era of data (and multimedia) stream-intensive services is emerging. While live data is massively becoming available, turning it into meaningful information that is not only actionable for decision makers, but also can be leveraged as a behavioral service property, or even reused across services, is a challenge that demands a systematic approach. In this paper we propose such systematic approach, towards establishing an Internet of Things service platform architecture that leverages real-world knowledge for faster service creation and more efficient execution. Illustrated by example scenarios, we go further beyond this, proposing a method to systematically leverage machine learning techniques for revising, improving or ultimately semi-automatically extending this real-world knowledge 'in-situ', i.e. during system operation, leveraging real-world observation in-context of requested service execution.**

***Index Terms* — IoT service platform, service creation, machine learning, real world phenomena, cognitive feedback loop**

I. Introduction

With the steady growth of the Machine-to-Machine (M2M) industry and the promise of a trillion dollar market outlook that market analysts derive from the fact that billions of devices could potentially become Internet-connected – the Internet of Things (IoT) –, massive data availability is quickly becoming a reality. Without even considering the promise of further IoT object and device connectivity, already now data analytics are a major emerging trend, the so-called 'Big Data' analytics, applied on enterprise workflow, computing and networking data, and e.g. on social network activity in the public space.

However, beyond the initial wave of Big Data ever more opportunities are identified to process and analyze live information, in near real time as soon as it occurs or is requested. In fact, the scale by which data is becoming available, often opening up data sources on the public Internet, requires more efficient ways to (pre-)process it, as simply storing it for later offline analysis becomes virtually impossible in many cases. This is well illustrated by the platform and tooling evolution that e.g. Google has been developing for its search engine (*MapReduce*, *Dremel* [1], *Pregel*) or similar data processing platform solutions brought forward by the open source community such as *Hadoop* [2], the real-time efficiencies brought by *Twitter*'s *Storm*, e.g. in social network apps [3], and many more tools and platforms evolving beyond non-real-time database technology.

On the data source side, not just temperature, humidity or acceleration sensors are part of the IoT equation. E.g. also actuated video-streaming devices are becoming part of it, ranging from crowd-sourced video footage (with e.g. smartphone video ending up in broadcast TV news as the only just-in-time recording available), over government and private CCTV infrastructures (with e.g. a massive amount of cameras found in cities such as London [4]), up to consumer gadgets like drones (e.g. *Falkor Systems* bringing video tracking to the *Parrot* drone) or little robots (e.g. *romotive.com*).

Yet also, many domain-specific IoT services and platforms are emerging. Sampling the wide spectrum, we just mention smart home automation solutions (*Nest*, *Belkin WeMo*), personal health monitoring (*FitBit*, *NikePlus*, *Withings*), IoT fleet management (e.g. fuel tax management by *Roamworks*), enhanced private car navigation (*Raspberry Pi* aiming at connected cars, *TomTom* enhancing their navigation service with real time traffic data, and others), and somewhat broader oriented platforms such as offered by *ioBridge* or *ThingWorx*.

A new era of **data (and multimedia) stream-intensive services** is thus emerging, justifying the definition of horizontal industry solutions for IoT service platforms on which potential platform operators expose value for many 3rd parties in turn to launch and manage services easily, at a low operation cost. Services e.g. range from context-aware cloud services leveraging massive live and predicted environment data, which users interact with via smartphones and wearables, up to personalized broadcast services processing many, often ad-hoc video streams for fine-grained subject coverage.

From this onset, we address the general architectural options and requirements for an IoT service platform in chapter II of this paper. Within that frame, we discuss the enhancement that the modeling of real-world knowledge (RWK) can bring for such a horizontal service platform in chapter III, pointing at the potential for execution resource use optimization and a first way of positioning 'in-situ' machine learning in the architecture. Beyond that, we introduce a systematic approach to have a cognitive feedback loop for 'in-situ' validation of knowledge hypotheses in chapter IV, concluding the paper with a perspective on the ultimate value of such new approaches.

978-1-4799-0540-9/13/$31.00 ©2013 IEEE

II. Design Options and Requirements for IoT Service Creation, Deployment and Execution

In the past decennia, the telecom industry has traditionally been looking into standardizing, or in some cases at least 'blueprinting', horizontal service platforms for obtaining economies of scale across a supported range of (3rd party) services. Ranging from the early Intelligent Networks (IN) service programmability, over the Service-Oriented Architecture (SOA) approach of Service Delivery Platforms (SDP) on top of the IP-based Multimedia Subsystem (IMS) of 3G mobile (and later also fixed-mobile converged) telecom networks, the vision of a horizontal service platform, distinguishing a **service creation environment (SCE)** and an **execution environment (EE)** in the architecture, is now also being applied to M2M. Once more in the history of telecom, potential platform operators seek to expose a value-added platform for many 3rd parties to launch and manage services easily, at a low operation cost for all parties involved. Leading standardization bodies such as ETSI [5] and the OneM2M initiative [6], bringing multiple bodies together around the M2M theme, are now adding semantic technology and other horizontal service layer aspects on the standardization agenda [7][6], thus addressing – and going beyond – device abstraction and virtualization. The M2M context indeed provides an illustrative subset of services that operate on distributed data stream sources (and sinks), which implies a potential for a distributed service execution model. Many related service platform architectural aspects are also addressed by the European Commission's 7th Framework Program, with e.g. *FP7 IoT-A* building an architecture reference framework for IoT [8], and the Future Internet Flagship project *FI-Ware* [9] and e.g. *FP7 SENSEI* [10] addressing intelligent enabling functions in such horizontal service platform context.

