IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2017, accepted June 8, 2017, date of publication June 15, 2017, date of current version July 24, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2716105

Probabilistic Models Toward Controlling Smart-* Environments

GÉRALD ROCHER[1,2], JEAN-YVES TIGLI[2], AND STÉPHANE LAVIROTTE[2]
[1]GFI Informatique, Groupe Innovation, 93400 Saint Ouen, France
[2]Université Côte d'Azur, CNRS, I3S, 06900 Sophia-Antipolis, France

Corresponding author: Gérald Rocher (gerald.rocher@gfi.fr)

This work was supported by GFI Informatique, Groupe Innovation, Saint Ouen, France.

ABSTRACT Today, a growing number of physical objects in our surroundings are connected to the Internet and provide the digital world with an interface to the physical world through *sensors* and *actuators*. At the heart of this trend, smart-* systems and applications leverage this interface to smartly and seamlessly assist individuals in their everyday lives. However, when interacting with the physical world by means of actuators, these applications introduce a methodological disruption. Indeed, in comparison to the classical distributed software applications that operate in the bounded and predictable digital world, these applications operate in and through the physical world, open and subject to *uncertainties* that cannot be modeled accurately. These uncertainties lead the behavior of the applications to potentially *drift* at runtime, compromising their intrinsic functionality. In this paper, we propose a framework to estimate the behavioral drift of smart-* systems and applications at runtime. To this end, we first rely on the Moore finite state machine (FSM) modeling framework. This framework is used for specifying the *ideal behavior* of a smart-* application in terms of the *effects*, and it is expected to produce within the physical environment as it executes. We then appeal on the control theory and propose a framework for projecting the Moore FSM to its associated continuous density Input/Output hidden Markov model (CD-IOHMM) *state observer*. By accounting for uncertainties through *probabilities*, it extends Moore FSM with *viability zones*, i.e., zones where the effects of a smart-* application within the physical environment are satisfactory without necessarily being perfect. At runtime, the CD-IOHMM state observer can compute the probability of the observed effects, i.e., it gives direct insight into the behavioral drift of the concrete application. We validate our approach on a real data set. The results demonstrate the soundness and efficiency of the proposed approach at estimating the behavioral drift of smart-* applications at runtime. In view of these results, one can envision using this estimation for supporting a decision-making algorithm (e.g., within a self-adaptive system).

INDEX TERMS Ambient intelligence, hidden Markov models, probabilistic modeling, smart-* systems and applications, ubiquitous computing, uncertainty.

I. INTRODUCTION

The last decade progresses in computer hardware miniaturization and power consumption reduction enable a growing amount of everyday life objects and physical environments to be wirelessly connected to the Internet (*Internet of Things*, IoT). By means of software services, they provide software applications with an interface to interact with our physical surroundings through *sensors* and *actuators*. This trend paves the way for opportunities covering a wide range of application areas promising huge sociological, economical and ecological impacts as shown quite clearly by the widespread interest for *smart-** applications and systems (*e.g.* smart-home, smart-building, smart-city, smart-factory, etc...). At a glance, smart-* systems and applications can be classified in two categories, namely:

1) **Inference systems and applications**. Applications within this category *consume* data gathered from sensors scattered in our surroundings or worn by the users (*i.e.* do not modify the environment) and rely on data mining techniques to infer *relevant* information. The responsibility of the actions to be undertaken is delegated to the end-users. These applications cover a large spectrum of fields ranging from mobile assistance [1] to context recognition [2], healthcare [3] and energy analytics [4], just to name a few.

2) **Automation systems and applications**. Besides sensors, many connected objects in our surroundings also embed actuators that the applications can control through software services in order to *modify* the environment. Leveraging the set of sensors and actuators available through a *synergistic* approach allows to imagine as many scenarios as relevant physical interactions with the users, the physical environment and the other surrounding objects. It is then possible to envisage applications offloading users with their physical and cognitive demanding tasks [5], managing resources efficiently, thereby drastically reducing their economical and ecological impacts [6], [7]. Here again, applications within this category cover a large spectrum of fields ranging from smart-home to smart-building, smart-city, smart-factory, etc…

In this paper, we focus on smart automation systems and applications as they introduce a clear methodological disruption inherent to their operational environment [8]. Indeed, as opposed to the classical distributed software whose intrinsic functionality is not supposed to depend on the underlying hardware and communication infrastructures, the functionality of automation systems and applications (simply named applications in the sequel) is achieved through a composition of services (1) interacting with each other *in* and *through* the physical environment by means of actuators, (2) whose availability is not necessarily ensured over time. Therefore, the intrinsic functionality of these applications is not immune to the parasitic interactions produced by the neighboring objects and physical processes [9]. Consequently, their behavior may potentially *drift* to the extent that it may even threaten their intrinsic functionality.

As an example, in Building Automation Systems (BAS) it is quite common for the sensors and the actuators to be linked into BAS rules (or more generally Event-Condition-Action (ECA) rules) that determine automation behaviors such as adjusting lighting based on whether people are around or not [10]:

```
1 rule 'LightOn'
2 when
3   Item Presence_office changed from 0 to 100
4 then
5   sendCommand(Light_office_dimmer, 200)
6 end
```

As such, the rule assumes that the office is effectively enlightened to the required luminosity value (*i.e.* 200 Lux) once the command has been issued. Somehow, it adopts a classical programing approach where, for instance, *writing* a value into the memory is not challenged and is assumed to be effective once the command has been issued thanks to some processes running in background whose role is to refresh, on a regular basis, the content of the memory cells. This assumption is obviously not applicable anymore when operating in and through the physical environment, subject to *uncertainties* [8] (a defective light bulb, a newly added furniture conceals the light bulb, etc…).

FIGURE 1. Most of the current applications delegate the responsibility of the actions to be undertaken by the users who are burdened with a continuous flow of information and notifications [16], [17].

Some BAS rely on *analytical models* of the considered physical environment for computing optimal lighting [11] (Fig. 4). However, models are abstractions of the real world and are, by definition, incomplete [12] and lack the integration of stochastic dynamics that would be necessary in the context of smart-* systems and applications operating in the physical environment [13]. Therefore, although efficient for critical systems whose environments are whether known or at least controlled over time [14], model based techniques are seriously limited as means to predict and provide guarantees that the functionality of the smart-* systems and applications is going to be met and maintained over time [15]. A direct consequence of this limitation is that most of the current applications delegate the responsibility of the actions to be undertaken to the users who are burdened with a continuous flow of information and notifications [16] (Fig. 1).

Thus, the objective is no longer to verify whether the behavior of an application is going to be conform or not over time, this is illusory in this context. More realistically, we aim at estimating, at runtime, the gap between the observed behavior and the expected ideal behavior, specified in terms of the effects the concrete application is expected to produce within the physical environment as it executes. As stated in [8], if we cannot completely determine system behavior or guarantee correct behavior in advance, we must find ways to make sure that systems work ''well enough''. The main idea behind our approach is to use a probabilistic modeling framework for specifying the effects an application is expected to produce within the environment as it executes. Here, the point is that the effects are specified irrespective of the concrete physical environment the application operates in and the underlying software components it is composed with. At runtime, the concrete application is seen as a *black box*. The observation of the effects of its interactions with the physical environment is then applied against the probabilistic model from which the probability of

the observed effects is computed. The probability gives direct insight into the conformity of the concrete application.

The contributions of this paper are the following:

1) We rely on the Moore FSM modeling framework for modeling the ideal behavior an application must meet in terms of the effects it is expected to produce within the environment as it executes [18] (Section IV-A). This modeling framework is very convenient and largely used for representing rule-based or event-based behaviors as it implements rules (*i.e.* states with associated expected observations) triggered by input events constraining the state transition dynamics [18].
2) We appeal on the control theory and the notion of *state observer*. The role of a state observer is to estimate the underlying state of a concrete system (not directly observable) by means of its dynamical model and the observation of its inputs and the effects it produces within the physical environment as it executes (*a.k.a. state estimation problem*). We assume that the dynamics of the physical environment are *non-linear* and possibly subject to *non-Gaussian noises*. Based on these assumptions, we propose to model the state observer through the *Hidden Markov Model* (HMM) probabilistic modeling framework [19] (Section IV-B).
3) The Moore FSM modeling framework doesn't allow to compute the conformity of the behavior of an application beyond a discrete PASS/FAIL result. Therefore, we provide a framework for projecting the model of the ideal expected behavior (Moore FSM) to its associated HMM-based probabilistic state observer (Section V). In a previous paper [20] we proposed to model the state observer through the *Continuous Density Hidden Markov Model* (CD-HMM) probabilistic modeling framework. The current paper extends the previous paper by considering modeling the state observer through the *Continuous Density Input/Output Hidden Markov Model* (CD-IOHMM) probabilistic modeling framework [21] whose semantics is intimately linked to the Moore FSM model (Section V-B). We then leverage the ability of the HMM-based state observer to estimate the probability of an observation sequence collected from sensors buried in the physical environment.
4) For the sake of clarity and ease of comprehension, we validate our approach through a simple yet concrete use-case in the field of smart home lighting automation from the MavHome dataset [22] (Section VI-A). The results demonstrate the soundness and the efficiency of the proposed approach at estimating the behavioral drift of applications at runtime in the presence of environmental disturbances and unexpected behaviors (Section VI-C). In light of these results, one can envision to use this estimation for supporting a decision-making algorithm, allowing applications to smartly react to unexpected environmental events.

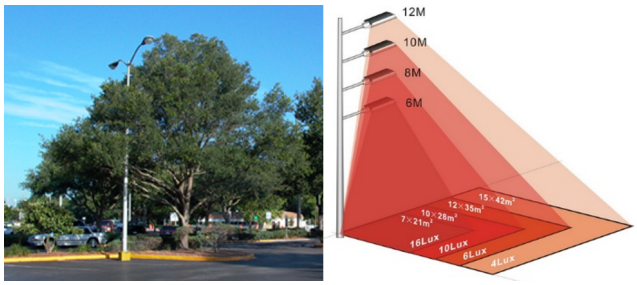

FIGURE 2. The tree conflicts with the street light and jeopardizes the model [24].

II. ILLUSTRATIVE USE-CASES

In this section we expose the problem through two use-cases in the fields of outdoor and indoor smart lighting automation.

A. OUTDOOR SMART LIGHTING

In 2013, there were more than 90 million traditional street lights in Europe, accounting for up to 60% of a typical electricity cost of a city [23], not to mention the ecological and environmental impacts (light pollution and CO_2 emission). Consequently, many cities have replaced old incandescent light bulbs with Light-Emitting Diode (LED) bulbs, leading up to 70% energy savings. However, most of the street lights remain uselessly switched on overnight and there is still room for energy consumption reduction. To address this problem, recent works propose to adaptively control street lighting, street lights in smart-cities being connected, they can be monitored and managed wirelessly [7]. Adaptive street lighting could be as complex as dimming different values depending on whether pedestrians or vehicles are present in the street or depending on whether the lunar illumination suffices to enlighten the streets (clear skies) or not (low clouds), etc... Current smart lighting solutions rely on sensors *embedded on the street lights* and more particularly on luminosity sensors as a means to ensure that the luminosity is at the right level whether vehicles or pedestrians are detected or not. However, as they operate in the physical environment, any unexpected event in between the street light and the area of the street to be enlightened may jeopardize the expected functionality. For instance, as trees along roads mature, they are likely to conflict with the street lights (Fig. 2). In addition to not match the functional expectations, it is likely to endanger the physical safety of pedestrians and car drivers (and this is worsened considering that the neighboring street lights now could potentially be switched off by the controller).

B. INDOOR SMART LIGHTING

Daylighting is of importance in the context of classrooms. It plays a significant role on students' well beings and numerous studies show a direct correlation between cognitive abilities and a good visual environment [25]. Consequently, architects rely on standards (*e.g.* Illuminating Engineering Society of North America [26]) and physical models to design classrooms satisfying luminosity requirements while

FIGURE 3. The luminosity in the classrooms is not homogeneously distributed [28].

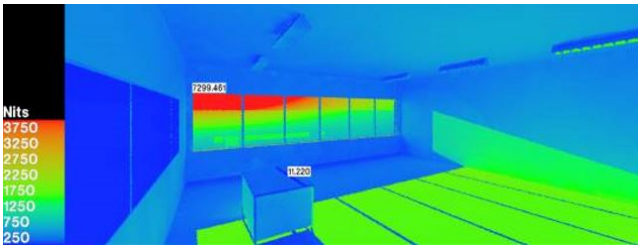

FIGURE 4. The distribution of the luminosity in a classroom based on a 3D model [28].

maximizing daylighting. However, due to the building orientation, weather, time of the day, classroom occupancy, etc…, the luminosity in the classroom is not homogeneously distributed over time. Consequently, classrooms are provisioned with a bunch of hardwired switches allowing to independently illuminate rows of school desks, blackboards, etc…. Managing lighting in such conditions requires teachers and students to mobilize cognitive resources and, *in fine*, lights, once switched on, are never switched off. This results in a huge waste of energy. According to the International Energy Agency (IEA):

"Lighting accounts for about 20% of global building electricity consumption. The latest scenarios show the total electricity savings potential in building lighting by 2030 could be equivalent to all the electricity consumed in Africa in 2013" [27].

Relying on users to manage lighting does not allow to ensure systems effectiveness both from users and energy savings perspectives. To address this problem, BAS are deployed in smart-buildings to adaptively control lighting in the classrooms. These systems are generally rule-based and possibly rely on a mathematical model of the building defined at design time (Fig. 4). Sensors and actuators are linked into rules determining the desired automation behaviors and validated once the building has been equipped.

However, unanticipated events may still occur inside and outside the classroom (Fig. 3): (1) the weather turns cloudy, (2) the shade of a tree is projected on the blackboard, (3) a furniture recently placed in the classroom prevents the luminosity to meet the expected level at some point in space, etc…At any time, models and rules are questioned and the behavior of the BAS is likely to drift over time. Such situations are uncountable in the context of applications acting within the physical environment through actuators.

Thus, although these applications promise huge economical, sociological and ecological breakthroughs, their underlying operational environment, by leading their behavior to potentially drift unexpectedly over time, creates a serious limitation in their capability to safely act on the behalf of the users. This calls for a mechanism aiming at estimating at runtime, from observations, the behavioral drift of these applications against their expected ideal behavior.

III. RELATED WORK

Provisioning smart-* systems and applications with the ability to estimate, at runtime, their behavioral drift against their expected ideal behavior is an open challenge. This challenge is at the heart of the *self-adaptive* systems (SAS) providing pervasive computing systems with *self-** properties. Indeed, when operating within open and uncertain environments, self-adaptation is required. This poses new challenges in terms of *assurance*, *i.e.*, the ability to provide *evidence* that the systems satisfy their behavioral requirements, irrespective of the adaptations occurring over time [29], [30]. Close to the problem addressed in the present paper, the authors in [31] are concerned with the quantification of the deviation gap from the original specified behavior of a SAS due to uncertainties and propose future research agenda to tackle this problem. At the base of this work is the notion of **Bayesian surprise** [32]. Design-time *beliefs* for specific decisions are specified using *Bayesian Dynamic Decision Networks* (DDNs) [32]. A Bayesian surprise then quantifies how observations affect beliefs at runtime by measuring the *distance* between the posterior and prior belief distributions. The distance is calculated by using the *Kullback-Leibler divergence*. Somehow, this approach leans closely to what is known in the field of machine learning and predictive analytics as **Concept drift** [33]. The notion of Concept drift is relative to the unexpected evolution of the statistical properties of a model variable, leading the model to deteriorate over time. Authors in [34] focus on quantitative measure of concept drift and introduce the notion of *drift magnitude* whose value can be quantified through distance functions such as Kullback-Leibler Divergence or *Hellinger Distance*.

These approaches focus on measuring the distance between the posterior and prior belief distributions of some model variables. The approach proposed in the present paper goes beyond and allows to compute the probability of an observation sequence, thereby encompassing the expected dynamics of the observed application.

Underlying the challenge of assurance, the notion of **viability zone** [35] is crucial to ensure the accuracy of the measures. The viability zone of a system is the set of states in which the system operation is not compromised, *i.e.* the set of states where the behavior of the system is satisfactory [36]. A viability zone is characterized in terms of relevant attributes and corresponding desired values. These attributes are associated with measurements of variables (either internal or environmental) whose variations can take the system outside its viability zone. Managing viability zones at runtime

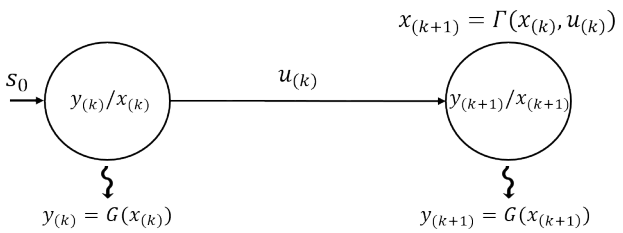

FIGURE 5. Moore Finite State Machine (FSM) in a nutshell.

is crucial for the assurance of SAS and is still an open challenge. To some extent, our approach allows to define the viability zone of an application through the probabilistic model parameters, *e.g.* mean, variance, etc.... Specifically, the model parameters can be *learned* during operation [37], allowing the viability zone to be refined at runtime.

Interestingly, the ideal behavior of the applications might be defined by the users themselves (*e.g.* through end-user programming [38]). In this context, the estimation of the behavioral drift of these applications is intimately linked into the **Quality of Experience** (QoE) [39] as an assessment of the human satisfaction when interacting with technology in a particular context. Although determining a measure of the QoE is not a trivial task, recently, researchers considered Artificial Intelligence (AI) and Machine Learning techniques for measuring the QoE. In [40], the authors propose to model users' satisfaction with HMM in the domain of Spoken Dialog Systems (SDS). The user's judgment about the dialog is modeled as states representing a specific judgment. Each judgment has a probabilistic relation to the current events in the dialog. Here a parallel can be done with our approach if one considers the states in the HMM-based observers presented in this paper as the user's judgment about the application with each judgment having a probabilistic relation to the current events in the physical environment.

IV. BACKGROUND

The theoretical foundations of this paper are based on two modeling frameworks, namely the deterministic Moore Finite State Machine (Moore FSM) modeling framework and the probabilistic Hidden Markov Modeling framework. We provide hereafter a brief overview of these modeling frameworks and identify their advantages and disadvantages.

A. MOORE FINITE STATE MACHINE

In this paper, we suggest modeling the expected ideal behavior an application must meet, as it executes, through the Moore Finite State Machine modeling framework (Moore FSM) [18]. The expected ideal behavior is defined in terms of effects the application is expected to produce within the physical environment as it executes. Moore FSM models systems dynamics as *deterministic processes* where each state is associated with an output emission.

More formally, a discrete-time Moore FSM is defined by the tuple $M = \langle S, S_0, I, Y, \Gamma, G \rangle$ (Fig. 5) where:

- $S = \{x_1, x_2, \ldots, x_N\}$ is the finite set of states. A state x visited at time k is denoted $x_{(k)}$,
- $S_0 \in S$ is the initial state the machine starts with,
- $I = \{u_1, u_2, \ldots, u_M\}$ is the finite set of input vectors; $u_{(k)}$ denotes the input vector at time k,
- $Y = \{y_1, y_2, \ldots, y_L\}$ is the finite set of expected output vectors; $y_{(k)}$ denotes the output vector at time k,
- Γ is the state transition function mapping a state and an input vector to the next state ($x_{(k+1)} = \Gamma(x_{(k)}, u_{(k)})$),
- G is the output function mapping each state to an expected output vector ($y_{(k)} = G(x_{(k)})$), *i.e.* what is expected to be *observed* while being in each state. The outputs of a Moore FSM depend only on their underlying states. Thus, as it executes, a Moore FSM produces an output sequence $y_{1:K} = \{y_1, y_2, \ldots, y_K\}$, where K is the sequence length.