Motivated by the known service platform paradigms and the specific aspects that M2M and IoT add to it, Fig. 1 provides a high-level view of the IoT service platform architecture thus taken as a working basis for the research. The figure depicts the three distinguished working planes of the architecture: a *Data Plane*, an *Information Plane*, and a *Knowledge Plane* (taking the DIKW pyramid [11] as a naming convention).

The ***Data Plane*** is concerned with the virtualization of all data sources and sinks connected to the platform, so that they can be selected, for service use, based on semantic (as well as cost) properties without further need to consider the virtualized sensors' and actuators' technical internals related to addressing, connection maintenance, and data transfer and buffering using either streaming, or REST or RPC-style protocols. Connected device instances are introduced in the platform by an *Installing Actor*, an actor role e.g. taken up by a physical person installing the physical device in a building, or by an end-user registering a personal mobile device to the platform. *Device Vendors*, another actor role (not shown in the picture), are well positioned to stipulate the exact technical constructs needed for connecting their device types to the IoT service platform. They can provide type information and specific device driver code as part of a *Virtual Device Template* for the platform, thus automating to a great extent the installation process to be followed by any installing actor. As such, the platform can become aware of a range of *Real World Entities*, which services running on the platform can observe and/or actuate. (Installation of devices can occur a priori, before a service is launched, e.g. as a structured installation, or as part of the software flow of service launch, installing it when a dedicated need occurs.)

The ***Information Plane*** in the architecture is associated with the (potentially distributed) execution environment in which actual IoT services are to be executed. An IoT service (and, by generalization, any service type on the platform) is considered to be represented platform-internally as a graph of processing nodes interconnected by streaming pipes. Again here, the processing nodes, containing particular processing logic, are described in templates, *Processing Node Templates*, easing the automatic deployment of such components when they are needed for graph execution. While exposing their actual logic as a component, e.g. as a function signature with semantic typing of inputs and results, the processing node templates hide their implementation, and so can exist in multiple interchangeable versions with different implementations. The resource-optimal choice for either implementation of a given processing node type, each of which may e.g. correspond to another execution container type available in the platform, can thus be made independently of the actual functionality requested to be executed. In this way, the information plane thus is concerned with the resource management for executing a collection of services, processing live information from data sources, turning it in more meaningful information, and eventually sinking actuation information to controllable devices.

The ***Knowledge Plane*** of the architecture exposed the actual service creation interface, in which the provisioning of *Service Templates* and the actual *Service Requests* are distinguished. While the service templates describe *how* particular service goals can be obtained, in terms of a logical graph composition of underlying processing node templates, the service request provides the actual external command to

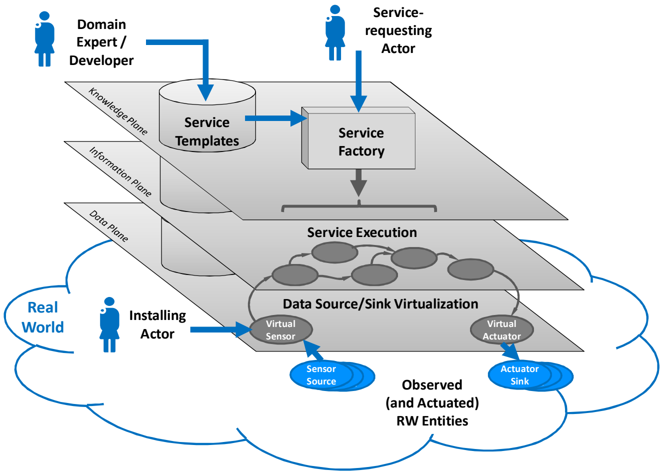

Figure 1. Base IoT service platform architecture leveraging template-based service creation.

target such a *goal*. In this way, a *Domain Expert* and/or *Developer* can play an ecosystem role serving many *Service-requesting Actors*, which in turn can focus on their individual business objectives as platform users. When a service requests is issued, a service factory function matches it with one or more service templates, producing a full service specification to be passed to the information plane for execution. Fig. 1 thus denotes that goal-oriented service requests get translated on-the-fly, by means of template instantiation, into a service instance to be executed as a processing node graph.

As discussed, the **template hierarchy** has been chosen to be organized according to the three platform planes, following the respective concerns addressed by those planes, thus distinguishing:

- service templates: a logical, abstract level of templates capable of breaking down high-level, real-world service goal expressions into the logical processing steps needed, as a real-world status and behavior transformation, e.g. a template that stipulates how to combine several video analysis algorithms into a desired video output,
- processing node templates: a library of technical templates implementing, application domain-independently, the more fine-grained information processing steps, in terms of the actual algorithms needed and in terms of the platform's available execution capabilities, e.g. implementations of a video analysis, modeling or signal processing algorithm, and
- virtual device templates: a set of templates capturing the connectivity and control details for each supported device type.

While the provisioning of a range of templates comes at a cost and may impose some domain limitations implied by that, **software reuse** is promoted in such a service platform that supports ranges of services that can be created and **deployed fast**, and executed in an **efficient yet controllable** way, through service requests easily expressible by many parties.

From this basis, we propose several platform enhancements in the next sections, as a solution to systematically leveraging knowledge about the real world, i.e. the environment in which the system needs to effectively provide value to the stakeholders. We recap use case examples along the way, to illustrate the introduced concepts further.

III. Leveraging Real-World Knowledge for smart, efficient services

A clear first element in order to be able to express a service goal in a real world setting, is by using the defined vocabulary of a domain ontology, e.g. implemented in RDF or OWL [12][13] to make the goal expression. Decomposition of the service goal into more fine-grained steps, as done by means of the service templates, is another needed element. In fact, as a generalization to classical service templates, one could see those templates as a set of rules that fire when a service request is issued, to logically become part of a **Real-World Knowledge Model** (RWK Model) maintained by the system. This can be implemented, for example, as SPARQL query and rule mechanisms [14].

A. Explicitly Capturing Real-World Knowledge, as opposed to System Knowledge

Domain ontologies, service templates and further knowledge inference from them as triggered by service requests are thus a means to capture RWK in a system, in the course of services being defined in and requested to be performed by the system.

An important system design decision that is made explicit by formally distinguishing a stored RWK Model in the service platform architecture is the separation of concerns between modeling of the real world, on the one hand, and the modeling of system-operational resource constraints and limitations on the other. Capturing the rules that govern the actual execution capabilities of the system could be denoted as the capturing of **System Knowledge** (SK). Where RWK will allow the system to be aware of the behavioral rules that govern the world outside the system (and to grasp the meaning of service goals and what real-world effects they resort in), the maintenance of SK allows the system to be aware of its own working (and how service execution can be fulfilled through its computing capabilities). When, as a system design decision, the RWK-SK functional split is done rigorously, machine reasoning can be performed on each knowledge domain independently, and so more optimally targeted. Considering some examples helps imagining that this makes sense, as different concepts and behavior is expressed in each knowledge domain.

As examples of RWK, we could e.g. have *"physical objects follow a trajectory that is continuous in space and time"*, complemented with an ontology of physical object types that allows applying this rule to many types of objects being the subject of a service request, or more general laws of nature such as "*speed of an object equals the derivative of position over time*" or still many more scientific facts beyond this. Next to such generic real-world rules, also many types of application domain-specific knowledge could be considered for inclusion in a system's RWK Model, ranging from very basic rules up to more complex knowledge such as *"for any given street map A, when traffic density is high at crossroad a, the likelihood of a traffic jam occurring at roads with topological relation r to a can be estimated by the formula X"* for the domain of traffic management.

As examples of SK, we can e.g. think of knowledge in the area of operational cost considerations, such as "*On a platform of type P, deployment and execution of algorithms of type A have timing and resource occupation characteristics x, y and z*", which could be the basis for the system to autonomously decide on execution container type and placement optimization strategies.

The differences in the nature of RWK and SK can furthermore be seen to be articulated from the system lifecycle perspective. **Learning or 'growing' RWK and SK during system operation**, i.e. extending it beyond what is a priori known, has different constraints for both knowledge domains. As SK is exclusively confined to the system itself, we can consider the system to be self-learning *autonomously* on how to objectively improve its operations. This is the approach generally taken in autonomous computing with so-called *Self-X* technology [15]. In contrast, interaction with the outside world, as a peer intelligent entity, is an intrinsic need for learning additional RWK. This then is rather similar to a robot that fulfills service requests, a research field where machine learning from real-world experience is also explored [16][17]. Furthermore, as SK is exclusively confined to the system itself, SK is to a large extent *a priori* knowledge that can be integrated *implicitly* in the architecture design, while ***a posteriori* knowledge needs to be captured by explicit interfaces during system operation**. Especially when the system is designed to be horizontal, i.e. agnostic of specific service instances or application domains, RWK is typically mostly of such nature and its full range can impossibly be anticipated exhaustively, even not when confined to a particular application domain.

The discussed consideration have lead us to explore whether real-world-aware system behavior obtained by a controlled combination of RWK-concluded 'constraints' with a service platform's (SK-based) self-management capabilities, can lead to a more expressive, yet also more efficient service platform architecture, as compared to a 'flat' architecture.

B. Real-World Knowledge versus Real World Information

Another distinction assumed by the concepts discussed further in this paper is the distinction made between RWK and Real World *information* (RW information). In the IoT service platform context, one can see RW information as the meaningfully transformed data read, observed or sensed in the outside world, especially also as live streams, related to actual *instances existing* in the real world. When taking that convention, RWK pertains more to the **rules according to which the world will behave**. Put more formally, RWK is the higher order logic expressed over the propositional logical facts expressed by RW information [18][19]. RW information is what is captured and flows through the system at service execution time, in (near) real-time, observing and actuating the real world with proper system responsiveness, while RWK concerns a meta-expression and meta-observation about this on which reasoning and learning only is required to happen at a slower, more 'human-like' pace. Taking this as a design consideration, allows considering technologies such as Complex Event Processing (CEP) [20] for the massive RW information processing, while leveraging semantics-oriented technology [21][22] as a basis for less time-critical RWK capturing and reasoning. From the service lifecycle perspective, RWK handling can be done before the actual service execution, as a planning or 'design-time preparation' step, to determine **RWK dependencies and constraints** to be leveraged during the consecutive execution step.