Moore FSM equations can be stated as follow:

$$\begin{cases} x_{(k+1)} = \Gamma(x_{(k)}, u_{(k)}) & \text{(State equation)} \\ y_{(k)} = G(x_{(k)}) & \text{(Observation equation)} \end{cases} \quad \text{(IV-A.1)}$$

The state transition function Γ constrains the paths an application is expected to go through as it executes (*i.e.* the dynamics of the application). The function G defines the observations one can expect while being in a particular state (*i.e.* the effects the application is expected to produce within the physical environment as it executes). This modeling framework is largely used for representing rule-based or event-based behaviors as it implements rules (*i.e.* states with associated expected observations) triggered by input events constraining the state transition dynamics [18]. For instance, let's consider the two following BAS behavioral rules:

```
1  rule 'LightOn'
2  when
3    Item A1 changed from 0 to 100
4  then
5    sendCommand(Light_office_dimmer, 200)
6  end
7
8  rule 'LightOff'
9  when
10   Item A1 changed from 100 to 0
11 then
12   sendCommand(Light_office_dimmer, 10)
13 end
```

These two rules define the effects expected within the physical environment (luminosity in the office set to either 200 Lux or 10 Lux) in response to events (A1 changes). The resulting Moore FSM is depicted in Fig. 6. This modeling framework is convenient for specifying, in an intuitive manner, the ideal effects an application is expected to produce within the physical environment as it executes. However, due to the fact that I and Y are finite, this modeling framework suffers from several limitations in our context:

1) Firstly, it doesn't handle *noisy observations*, whereas sensors introduce noise into the gathered values,
2) Secondly, output values are idealized values. However, it is unlikely that, for a given state, the observed output

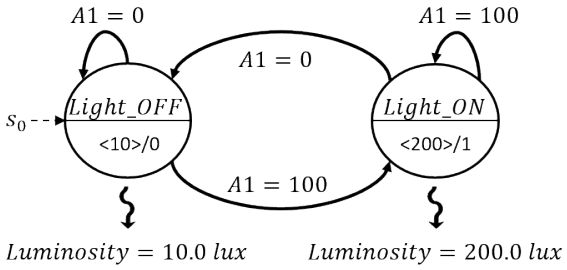

FIGURE 6. BAS rules transformed to equivalent Moore FSM.

corresponds exactly to the one expected, even though it might be still *acceptable* to the end-user.

Consequently, formal methods and tools associated with this class of modeling framework, as means to reason about the conformity of the behavior of an application beyond a simple PASS/FAIL result, are not directly applicable [15].

B. HIDDEN MARKOV MODELS

To solve the aforementioned limitations, we appeal on the control theory and the notion of *state observer*. The primary goal of a state observer is to estimate the underlying state $\hat{x}_{(k)}$ of an application by means of its dynamical model, the observation of its inputs and the direct and indirect effects of its execution within the operational environment.

In this paper, we consider applications whose operational environment is the physical environment. The physical environment dynamics are intrinsically *stochastic* and physical phenomena are most of the time *nonlinear*. Therefore, the applications, when operating within the physical environment through actuators and sensors, are driven by *random processes* (*non-Gaussian* noises, uncertainties and non-anticipated interactions) potentially yielding unexpected behaviors. So, from an observer point of view, a given application buried in the physical environment can be described by the following discrete-time stochastic dynamical system [41]:

$$\begin{cases} x_{(k+1)} = \Phi(x_{(k)}, \omega_{(k)}) & \text{(State process)} \\ y_{(k)} = \psi(x_{(k)}, v_{(k)}) & \text{(Observation process)} \end{cases} \quad \text{(IV-B.1)}$$

where $\omega_{(k)}$ and $v_{(k)}$ are respectively denoting the system and the measurement noises (that can be assimilated to unknown inputs affecting both states and observations). $\{\omega_{(k)}\}$ and $\{v_{(k)}\}$ are random processes, sequences of *independent and identically distributed* (iid) random variables. Φ and ψ denote any non-linear functions. The system defined by Eq.IV-B.1 is said stochastic because it is driven by the random processes $\{\omega_{(k)}\}$ and $\{v_{(k)}\}$.

In this context, and assuming that the expected behavior of the application can be modeled over a discrete and finite state space, the only state observer modeling framework allowing to *optimally* estimate the underlying states from environmental noisy observation sequences is the Hidden Markov Modeling framework (HMM) [41]. HMM belongs to the *Dynamic Bayesian Network* (DBN) class. It is a *Stochastic Finite State Machine* (SFSM) assuming modeled systems to have the Markovian property, *i.e.*, the state $x_{(k+1)}$ only depends on the previous state $x_{(k)}$. The model is called hidden because the underlying stochastic process (*i.e.* a sequence of states) affecting the observed output sequence is not completely observable.

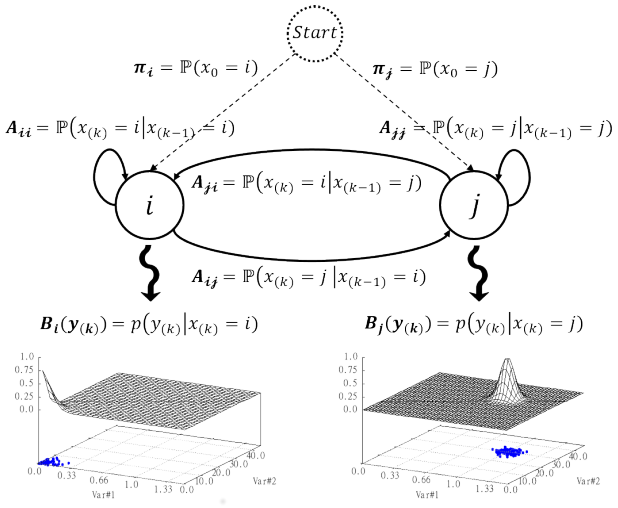

FIGURE 7. Multivariate CD-HMM in a nutshell.

More formally, a discrete-time finite-state HMM is defined by the tuple $H = \langle S, \pi, A, B \rangle$ where:

- $S = \{x_1, x_2, \ldots, x_N\}$ is the finite set of *hidden* states.
- $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ is the initial state distribution vector. $\sum_{i=1}^{N} \pi_{(i)} = 1$, where $\pi_{(i)}$ denotes the probability of the state i to be the first state of a state sequence.
- A is the state transition matrix ($N \times N$) of the underlying Markov chain. $A_{ij} = \mathbb{P}(x_{(k+1)} = j | x_{(k)} = i)$, $0 \leq A_{ij} \leq 1$, denotes the probability of being in state j at time $k+1$ given we are in state i at time k; $\sum_{j=1}^{N} A_{ij} = 1$.
- B is the state emission probability density function matrix. In this paper, we consider *univariate/multivariate continuous emissions*, where the emission probabilities are expressed, without loss of generality, as univariate/multivariate normal density functions.

$$B = diag(p(y_{(k)} | x_{(k)} = 1), \ldots, p(y_{(k)} | x_{(k)} = N)) \quad \text{(IV-B.2)}$$

Indeed, in the context of smart-* computing systems and applications, states can be characterized by emissions inherently *multidimensional* and *continuous*. This type of HMM (Fig. 7) is often referenced as Continuous Density HMM (CD-HMM). We denote $b_{x_{(k)}}(y_{(k)})$ the probability of being in the state $x_{(k)}$ and emitting the vector $y_{(k)}$.

The system equations can be stated as follows:

$$\begin{cases} x_{(k+1)} = \mathbb{P}(x_{(k+1)}|x_{(k)}) \Rightarrow A_{x_{(k+1)},x_{(k)}} \\ y_{(k)} = p(y_{(k)}|x_{(k)}) \Rightarrow B_{x_{(k)}}(y_{(k)}) \end{cases} \quad \text{(IV-B.3)}$$

HMM is used to solve the following canonical problems:

1) **Hidden state estimation problem**. Given an HMM with parameters $\Theta = \langle A, B, \pi \rangle$ and an observation sequence $y_{1:K} = \{y_1, \ldots, y_K\}$, evaluate the probability that the HMM ended in a particular state ($\mathbb{P}(x_{(K)}|y_{1:K})$). In other words, one obtains the log-likelihood ($]-\infty; 0]$) of a given observation sequence $y_{1:K}$ to have been produced by the model. In the HMM context, this computation is achieved by the classical recursive *forward algorithm* [37].
Let $\alpha_{(K)}(i) = \mathbb{P}(x_{(K)} = i|y_{1:K})$ be the probability that $x_{(K)} = i$ given the observation sequence $y_{1:K}$. Then, given

$$\alpha_{(1)}(i) = \pi_{(i)} b_{(i)}(y_{(1)}), \quad 1 \leq i \leq N \quad \text{(IV-B.4)}$$

where $\alpha_{(1)}(i)$ is the joint probability of starting in state i and observing $y_{(1)}$, the recursive computation

$$\alpha_{(k+1)}(i) = b_{(i)}(y_{(k+1)}) \left(\sum_{j=1}^{N} \alpha_{(k)}(j) A_{ji} \right) \quad \text{(IV-B.5)}$$

for $1 \leq k \leq K-1, 1 \leq i \leq N$, gives the joint probability of reaching the state i and emitting $y_{(1:K)}$.
2) **Hidden state sequence decoding**. Given an HMM with parameters $\Theta = \langle A, B, \pi \rangle$ and an observation sequence $y_{1:K} = \{y_1, \ldots, y_K\}$, decode the most probable underlying hidden state sequence Q that has been ran through to produce $y_{1:K}$. *Viterbi algorithm* is mainly used to this end.
3) **Model parameters learning**. Given an observation sequence $y_{1:K} = \{y_1, \ldots, y_K\}$, estimate the HMM parameters $\widehat{\Theta} = \langle \widehat{A}, \widehat{B}, \widehat{\pi} \rangle$. This can be done by using either supervised algorithms which expect the underlying state sequence to be associated with each observation sequence (*e.g.*, *Maximum Likelihood Estimate* (MLE)), or unsupervised algorithms (*e.g.*, *Baum-Welch algorithm*) which expect the number of hidden states and the state transition topology (*e.g.*, ergodic, forward, *etc*...).