With the distinctions between RWK versus SK, and RWK versus RW information, the system design choice taken thus is to have RWK handling and learning, as well as interpreting incoming service requests, as the main focus of the knowledge plane of the architecture, while RW information processing, in the context of the executing services, is kept purely in focus of the information plane. (SK handling, as related to the specifically available execution capabilities of a particular platform, is considered a 'built-in' information plane property.)

C. Resource Optimisation based on Real-World Knowledge

Beyond and combining with SK-based self-optimization, we propose a first step into making the service platform architecture 'smarter' based on RWK. As indicated as red-colored elements in Fig. 2, the modeled RWK as relevant to the context of a particular service request can be leveraged to determine run-time constraint predictions as part of the service description to be passed to the information plane for execution, opening up system options for lower resource spending beyond what can be derived by the system on its own, i.e. by SK considerations. As an example of this, consider the umbrella example of service requests that seek to **track a moving subject across a geographical area** for a particular purpose. This could be, for example, a public safety scenario using public camera surveillance or other sensor-based means to track people, traffic or crowds in a city, or a racing sport live broadcasting scenario where viewers track their favorite racer. Eventual service variations could use the moving subject location (or its derivative) information to feed a navigation app, giving directions to navigate to the subject, or could use the live location stream to select the right camera sources to obtain a live video compilation permanently showing the mobile subject.

In such scenarios, predicting the location where the subject is likely to appear in the near future, i.e. the likelihood of to where the trajectory will extend, is valuable RW information on top of the currently detected location. In the case of a camera network, subject-detection algorithms need to be deployed for each camera instance where the subject is potentially appearing. Without trajectory prediction of any kind, the resource cost would be proportional to the number of cameras to be assessed in the wider area, whereas when trajectory prediction information is available, the resource cost becomes proportional to the confidence of the prediction.

Essential to this prediction confidence is **the knowledge of what we could denote as 'the rules of behavior'** for subjects like the ones tracked, i.e. the relevant RWK Model part. For example, this could entail a knowledge collection equivalent to a set of relations like "*When a vehicle passes junction A, it has*

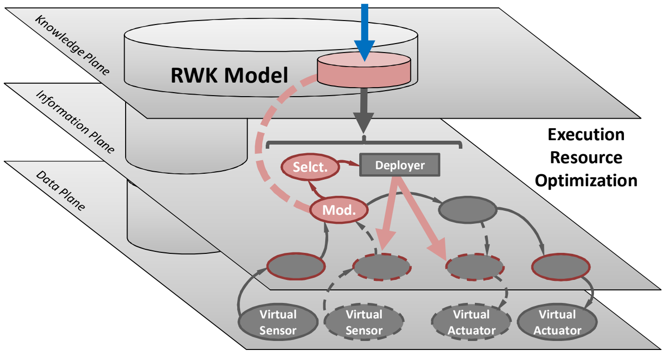

Figure 2. Real-world knowledge-based execution resource optimization.

a likelihood of 65% to pass junction B in time period T after it.". Based on such RWK, in a camera network case, the probabilistic relevance of particular camera locations, to which algorithm instances need to be deployed live during service execution for ad-hoc subject detection, can be derived and allows narrowing down the needed number of such live algorithm redeployments, and so resource spending can be lowered significantly. Fig. 2 illustrates this schematically; the part of the RWK Model as relevant to the service (processing node labeled "Mod." in the picture) is fed with observation data that confirms the model (tuned simultaneously in this case) and actualizes the observed status. This allows anticipating which observations (and actuations) are most likely to be needed by the service in the near future. For that, a selection node (labeled as "Selct.") is fed with the model status predictions from the modeling node, in order to instruct a deployment infrastructure ("Deployer" in the picture) with newly needed service sub-graphs, eventually leading to the corresponding just-in-time deployments (abstracted as the two pink arrows).

We conducted an experiment using a Hidden Markov Model (HMM, a basic type of Bayesian network [23]) for modeling the sequence of cameras along a race track, predicting where cars would likely appear next. For services to which this world behavior is relevant, e.g. services requiring images showing particular vehicles passing by, we showed that the model could quickly stabilize during service operation, under quite noisy conditions, leading to a video processing and streaming reduction of approximately 50%.

D. Reuse of Real-World Knowledge across services

An important aspect to emphasize in the previous discussion is the aim to leverage RWK *explicitly*, rather than implicitly as part of actual service instances. Indeed, in the resource optimization example, one could equally well obtain the optimal service execution by straightforwardly building a service assuming the RWK model as an undistinguished part of the service logic. However, as will become more apparent with the mechanisms discussed in the next section, distinguishing RWK as part of a *common* world model applicable for many service request types, and across many service or even observed subject instances, not only allows **code or execution node instance reuse**, but especially is a needed foundation for getting to a statistically relevant scale of RW information for *maintaining and extending* an RWK Model as part of a service platform, to the growing benefit of ever smarter service support.