This class of modeling framework seems to be well suited for reasoning about the conformity of the behavior of an application as it executes. Specifically, the solution it provides to the hidden state estimation problem through the log-likelihood estimation ($]-\infty; 0]$), can be used to give direct insight into the behavioral drift of an application interacting with the physical environment. However, developing a behavioral model through this framework is not trivial to the extent that it involves probabilities.

V. APPROACH OVERVIEW

In the sequel of the paper, we provide an overview of the approach we have adopted to take advantage of both modeling frameworks (Fig. 8), *i.e.* (1) the ability of the Moore FSM modeling framework to specify, in an intuitive manner, the behavioral rules and the expected effects an application must meet as it executes, (2) the ability of the HMM modeling framework to reason about the conformity of the behavior of an application as it executes.

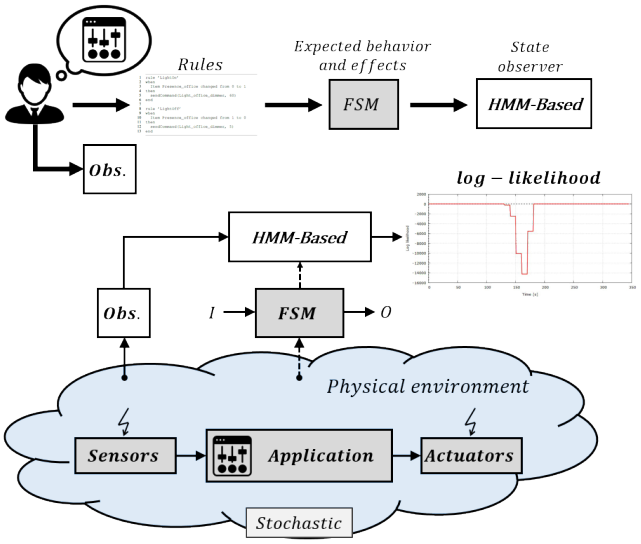

FIGURE 8. Approach overview. The model of the ideal expected behavior of an application is specified in terms of the effects it has to produce within the physical environment as it executes (Moore FSM). This model is then projected into its probabilistic HMM-Based state observer counterpart. The solution the state observer provides to the hidden state estimation problem through the log-likelihood value ($]-\infty, 0]$), gives direct insight into the behavioral conformity of the application as it executes.

A. MOORE FSM TO CD-HMM STATE OBSERVER PROJECTION

Let's assume that the effects an application is expected to produce within the physical environment as it executes are modeled through the Moore FSM modeling framework. The problem is then to find a way to project this model to its associated CD-HMM state observer. To this end, we consider the Moore FSM from a probabilistic point of view.

As discussed in the Section IV-A, the state transition function Γ of a Moore FSM maps a state $x_{(k)} \in S$ and an input vector $u_{(k)} \in I$ to a next state $x_{(k+1)} \in S$ ($x_{(k+1)} = \Gamma(x_{(k)}, u_{(k)})$). This can be traduced by (Fig. 9):

1) The probability of the occurrence of the input $u_{(k)}$ given the current state is $x_{(k)}$ $\left(\mathbb{P}(u_{(k)}|x_{(k)})\right)$,
2) The probability that this occurrence leads effectively the state transition $x_{(k)} \rightarrow x_{(k+1)}$ $\left(\mathbb{P}(x_{k+1}|x_{(k)}, u_{(k)})\right)$.

This leads a *transient* state to appear in the HMM state observer (Fig. 10). Indeed, the state transition probabilities are hard-coded in an HMM (matrix A) and do not depend on the occurrence of an input. Therefore, in order to qualify each state through observations, the output vector $y_{(k)}$ is partitioned with the input vector ($Y_{(k)} = (y_{(k)}, u_{(k)})$). It implies that the inputs have to be observed. For a given

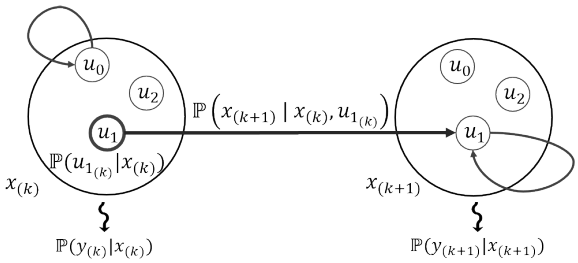

FIGURE 9. Moore FSM from a probabilistic point of view. Being in the state $x_{(k)}$ there is a probability of the occurrence of the input $u_{1_{(k)}} \in I$ ($\mathbb{P}(u_{1_{(k)}}|x_{(k)})$). Then $\mathbb{P}(x_{k+1}|x_{(k)}, u_{1_{(k)}})$ is the probability of the transition $x_{(k)} \rightarrow x_{(k+1)}$ given $x_{(k)}$ and $u_{1_{(k)}}$.

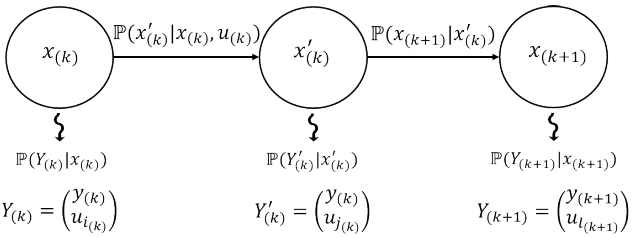

FIGURE 10. Considering Moore FSM from a probabilistic point of view leads (1) a transient state $x'_{(k)}$ to appear in the HMM state observer and (2) the output vector $y_{(k)}$ to be partitioned with the input vector $u_{(k)}$ ($Y_{(k)} = (y_{(k)}, u_{(k)})$).

Moore FSM, the probabilities discussed above are implicit and values constrained as follow:

1) **Input occurrence probabilities**. The occurrence of an input being *exogenous* to the model, given a state $x_{(k)} \in S$, any input $\{u_0, \ldots, u_\chi\} \in I$ that can lead a transition $x_{(k)} \rightarrow x_{(k+1)}$ may occur at time k.

$$\forall x_{(k)} \in S, \quad \sum_{u=u_0}^{u_\chi} \mathbb{P}(u_{(k)}|x_{(k)}) = 1 \quad \text{(V-A.1a)}$$

2) **State transition probabilities**. Moore FSM implicit constraint on state transitions (given by Γ) can be stated as follow: $\forall x_{(k)} \in S, u_{i_{(k)}} \in \{u_0, \ldots, u_\chi\} \in I$:

$$\exists! x_{(k+1)} \in S \backslash \mathbb{P}(x_{(k+1)}|x_{(k)}, u_{i_{(k)}}) = 1, \quad \text{(V-A.2a)}$$

$$\sum_{x_{(k+1)}=x_1}^{x_N} \mathbb{P}(x_{(k+1)}|x_{(k)}, u_{i_{(k)}}) = 1, \quad \text{(V-A.2b)}$$

$$\sum_{x_{(k+1)}=x_1}^{x_N} \mathbb{P}(x_{(k)}, x_{(k+1)}) = 1 \quad \text{(V-A.2c)}$$

Eq.V-A.2a and Eq.V-A.2b put together stipulate that a couple $x_{(k)}, u_{i_{(k)}}$ at time k is associated with a unique state $x_{(k+1)}$ at time $k+1$ (determinism). Eq.V-A.2c stipulates that the outgoing transitions from each state are equiprobable ($=\frac{1}{\chi}$). Indeed, inputs being exogenous to the model, one cannot presume the probabilities of the input occurrences (See V-A.1a).

3) **Output emission probabilities**. Moore FSM implicit constraints on outputs can be stated as follows: $\forall x_{(k)} \in S$:

$$\exists! y_{(k)} \in Y \backslash \mathbb{P}(y_{(k)}|x_{(k)}) = 1, \quad \text{(V-A.3a)}$$

$$\sum_{y_{(k)}=y_1}^{y_N} \mathbb{P}(y_{(k)}|x_{(k)}) = 1, \quad \text{(V-A.3b)}$$

Eq.V-A.3a and Eq.V-A.3b put together stipulate that each state is associated with a unique output vector.

Projecting a Moore FSM to its CD-HMM counterpart consists in computing the probabilities of the state transition matrix A, namely $\mathbb{P}(x'_{(k)}|x_{(k)}, u_{j_{(k)}})$ and $\mathbb{P}(x_{(k+1)}|x'_{(k)})$ (Fig. 10), the output emission probabilities of the matrix B and the initial state distribution vector π.

$\mathbb{P}(x'_{(k)}|x_{(k)}, u_{j_{(k)}})$ is given by Eq.V-A.2a and Eq.V-A.2b and is equal to $\frac{1}{\chi}$. Moore FSM constrains the transition $x_{(k)} \rightarrow x_{(k+1)}$ on the occurrence of the input and assumes that the transition is effective once the input has been triggered ($x_{(k+1)} = \Gamma(x_{(k)}, u_{(k)})$. Thus, $\mathbb{P}(x_{(k+1)}|x'_{(k)}) = 1 - \varsigma$.[1]

Assuming a model reduction is applied, the total amount of states $\aleph$ in the resulting CD-HMM is equal to $T - N$ where N and T are respectively the amount of states and transitions in the Moore FSM. Thus, the state transition matrix A is an $\aleph \times \aleph$ matrix.