In fact, the frequent reoccurrence of particular generic, often application domain-independent **real-world phenomena**, actually happening or existing in the real world, can be modeled as part of the RWK Model, thus occurring as a dependency across a wide variety of services and service instances, executing during long or short times, in service instances running subsequently or in parallel. The fact that a modeled phenomenon would effectively reoccur in a statistically relevant way as reused across a large amount of service instances, allows for **applying machine learning techniques 'in-situ' of the service execution process**, gradually improving the model of the particular phenomenon as part of the RWK Model, according to the impact the model has in the real world, via each service execution instance it is part of. Real-world phenomena that could be modeled as part of the RWK Model, to the benefit of various optimizations, can be of a wide range, such as:

- most typical examples like *behavior* of city traffic phenomena, from (interrelations between) individual vehicles or people, up to traffic jams, crowds, etc. given a particular city topology pattern, but also
- more *static* environmental properties such as the topology of buildings, a city or a stadium, including the position, orientation or other qualifiers of observation and actuation infrastructure, as in the previous race track cameras example, in particular when such model data is not easily measurable or only available by means of considerable field operations cost, as well as
- more *'hidden'* phenomena, only indirectly apparent in the environment, such as the influence of weather conditions on city traffic, for which identification would be beneficial.

Note that the optimization advantages based on such modeled knowledge can span aspects beyond resource spending at the level of network connectivity and processing, such as energy footprint of (a specific platform-internal or external part of) an overall ecosystem, or human attention resource occupation. Home or mobile *personal activity detection* [24] can thus be considered a specific case of the general real-world phenomena modeling concept, and so the concept can be applied for optimizations such as:

- the number of false-positive alerts posted to a doctor who is monitoring a patient in an e-health service, or
- the energy savings in the classic home automation cases of lighting and heating control,

for which dedicated solutions are gaining popularity today.

In fact, the successful application of machine learning in such dedicated cases shows that stable modeling is feasible for at least the phenomena under study [24][25][26].

IV. Cognitive Loop Knowledge Hypothesis Validation

Beyond the direct training of a part of the RWK Model as was discussed in section III.C, in particular towards the learning of the model properties and behavior of real-world phenomena as introduced in the previous section, we introduce a generalized mechanism to systematically validate knowledge hypotheses by means of a cognitive feedback loop structure.

A. General Cognitive Loop Principle

Fig. 3 shows how a ***Hypothesis*** collected as part of the RWK Model, e.g. as postulated by a domain expert for validation and reuse in a particular application domain or even a broader more universal scope, is used do derive situational data in the context of the processing graph of an executing service instance (processing node labeled as "Hut.", *Hypothesis-under-test*). The service instance interacts with the real world typically as a real-time control loop (i.e. actuating, steering or correcting the service effect upon sensing its result),

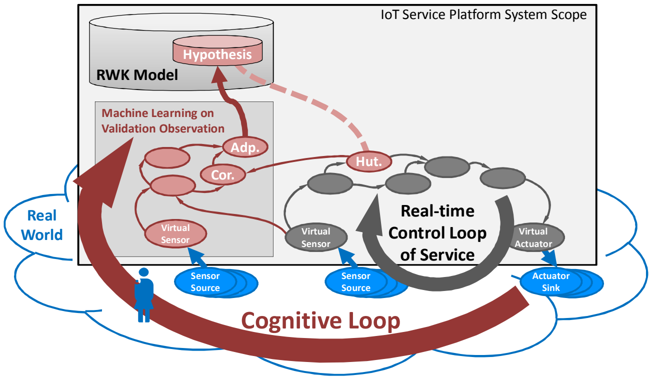

Figure 3. General Cognitive Loop overview.

in which the model of the Hypothesis and its predictive capacity thus is effectively used.

Considering now many service instances leveraging the same Hypothesis, or service instances using it during a long time, optionally making use of additional observations of the real-world response to the executing service (represented in Fig. 3 as an additional Virtual Sensor), indirect statistical evidence can be collected about the correctness of the Hypothesis in the specific service context. The box labeled "Machine Learning on Validation Observation" in Fig. 3 contains the Hypothesis-specific processing to collect and interpret the feedback received from an extended ***Cognitive Loop*** according to a prescribed validation 'procedure' for the Hypothesis (most notably containing a *Correlation* node, labeled "Cor.", and an *Adaptation Strategy* node, labeled "Adp." in Fig. 3, as we will discuss in the upcoming section IV.A.). While this validation mechanism is stored as part of the RWK Model, in a similar way as a Service Template, its main part is executed as *one* instance shared among *multiple* service instances that contain a Hypothesis-under-test node. As such, the validation mechanism leverages a high amount of context-relevant live feedback data from the real-world Cognitive Loop, e.g. including implicit or explicit feedback by service users, to tune parameters of the Hypothesis in the service context, **gradually improving statistical confidence of those parameters over time**. The adaptation of the parameters is done incrementally, according to a fixed strategy, after statistical evidence is accumulated, then closing the Cognitive Loop either by updating running service instances immediately (if the nature of the service permits it) or using the updated version for future service instance deployments. The Cognitive Loop, due to the validation data accumulation, runs typically much slower than what would be the case for a service-specific real-time control loop, relaxing also the *onlineness* requirement of the embedded machine learning, compared to what is possible when training directly in the Hypothesis-under-test node.