The initial state distribution vector $\pi = \{\pi_1, \ldots, \pi_\aleph\}$, where π_i denotes the probability of the state i to be the first state of a state sequence, is computed as follow: the states defined in the Moore FSM get equiprobable values ($\frac{1}{N}$) while the transient states get zero probability.

The output emission probabilities defined in the matrix B (Eq.IV-B.2) are probability density functions expressed, without loss of generality, as univariate or multivariate normal density functions depending on the number of emission variables used to characterize a state. In this context, the probability density functions are defined through $\mu = \mathbb{E}(y_{(k)})$, an n-dimensional vector (or simply the mean value for univariate density functions) and Σ, the $n \times n$ positive definite variance-covariance matrix of $y_{(k)}$ (or simply the standard deviation for univariate density functions). μ can directly be retrieved from the Moore FSM model output vectors ($y_{(k)} = G(x_{(k)})$ in Eq.IV-B.2). Variance, covariance or standard deviation values cannot be retrieved directly from the Moore FSM model and have either to be defined off-line by the users or learnt from observations (see model parameters learning in Section IV-B).

Following the aforementioned methodology, the FSM depicted in Fig. 6 is projected into its associated CD-HMM state observer depicted in Fig. 11.

As previously discussed, the parameters of the emission probability functions (matrix B) define viability zones (Fig. 12).

[1]Depending on the sampling rate of the observation, there is a non-negligible probability ς for a transient state to loop back on itself (*i.e.*, the observed values lead the CD-HMM to consider that the application is still in the same state due to the observation probability density functions defined in the matrix B).

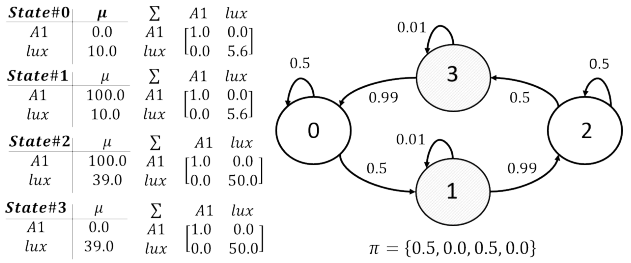

FIGURE 11. CD-HMM state observer corresponding to the Moore FSM depicted in Fig. 6. We assume that the probability density functions in the matrix *B* are multivariate normal distributions. Here, the parameter values of the distributions have been populated from the dataset used for the validation of the approach (Section VI-A).

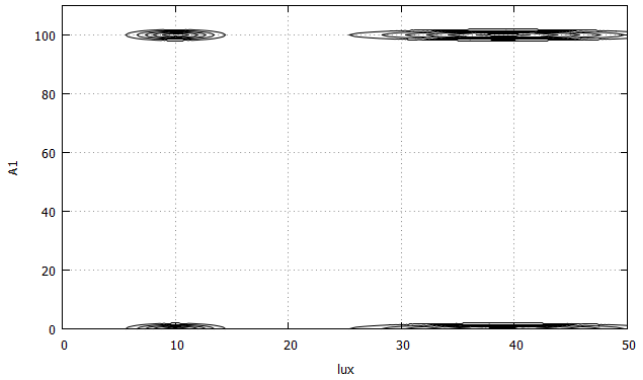

FIGURE 12. The parameters of the emission probability functions (matrix *B*), expressed as multivariate normal distributions, define viability zones, *i.e.* zones where the effects of an application within the physical environment are satisfactory without necessarily being perfect. This figure represents the viability zones for all the state emissions defined in Fig. 11.

This approach, although showing good results at estimating the behavioral drift of applications [20] has several drawbacks: (1) as to take into account input events, it requires transient states to be added, (2) it implicitly assumes, through Eq.V-A.1a, that the input space is bounded, *i.e.* from a given state $x_{(k)}$ only a subset of inputs $\in I$ is likely to occur leading the transition to the next state $x_{(k+1)}$. However, it may exist numerous unknown input events in the physical environment that may lead the transition to an unknown state $x_{(k+1)}$ from the state $x_{(k)}$ (called ''*regulons*'' and ''*tyches*'' in the viability theory [35], respectively controls that are not identified (and thus not part of the model) or disturbances over which nobody has any control). As a consequence, it is unlikely for the probability values of the state transition defined in the model to sum to one.

For these reasons, we introduce in the sequel the *Continuous Density Input/Output Hidden Markov Model* (CD-IOHMM) modeling framework.

B. CONTINUOUS DENSITY INPUT/OUTPUT HIDDEN MARKOV MODEL

A CD-IOHMM [21] is an HMM for which the state transition and emission probabilities can be conditionally dependent on

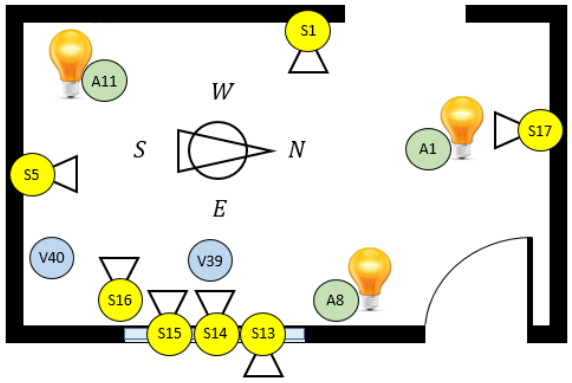

FIGURE 13. MavPad living room actuators and sensors location.

an input sequence (for this reason this model can also be found in the literature as Non-Homogeneous HMM (NHMM) [42]). In this paper, we consider *input-independent state-emitting* model variant [43], *i.e.* a model where only the state transition probabilities are conditionally dependent on an input sequence and whose semantics is intimately linked to the Moore FSM model.

We define the state transition matrix A as a $N \times N$ matrix where each cell of the matrix is the input probability density function. Then, given $u_{(k)}$, an input vector $\in I$ observed at time k, $A_{ij}(u_{(k)}) = \mathbb{P}(u_{(k)}|x_{(k+1)} = j, x(k) = i)$ and $A_{ijl} = \mathbb{P}(x_{(k+1)} = j|x_{(k)} = i, u(k) = l) = A_{ij}(l)$. Thus, the probability of transition from state $x_{(k)}$ to $x_{(k+1)} \in S$ is equal to the probability for the input $u_{(k)} \in I$ to trigger the state transition $x_{(k)} \rightarrow x_{(k+1)}$. Hence:

$$\forall x_{(k)} \in S, \quad \sum_{u=u_0}^{u_\chi} \mathbb{P}(u_{(k)}|x_{(k)}) \neq 1 \qquad \text{(V-B.1a)}$$

As for the observation probability density function matrix B (Eq.IV-B.2), we consider *univariate/multivariate continuous inputs*, where the input probabilities are expressed, without loss of generality, as univariate/multivariate normal density functions, *i.e.* a state transition can be dependent on one or multiple input events. This model can be referenced as Continuous Density Input/Output HMM (CD-IOHMM).

In this context, the **hidden state estimation problem** can be expressed as follow. Given a CD-IOHMM with parameters $\Theta = \langle A, B, \pi \rangle$, an observation sequence $y_{1:K} = \{y_1, \ldots, y_K\}$ and an input sequence $u_{1:K} = \{u_1, \ldots, u_K\}$, evaluate the probability that the CD-IOHMM ended in a particular state ($\mathbb{P}(x_{(K)}|y_{1:K}, u_{1:K})$). In other words, one obtains the log-likelihood of a given observation sequence $y_{1:K}$ to have been produced by the model knowing the input sequence $u_{1:K}$. This computation is still achieved by the classical recursive *forward algorithm* [37].

Let $\alpha_{(K)}(i) = \mathbb{P}(x_{(K)} = i|y_{1:K}, u_{1:K})$ be the probability that $x_{(K)} = i$ given the observation sequence $y_{1:K}$ and the input sequence $u_{1:K}$. Then, given

$$\alpha_{(1)}(i) = \pi_{(i)} b_{(i)}(y_{(1)}), \quad 1 \leq i \leq N \qquad \text{(V-B.2)}$$

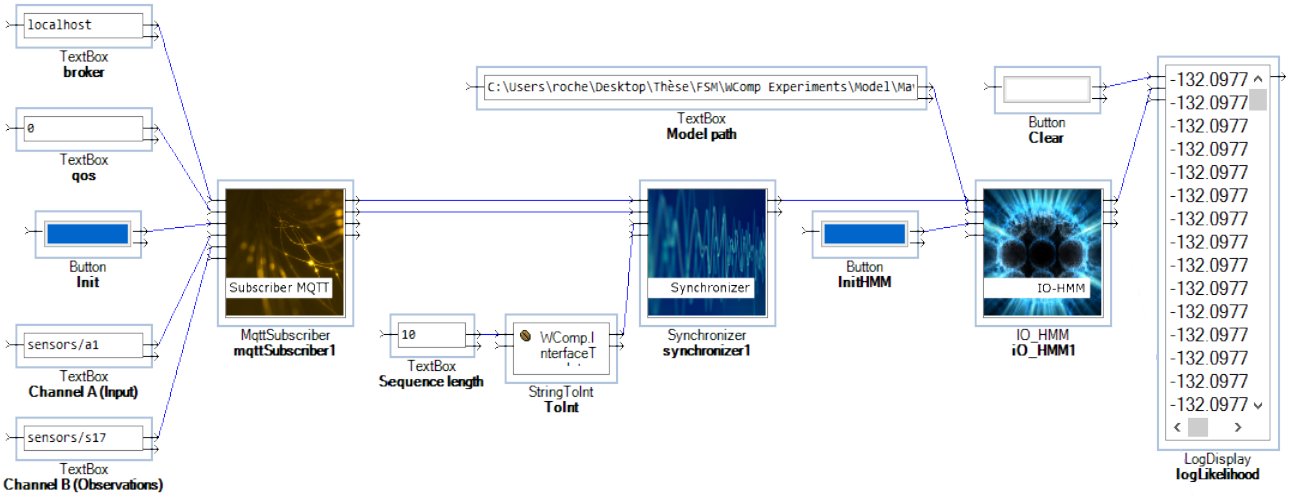

FIGURE 14. Data acquisition and treatment chain implemented as lightweight components instantiated within WComp [44], a middleware for ubiquitous computing. To fit as much as possible with a real scenario, each measure from the dataset is transmitted through MQTT, a lightweight publish/subscribe protocol designed specifically for machine-to-machine (M2M) and mobile applications. Then, the output and the input sequences are synchronized from time stamps before being transmitted to the CD-IOHMM component. The input sequence, the output sequence and the log-likelihood values are stored on a local ThingSpeak server [45].

FIGURE 15. Pictures from the MavPad apartment at the University of Texas at Arlington (TX, USA).

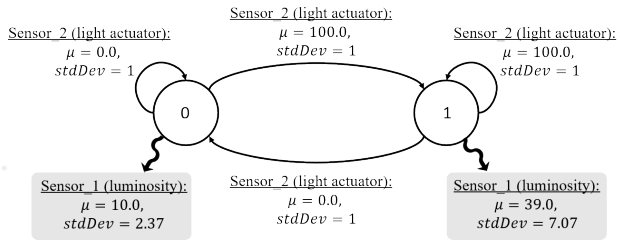

FIGURE 16. CD-IOHMM state observer corresponding to the Moore FSM depicted in Fig. 6. One sensor is used to observe the input event (light actuator A1) and one sensor is used to observe expected emissions for each state (luminosity sensor S17). Probability density functions are defined as univariate normal distribution.

where $\alpha_{(1)}(i)$ is the joint probability of starting in state i and observing $y_{(1)}$, the recursive computation

$$\alpha_{(k+1)}(i) = b_{(i)}(y_{(k+1)}) \left(\sum_{j=1}^{N} \alpha_{(k)}(j) A_{ji}(u_{(k)}) \right) \quad \text{(V-B.3)}$$

for $1 \leq k \leq K-1$, $1 \leq i \leq N$, gives the joint probability of reaching the state i and emitting $y_{(1:K)}$ knowing $u_{(1:K)}$.

C. CD-IOHMM STATE OBSERVER INITIALIZATION

The output emission probabilities from Eq.IV-B.2 and the initial state distribution vector π are computed in the same way as for the CD-HMM approach (Section V-B). The **state transition matrix** A is computed the same way as for the matrix B. A is a $N \times N$ matrix where each cell of the matrix is the input probability density function expressed, without loss of generality, as univariate or multivariate normal density functions depending on the number of input events required to characterize a state transition. In that case, $\mu = \mathbb{E}(u_{(k)})$ can directly be retrieved from the Moore FSM state equation $x_{(k+1)} = \Gamma(x_{(k)}, u_{(k)})$. Here again, variance, covariance or standard deviation values cannot be retrieved directly from the Moore FSM model and have either to be defined off-line by the users or learnt from observations (see model parameters learning in Section IV-B.

Additionally, the sensors used to gather observation and input sequences as input to the CD-IOHMM have to be specified. From the rules point of view, one can include statements for specifying the sensors to be used:

```
1 rule 'LightOn'
2 when
3   Item A1 changed from 0 to 100
4   measureState(A1, Sensor_1)
5 then
6   sendCommand(Light_office_dimmer, 200)
7   measureValue(Light_office_dimmer, Sensor_2)
8 end
```

VI. EVALUATION

A. DATASET

To evaluate our approach, we experimented with the MavPad published datasets from the MavHome (Managing an

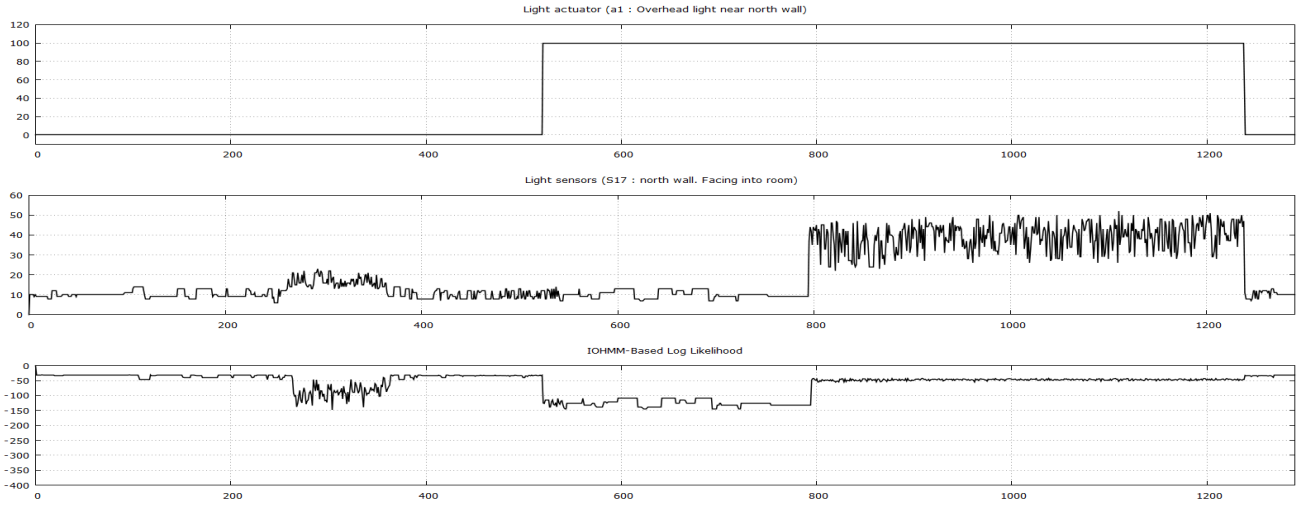

FIGURE 17. Results for the computation of the log-likelihood of an observation sequence $y_{1:K}$ (here the luminosity sensor S17 is used) given an input sequence $u_{1:K}$ and the CD-IOHMM state observer depicted in Fig. 16.

TABLE 1. MavPad living room actuators and sensors description.

ID	Type	S/A	Location
A1	Ceiling light	Actuator	Overhead light near north wall
A8	Table lamp	Actuator	Near door on side table
A11	Floor lamp	Actuator	Southwest corner
S1	Light	Sensor	West wall. Facing into room
S5	Light	Sensor	South wall. Facing into room
S13	Light	Sensor	Between window and blinds, facing out
S14	Light	Sensor	Between window and blinds, facing in
S15	Light	Sensor	Near window, facing miniblinds
S16	Light	Sensor	East wall. Facing into room
S17	Light	Sensor	North wall. Facing into room

Adaptive Versatile Home) project [22]. MavPad is an on-campus apartment at the University of Texas at Arlington (TX, USA) automated using 25 controllers and providing sensing of light, temperature, humidity, leak detection, vent position, smoke detection, CO detection, motion, and door/window/seat status. For the evaluation we focused on the lighting actuators and sensors in the living/dining room (Fig. 13, Fig. 15) described in Table 1.

B. EXPERIMENTATION SETUP

We used the data stored in the file `7-12-2004-raw.data` of the dataset. From the data, we issued a simple Moore FSM for describing the expected behavior of an application controlling lighting in terms of the effects it is expected to produce within the physical environment as it executes. To this end, we modeled the Moore FSM with *fizzim*, a free open-source GUI-based FSM design tool [46]. We then developed a tool for projecting the Moore FSM to its associated CD-IOHMM state observer. The CD-IOHMM state observer is built on top of the Accord.NET framework [47]. The resulting Moore FSM is depicted in Fig. 6 and its CD-IOHMM state observer counterpart is depicted in Fig. 16. The mean and standard deviation values for the probability density functions of the state transition and emission matrices have been computed from the dataset.

The CD-IOHMM engine is instantiated as a component in the WComp platform, a middleware for ubiquitous computing [44] allowing service composition by assembling lightweight components. It implements the SLCA model (Lightweight Service Component Architecture) [48] where the application is formed by an assembly of software components based on the LCA model (Lightweight Component Architecture) and services communicating using events (Fig. 14).

C. RESULTS

We executed the engine using a representative luminosity sensor available in the living room, namely S17 (north wall) and the actuator A1 (Overhead light near north wall). The length K of the input and observation sequences, respectively $u_{(1:K)}$ and $y_{(1:K)}$, is set to 10.

Results using a CD-IOHMM state observer are depicted in Fig. 17. From 7AM to 10AM (in between indexes 200 and 400 in Fig. 17), the luminosity in the room, measured by the sensor S17 (north wall close to the light controlled by A1), increases as the sun rises. Here, the CD-IOHMM state observer detects a violation as the actuator A1 is set to 0 and the expected luminosity is 10 Lux. From 11AM, the state of the actuator A1 is changed from 0 to 100. However, for some unexpected reasons, the expected effect (*i.e.* living room enlightenment is supposed to be at 39 Lux from that time) is not perceived within the environment by the sensor S17. This violation here again is detected by the CD-IOHMM state observer (in between indexes 500 and 800 in Fig. 17). From 7.20PM (index 800), the luminosity measured by the sensor S17 is finally at the expected level and no more violation is detected by the CD-IOHMM state observer.