B. Elaboration of an Example Case

Consider again the example case of the **traffic management domain**. For various services as used by citizens, the municipalities or public transport operators, it is of interest to identify which traffic densities are significant, assuming that such is a common denominator across that family of services, spanning e.g. road maintenance works, route planning or live navigation. A cognitive feedback loop is thus deployed from each service template context as part of the Hypothesis validation. Let us assume a service context of e.g. a live navigation service offered to citizens, which takes into account current traffic jams along candidate routes, using the Hypothesis to estimate their significance in the context. The feedback loop in this case can consist of the observation of trajectories of (vehicles of) citizens that use the service. If citizens are observed to follow the routes as suggested by the navigation service indeed, this is considered a success for the Hypothesis for the road segments on that route. If however citizens systematically ignore a traffic obstruction indicated as significant by the Hypothesis for a particular road segment, then this is considered a failure of the Hypothesis for that road segment. (Although there may be individual measurements that add noise to the observation data, e.g. preferences or opinions of some individual service users, not objectively related to the traffic conditions and best-route options, repeated and consistent feedback on the Hypothesis will become apparent in this way, as a validation for it on a collection of road segments.)

Keeping the example domain in mind, Fig. 4 denotes:

- ***H*** as the Hypothesis function that, based on its sensor-based observation inputs, provides an output indicating the ***traffic obstruction significance for a specific road segment*** to which the sensor data related,
- ***A***, ***B*** and ***C*** as the respective outcomes of ***H*** for the road segments ***a***, ***b*** and ***c*** (the number of road segments considered in the process should be a sufficient number for proper Hypothesis validation, but a simple, demonstrative notation with only 3 instances is taken here),
- $\boldsymbol{p_a}$, $\boldsymbol{q_a}$ and $\boldsymbol{r_a}$ as the sensor-based observations related to road segment ***a*** (and analogues notations for road segments ***b*** and ***c***),
- ***u***, ***v*** and ***w*** as weight functions for each respective

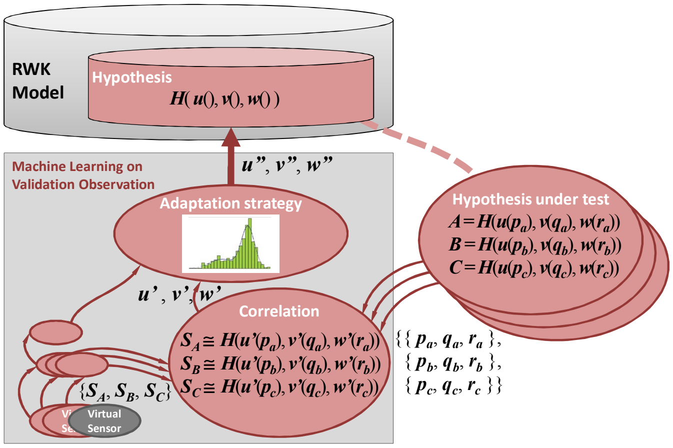
RWK Model
Hypothesis
$H(u(), v(), w())$
Machine Learning on Validation Observation
u'', v'', w''
Adaptation strategy
u', v', w'
Correlation
$S_A \cong H(u'(p_a), v'(q_a), w'(r_a))$
$S_B \cong H(u'(p_b), v'(q_b), w'(r_b))$
$S_C \cong H(u'(p_c), v'(q_c), w'(r_c))$
$\{S_A, S_B, S_C\}$
Virtual Sensor
Hypothesis under test
$A = H(u(p_a), v(q_a), w(r_a))$
$B = H(u(p_b), v(q_b), w(r_b))$
$C = H(u(p_c), v(q_c), w(r_c))$
$\{\{p_a, q_a, r_a\},$
$\{p_b, q_b, r_b\},$
$\{p_c, q_c, r_c\}\}$

Figure 4. Cognitive Loop-based Hypothesis validation mechanism.

sensor-based observation input $\boldsymbol{p_x}$, $\boldsymbol{q_x}$ and $\boldsymbol{r_x}$ (e.g. basic single-parameter polynomials) as internal components to $\boldsymbol{H}$, allowing adaptation of its variance with $\boldsymbol{p_x}$, $\boldsymbol{q_x}$ and $\boldsymbol{r_x}$ (note also here that the number of weight functions essentially equals the number of observation inputs chosen for $\boldsymbol{H}$, and is just considered 3 here for simplified notation), and finally

- $\boldsymbol{S_A}$, $\boldsymbol{S_B}$ and $\boldsymbol{S_C}$ as the a posteriori expected outcomes of $\boldsymbol{H}$ as derived from the Cognitive Loop feedback observation, e.g. the observation of implicit user feedback of vehicles not following navigation-suggested routes.