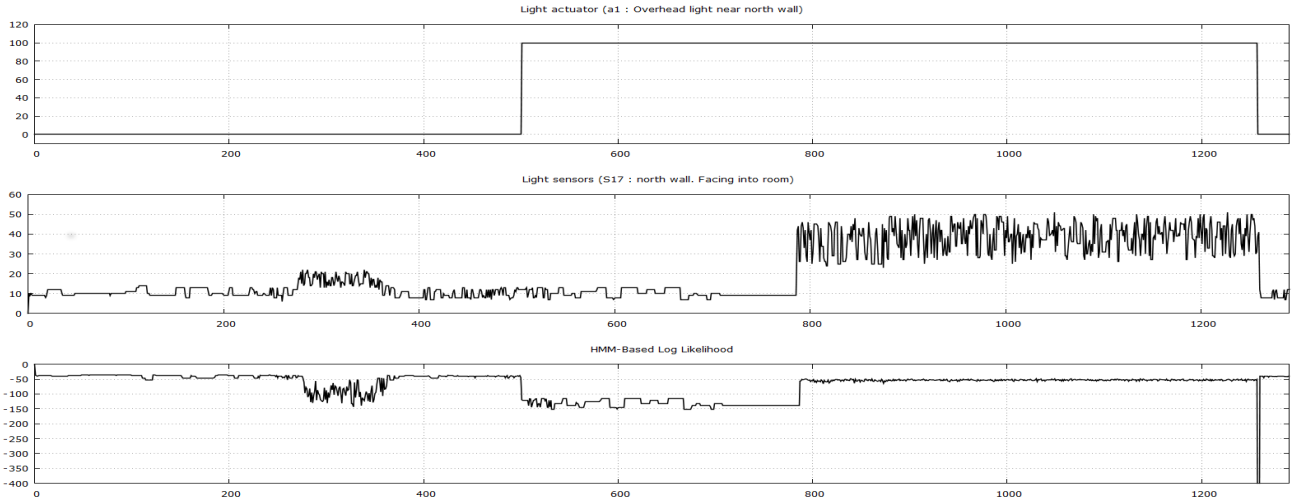

FIGURE 18. Results for the computation of the log-likelihood of an observation sequence $y_{1:K}$ (here the luminosity sensor S17 is used) given an input sequence $u_{1:K}$ and the CD-HMM state observer depicted in Fig. 11.

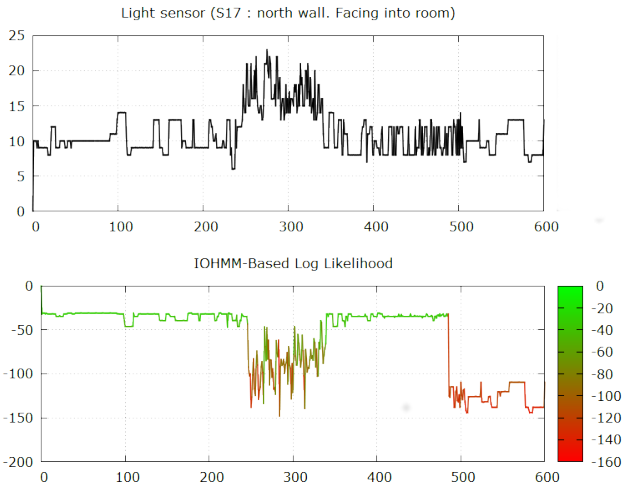

FIGURE 19. The CD-IOHMM-based state observer approach, through the log-likelihood ($]-\infty;0]$) estimation, allows to quantify, at runtime, the magnitude of the behavioral drift of a concrete application beyond a discrete PASS/FAIL result.

These results compare well with the CD-HMM approach (Section V-A) whose results for the same dataset are depicted in Fig. 18. Interestingly, the proposed approach, through the log-likelihood ($]-\infty;0]$) estimation, allows to quantify, at runtime, the magnitude of the behavioral drift of a concrete application beyond a discrete PASS/FAIL result generally obtained through most of the software engineering techniques (Fig. 19).

For the sake of clarity and ease of comprehension, the examples used in this paper are simple with regards to their dynamics and the amount of sensors and actuators used. However, one can imagine as complex as possible application behaviors modeled through ECA rules further transformed to their Moore FSM and CD-IOHMM state observer counterpart.

VII. CONCLUSION AND FUTURE WORK

Today, a growing amount of physical objects in our surroundings are connected to the Internet and provide the digital world with an interface to the physical world through sensors and actuators. At the heart of this trend, smart-* systems and applications (*e.g.* smart-home, smart-building, smart-city, smart-factory, etc…) leverage this interface to smartly and seamlessly assist individuals in their everyday lives. However, as soon as these systems and applications come to interact with the physical world by means of actuators, this promising trend is seriously hampered by the intrinsic stochastic nature of the physical world, subject to uncertainties that cannot be modeled entirely off-line. As a consequence, most of the current applications delegate to the users the responsibility of the actions to be undertaken. This leads users to be burdened with a continuous flow of information and notifications.

We argue that being subject to uncertainties and physically interacting with the users, what is at stake in this context is no longer to verify whether the expected behavior of an application is going to be maintained over time (which is illusory without a comprehensive model of the physical environment), but, more realistically, to estimate, at runtime, the gap between the observed concrete behavior and the expected ideal behavior, specified in term of effects the concrete application is expected to produce within the physical environment as it executes. In this context, we presented in this paper an approach providing smart-* systems and applications with this estimation. (1) We modeled the effects an application is expected to produce within the physical environment through the Moore Finite State Machine (Moore FSM) modeling framework. This modeling framework is very convenient and largely used for representing rule-based or event-based behaviors; (2) the Moore FSM modeling framework doesn't allow to compute the conformity of the behavior of an application beyond a discrete PASS/FAIL

result. Therefore, we considered this modeling framework from a probabilistic point of view and presented a framework for projecting this model to its associated Continuous Density Input/Output Hidden Markov Model (CD-IOHMM) state observer. We then leveraged the ability of this probabilistic modeling framework to estimate the log-likelihood of an observation sequence $y_{1:K}$ to have been produced by the model given an input sequence $u_{1:K}$. As such, the log-likelihood value gives direct insight into the behavioral drift of the concrete application as it executes.

We validated our approach through a concrete dataset in the field of smart-home. The results obtained demonstrate the soundness and the efficiency of the proposed approach for estimating the gap between the concrete and the expected behaviors of smart-* systems and applications at runtime in the presence of unforeseen environmental disturbances.

In view of these results, one can envision to use this estimation for supporting a decision-making algorithm. Also, in order to extend the application areas of the proposed approach, we plan to investigate cases where the expected behaviors have to integrate temporal properties. For instance, we target situations where the expected behaviors could be subject to some temporal inertia (e.g., (1) in order to reduce energy consumption, recent light bulbs reach their optimal luminosity after a given period of time; (2) a user requiring ambient temperature to be increased cannot expect it to be set instantaneously). On that front, we plan to specify the expected behavior of an application through timed automaton further projected into its associated Hidden Semi-Markov Model (HSMM) state observer [49]. HSMM is said semi-Markov because the transition from a state $x_{(k)}$ to a subsequent state $x_{(k+1)}$ does not only depend on the state $x_{(k)}$ but also on its duration.

Additionally, one can consider the case where the ideal expected behavior of an application is defined by the users themselves (*e.g.* through end-user programming [38]). We introduced the fact that the probabilistic modeling framework extends Moore FSM modeling framework with viability zones, *i.e.* zones where the behavior of the application is satisfactory. The boundaries of these zones are defined through the state emission and state transition probability density functions whose parameters (*e.g.* mean, variance and covariance values in the case of multivariate normal distributions), when defined by users, describe what they are able to accept in terms of effects produced within the environment. In this context, the approach proposed in this paper, by estimating the behavioral drift of a concrete application against its expected ideal behavior, is intimately linked to the *Quality of Experience* (QoE) [39] as an assessment of users' satisfaction when interacting with the controlled environment. We plan to run some experiments in this direction and correlate the state observer estimations with users' satisfaction feedback.

Although promising, the proposed approach raises a number of research challenges that will need to be addressed:

1) So far, it is assumed that the sensors required for evaluating the drift of an application are available. However, the cost/benefit analysis of the sensors required to enable the implementation of the proposed approach should be carried out (the number of sensors required is dependent on the applications and might not be economically viable from case to case). As the number of sensors in our surroundings continues to grow, a possible approach for mitigating this concern would be to leverage them opportunistically (through sensor discovery and semantic selection mechanisms for instance),
2) Also, it might be worthwhile sharing knowledge across different state observers, and thereby allowing them to possibly use more relevant sensors or respond appropriately with fewer sensors,
3) As it relies on sensors via possibly compromised network paths, the proposed approach is not immune to cyber security risks, one of the main concerns of the IoT research community [50].

ACKNOWLEDGMENT

The authors would like to thank Prof. Nhan Le Thanh and the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

REFERENCES

[1] J. Wen, J. Cao, and X. Liu, "We help you watch your steps: Unobtrusive alertness system for pedestrian mobile phone users," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Louis, MO, USA, Mar. 2015, pp. 105–113. [Online]. Available: http://dx.doi.org/10.1109/PERCOM.2015.7146516

[2] T. Nguyen, V. Nguyen, F. D. Salim, and D. Q. Phung, "SECC: Simultaneous extraction of context and community from pervasive signals," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Sydney, Australia, Mar. 2016, pp. 1–9. [Online]. Available: http://dx.doi.org/10.1109/PERCOM.2016.7456501

[3] D. Riboni, C. Bettini, G. Civitarese, Z. H. Janjua, and R. Helaoui, "Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Louis, MO, USA, Mar. 2015, pp. 149–154. [Online]. Available: http://dx.doi.org/10.1109/PERCOM.2015.7146521

[4] M. Jain, A. Singh, and V. Chandan, "Non-intrusive estimation and prediction of residential AC energy consumption," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Sydney, Australia, Mar. 2016, pp. 1–9. [Online]. Available: http://dx.doi.org/10.1109/PERCOM.2016.7456509

[5] J. Nehmer, M. Becker, A. I. Karshmer, and R. Lamm, "Living assistance systems: An ambient intelligence approach," in *Proc. 28th Int. Conf. Softw. Eng. (ICSE)*, Shanghai, China, May 2006, pp. 43–50. [Online]. Available: http://doi.acm.org/10.1145/1134293

[6] Y. Agarwal, B. Balaji, R. E. Gupta, J. Lyles, M. Wei, and T. Weng, "Occupancy-driven energy management for smart building automation," in *Proc. 2nd ACM Workshop Embedded Sensing Syst. Energy-Efficiency Buildings (BuildSys)*, Zurich, Switzerland, Nov. 2010, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/1878431.1878433

[7] G. Shahzad, H. Yang, A. W. Ahmad, and C. Lee, "Energy-efficient intelligent street lighting system using traffic-adaptive control," *IEEE Sensors J.*, vol. 16, no. 13, pp. 5397–5405, Sep. 2016.