For each road segment $\boldsymbol{X}$, the outcome of $\boldsymbol{H}$ as used in the Hypothesis-under-test processing nodes of a range of active service instances can thus be expressed as

$$\boldsymbol{X} = \boldsymbol{H}(\ \boldsymbol{u}(\boldsymbol{p_x}),\ \boldsymbol{v}(\boldsymbol{q_x}),\ \boldsymbol{w}(\boldsymbol{r_x})\ )\ . \qquad (1)$$

As schematically represented in Fig. 4, the *Correlation* node in the validation mechanism needs to perform an **optimization of the Hypothesis weight functions** such that the correlation between the vector of observed $\boldsymbol{S_X}$ and the Hypothesis function applied to the corresponding sensor-based observations is maximized , i.e. such that

$$\boldsymbol{S_X} \cong\ \boldsymbol{H}(\ \boldsymbol{u'}(\boldsymbol{p_x}),\ \boldsymbol{v'}(\boldsymbol{q_x}),\ \boldsymbol{w'}(\boldsymbol{r_x})\ )\ . \qquad (2)$$

This parameter matching can be considered as a (semi-) online learning technique, be it with a sample window designed to be at least larger than the delay of the real-world response time out of the Cognitive Loop observation (in the example this is in the order of magnitude of vehicle travel time for multiple road segments), and preferably larger in order to account for noisy observations.

At an appropriate sample window frequency, parameters updates for the new candidate weight functions $\boldsymbol{u'}$, $\boldsymbol{v'}$ and $\boldsymbol{w'}$ are thus accumulated and passed to the *Adaptation Strategy* node of the validation mechanism. As an optional second phase node, the *Adaptation Strategy* may accumulate further statistical evidence of the stability of new weight function parameters of $\boldsymbol{H}$, e.g. as probability distributions. A criterion for ultimately deciding on effectively updating the actual RWK Model of $\boldsymbol{H}$ with parameters to new values $\boldsymbol{u''}$, $\boldsymbol{v''}$ and $\boldsymbol{w''}$ may e.g. be based on a maximum threshold to the variance of the parameter distribution. As an additional optional design measure for further stabilization, the evolution of the Hypothesis parameters can further be tempered e.g. by linearly combining the parameter updates with the current parameter values, or similar low-pass filter operations.

Slow iterative updates of *H* thus result, as applicable to the context of the considered collection of road segments, and as relevantly 'colored' for the envisioned service family. The updated Hypothesis will consequently be used for any future service instances of that service family (and can optionally also be adopted immediately in running service instance graphs, if their logic permits this).

The **probabilistic confidence of *H*** may also be stored as part of the RWK Model, concluding ever higher confidence when the adaptation strategy reveals consistently converging weight function parameters, ultimately indicating the experimental 'proof' of the Hypothesis. A human supervisor, as domain expert, may survey the evolution of the results and could potentially distinguish (by design) multiple refined models, e.g. by distinguishing different Hypothesis parameters or probabilistic confidence for different types of road segments.

C. Advantages and possible limitations of the approach

In retrospective, we introduced a generically decomposed learning approach in a service platform context, splitting out by design

- the RWK Model containing postulated Hypotheses, to be gradually and systematically validated as generally applicable for a large service domain, i.e. distinguished from more dedicated service template logic also contained in the RWK Model,
- the executed instances of such Hypotheses, under test in instances of the range of related services, and
- the cognitive feedback validation graph for each Hypothesis, containing additional components to the decomposed learning mechanism.

The systematic support of a growing collection of new real-world knowledge, leveraging the modeling of real-world phenomena, can be seen as a strong value advantage for a horizontally offered IoT service platform, e.g. with respect to situation-aware service expression and efficient service execution. By choosing such a decomposed learning design, however, we may also have introduced modeling limitations as compared to a direct, non-programmable single-service platform design with service-specific online model learning. For example, dedicated machine learning techniques may be shown to stabilize in a more controlled way than the proposed multi-step approach, however with our approach being applicable more flexibly for broader observation and feedback conditions.

A potential problem of *concept drifting* or non-convergence may occur, e.g. when a real-world concept assumed by the Hypothesis would not hold in the whole range of the service family for which the Hypothesis was designed, possibly requiring a posteriori domain expert management of such problem cases. However, this is not to be confused with the *deliberate bias* introduced by the 'coloring' of the observations in the approach through the in-service-context validation. This is indeed deliberate, as the Hypotheses are meant to highlight the RWK aspects *as relevant* to the assumed service family, as exactly this leads to the optimization and service expression benefits envisioned.

Despite the potential pitfalls, a notable resemblance exists to (partly more dedicated) learning systems researched in the ubiquitous computing field, such as *user preference learning*, to which our cognitive loop validation graph is a generalization, or the *FP7 Socionical* [25] and *FP7 Opportunity* [26] work where specific machine learning techniques are integrated into a dedicated service system with real-world feedback. Also some parallels to the technique of *transfer learning* [27] can be noted, as the learning functions and modeling done in the cognitive validation graph are in a sense transferred to the Hypothesis under test modeling that uses other, potentially more limited sensor-based observations.

Inspired by the strengths and weaknesses of these more dedicated approaches, where specific machine learning techniques have been selected, our further experimental iterations validating the cognitive loop approach may consider Bayesian classifiers of Hypothesis success, and the Hypothesis under test processing node may leverage a Markov or Artificial Neural Network model as seeded from the RWK Model.

V. Conclusion

In this paper we have proposed a novel approach to enhance an IoT service platform with 'in-situ' mechanisms for incrementing the leveraging of real-world knowledge. To this extent, we have distinguished different knowledge and information aspects of the architecture, and positioned hypothesis-based machine learning in the service request, deployment and execution process, thereby extending the service execution with a cognitive feedback loop. We envision that this may lead to essential new enabling value for IoT service platforms, by allowing such platforms to leverage models of real-world phenomena in more powerful service expression and more efficient service execution.

The described approach serves as a research framework for further experimentation concerning the use of particular machine learning technique combinations in concrete proof-of-concept example cases, in order to further validate the effectivity of the cognitive loop and real-world phenomena modeling aspects of the approach.

Acknowledgment

The work described in this paper is part of the contribution of Alcatel-Lucent Bell Labs in the context of the EU FP7 project iCore, contract number 287708.

References

[1] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, "Dremel: Interactive Analysis of Web-Scale Datasets", Proc. of the 36th International Conference on Very Large Data Bases, 2010.

[2] Hadoop Apache Project, http://hadoop.apache.org

[3] V. Khadilkar, M. Kantarcioglu, and B. Thuraisingham, "StormRider: Harnessing 'Storm' for Social Networks", 21st World Wide Web Conference, Lyon, France, April 16-20, 2012.

[4] "Revealed: Big Brother Britain has more CCTV cameras than China", DailyMail, August 2009. Available at http://www.dailymail.co.uk/news/article-1205607/Shock-figures-reveal-Britain-CCTV-camera-14-people--China.html

[5] ETSI Technology Cluster on Machine to Machine Communication (TC M2M), http://www.etsi.org/technologies-clusters/technologies/m2m

[6] OneM2M, http://www.onem2m.org/

[7] ETSI TC M2M Study Item on "Semantic support for M2M Data", M2M(12)18_006r2, 2012.

[8] FP7 IoT-A, www.iot-a.eu

[9] FI-WARE Context and Data Management, http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Data/Context_Management

[10] FP7 Sensei, www.sensei-project.eu

[11] C. Zins, "Conceptual Approaches for Defining Data, Information, and Knowledge", Journal of the American Society for Information Science and Technology (Wiley Periodicals, Inc.) 58 (4), 2007.

[12] W3C Semantic Web – Resource Description Framework (RDF), http://www.w3.org/2001/sw/wiki/RDF

[13] W3C Semantic Web – Ontology Language (OWL), http://www.w3.org/2001/sw/wiki/OWL

[14] W3C Semantic Web – SPARQL Query Language for RDF (SPARQL), http://www.w3.org/2001/sw/wiki/SPARQL

[15] C. Klein, R. Schmid, C. Leuxner, W. Sitou, B. Spanfelner, "A Survey of Context Adaptation in Autonomic Computing", IEEE ICAAS'08, 2008.

[16] P. Maes and R. A. Brooks, "Learning to Coordinate Behaviors", AAAI, Boston, MA, August 1990.

[17] R. C. Arkin, Behavior-Based Robotics, MIT Press, 1998.

[18] First Order Logic (definition in mathematics), http://mathworld.wolfram.com/First-OrderLogic.html

[19] R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T Sasaki, S. F. Smith, chapter "Introduction to Type Theory", Implementing Mathematics with The Nuprl Proof Development System, 1985, http://www.nuprl.org/book/Introduction_Type_Theory.html

[20] G. Cugola and A. Margara, "Processing Flows of Information: From Data Stream to Complex Event Processing", ACM Computing Surveys, Vol. 44, n. 3, ACM Press, May 2012.

[21] P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges", IEEE Transactions on Knowledge and Data Engineering, 2013.

[22] M. Ramezani, H. Friedrich Witschel, S. Braun, V. Zacharias, "Using Machine Learning to Support Continuous Ontology Development", ACM Proc. of 17th International Conf. on Knowledge engineering, Springer-Verlag Heidelberg, 2010.

[23] Tom M. Mitchel, Machine Learning, McGraw Hill Computer Science Series, 1997.

[24] C. L. Isbell, Jr, O. Omojukon, and J. S. Pierce, "From Devices to Tasks: Automatic Task Prediction for Personalized Appliance Control", Personal Ubiquituous Computing, Vol. 8, 2004.

[25] K. Zia, A. Ferscha, A. Riener, M. Wirz, D. Roggen, K. Kloch, and P. Lukowicz, "Pervasive computing in the large: The socionical approach", Adjunct Proceedings of the 8th International Conference on Pervasive Computing, Helsinki, Finland, May 2010.

[26] M. Kurz, G. Hölzl, A. Ferscha, A. Calatroni, D. Roggen, G. Tröster, H. Sagha, R. Chavarriaga, J. del R. Millán, D. Bannach, K. Kunze and P. Lukowicz, "The OPPORTUNITY Framework and Data Processing Ecosystem for Opportunistic Activity and Context Recognition", International Journal of Sensors, Wireless Communications and Control, Vol.1, 2011.

[27] S. J. Pan, Q. Yang, "A survey on transfer learning", IEEE Trans. On Knowl. and Data Engineering; 22(10), 2010.