[8] D. Garlan, "Software engineering in an uncertain world," in *Proc. FSE/SDP Workshop Future Softw. Eng. Res.*, 2010, pp. 125–128.

[9] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding uncertainty in cyber-physical systems: A conceptual model," in *Proc. Eur. Conf. Modelling Found. Appl.*, Vienna, Austria, Jul. 2016, pp. 247–264. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-42061-5_16

[10] L. I. L. Gonzalez and O. Amft, "Mining relations and physical grouping of building-embedded sensors and actuators," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, St. Louis, MO, USA, Mar. 2015, pp. 2–10. [Online]. Available: http://dx.doi.org/10.1109/PERCOM.2015.7146503

[11] G. Bovet, A. Ridi, and J. Hennebert, "Toward Web enhanced building automation systems," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Cham, Switzerland: Springer, 2014, pp. 259–283. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-05029-4_11

[12] H. Giese *et al.*, *Living With Uncertainty in the Age of Runtime Models* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, Nov. 2011, pp. 47–100. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08915-7_3

[13] M. Z. Kwiatkowska, "Advances in quantitative verification for ubiquitous computing," in *Proc. 10th Int. Colloq. Theor. Aspects Comput. (ICTAC)*, Shanghai, China, Sep. 2013, pp. 42–58. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39718-9_3

[14] M. Kwiatkowska, "From software verification to 'everyware' verification," *Comput. Sci. Res. Develop.*, vol. 28, no. 4, pp. 295–310, 2013.

[15] T. Bures *et al.*, "Software engineering for smart cyber-physical systems–towards a research agenda: Report on the first international workshop on software engineering for smart CPS," *ACM SIGSOFT Softw. Eng. Notes*, vol. 40, no. 6, pp. 28–32, 2015.

[16] *Schloss Dagstuhl: Ambient Notification Environments*, accessed on Feb. 2, 2017. [Online]. Available: http://www.dagstuhl.de/17161

[17] *Light Control Systems, fox Domotics*, accessed on Feb. 17, 2017. [Online]. Available: http://www.foxdomotics.com/light-control-mobile-apps.html

[18] F. Wagner, R. Schmuki, T. Wagner, and P. Wolstenholme, *Modeling Software With Finite State Machines: A Practical Approach*. Boca Raton, FL, USA: CRC Press, 2006.

[19] L. R. Rabiner, "Readings in speech recognition," in *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann, 1990, pp. 267–296. [Online]. Available: http://dl.acm.org/citation.cfm?id=108235.108253

[20] G. Rocher, J.-Y. Tigli, and S. Lavirotte, "On the behavioral drift estimation of ubiquitous computing systems in partially known environments," in *Proc. 13th Int. Conf. Mobile Ubiquitous Syst. Comput. Netw. Services (MOBIQUITOUS)*, New York, NY, USA, 2016, pp. 264–273. [Online]. Available: http://doi.acm.org/10.1145/2994374.2994398

[21] Y. Bengio and P. Frasconi, "Input-output HMMs for sequence processing," *IEEE Trans. Neural Netw.*, vol. 7, no. 5, pp. 1231–1249, May 1996.

[22] G. M. Youngblood and D. J. Cook, "Data mining for hierarchical model creation," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 4, pp. 561–572, Jul. 2007. [Online]. Available: http://dx.doi.org/10.1109/TSMCC.2007.897341

[23] European Commission. (2013). *Lighting the Cities*, accessed on Sep. 3, 2016. [Online]. Available: http://cordis.europa.eu/fp7/ict/photonics/docs/ssl-cip/lighting-the-cities en.pdf

[24] University of Florida. *Landscape Plants*, accessed on Sep. 3, 2016. [Online]. Available: http://hort.ifas.ufl.edu/woody/street-lights.shtml

[25] M. Pinto, R. Almeida, P. Pinho, and L. Lemos, "Daylighting in classrooms—The daylight factor as a performance criterion," in *Proc. ICEH-3rd Int. Congr. Environ. Health*, 2014. [Online]. Available: http://hdl.handle.net/10400.19/2410

[26] *Illuminating Engineering Society*, accessed on Sep. 3, 2016. [Online]. Available: http://www.iesna.org/

[27] *International Energy Agency*, accessed on Sep. 3, 2016. [Online]. Available: http://www.iea.org/topics/energyefficiency/subtopics/lighting/

[28] T. de Bruin-Hordijk and E. de Groor. TU Delft. Delft, The Netherlands. (2012). "Lighting in Schools Climate Design/Building Physics, Faculty of Architecture." accessed on Sep. 2016. [Online]. Available: http://lightinglab.fi/IEAAnnex45/publications/Technical_reports/lightin%g_in_schools.pdf

[29] R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, "Software engineering for self-adaptive systems: Assurances (dagstuhl seminar 13511)," *Dagstuhl Rep.*, vol. 3, no. 12, pp. 67–96, 2013. [Online]. Available: http://dx.doi.org/10.4230/DagRep.3.12.67

[30] D. Weyns *et al.*, "Perpetual assurances in self-adaptive systems," in *Proc. Assurances Self-Adaptive Syst. Dagstuhl Semin.*, 2014, p. 13511.

[31] N. Bencomo, "Quantun: Quantification of uncertainty for the reassessment of requirements," in *Proc. 23rd IEEE Int. Requirements Eng. Conf.*, Ottawa, ON, Canada, Aug. 2015, pp. 236–240. [Online]. Available: http://dx.doi.org/10.1109/RE.2015.7320429

[32] N. Bencomo and A. Belaggoun, "A world full of surprises: Bayesian theory of surprise to quantify degrees of uncertainty," in *Proc. 36th Int. Conf. Softw. Eng. (ICSE)*, Hyderabad, India, May 2014, pp. 460–463. [Online]. Available: http://doi.acm.org/10.1145/2591062.2591118

[33] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2523813

[34] G. I. Webb, R. Hyde, H. Cao, H. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining Knowl. Discovery*, vol. 30, no. 4, pp. 964–994, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10618-015-0448-4

[35] J.-P. Aubin, A. Bayen, and P. Saint-Pierre, *Viability Theory: New Directions*. Berlin, Germany: Springer, 2011. [Online]. Available: https://hal.inria.fr/inria-00636570

[36] G. Tamura *et al.*, "Towards practical runtime verification and validation of self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems*. Wadern, Germany: Dagstuhl Castle, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35813-5_5.

[37] S. D. Stoller *et al.*, "Runtime verification with state estimation," in *Proc. 2nd Int. Conf. Runtime Verification (RV)*, San Francisco, CA, USA, Sep. 2011, pp. 193–207. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29860-8_15

[38] J. Coutaz, S. Caffiau, A. Demeure, and J. L. Crowley, "Early lessons from the development of SPOK, an end-user development environment for smart homes," in *Proc. Conf. Ubiquitous Comput. (UbiComp)*, Seattle, WA, USA, Sep. 2014, pp. 895–902. [Online]. Available: http://doi.acm.org/10.1145/2638728.2641559

[39] K. Mitra, A. B. Zaslavsky, and C. Åhlund, "QoE modelling, measurement and prediction: A review," *CoRR*, vol. abs/1410.6952, Jun. 2014. [Online]. Available: http://arxiv.org/abs/1410.6952

[40] K. Engelbrecht, F. Gödde, F. Hartard, H. Ketabdar, and S. Möller, "Modeling user satisfaction with hidden Markov model," in *Proc. 10th Annu. Meet. Special Interest Group Discourse Dialogue Conf. (SIGDIAL)*, London, U.K., Sep. 2009, pp. 170–177. [Online]. Available: http://www.aclweb.org/anthology/W09-39260

[41] V. Krishnamurthy, *Partially Observed Markov Decision Processes*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[42] J. P. Hughes, P. Guttorp, and S. P. Charles, "A non-homogeneous hidden Markov model for precipitation occurrence," *J. Roy. Statist. Soc. C (Appl. Statist.)*, vol. 48, no. 1, pp. 15–30, 1999. [Online]. Available: http://www.jstor.org/stable/2680815

[43] P. Frasconi, *Input/Output HMMs: A Recurrent Bayesian Network View*. London, U.K.: Springer, 1997, pp. 63–79. [Online]. Available: http://dx.doi.org/10.1007/978-1-4471-0951-8_4

[44] J.-Y. Tigli *et al.*, "Wcomp middleware for ubiquitous computing: Aspects and composite event-based Web services," *Ann. Telecommun. Annal. Télécommun.*, vol. 64, no. 3, pp. 197–214, 2009. [Online]. Available: http://dx.doi.org/10.1007/s12243-008-0081-y

[45] *Thingspeak, the Open IOT Platform With MATLAB Analytics*, accessed on Feb. 17, 2017. [Online]. Available: https://thingspeak.com/

[46] P. Zimmer, *Fizzim, the Free FSM Design Tool*, accessed on Feb. 17, 2017. [Online]. Available: http://www.fizzim.com/

[47] C. R. Souza. (Dec. 2014). *The Accord.Net Framework*. [Online]. Available: http://accord-framework.net

[48] V. Hourdin, J.-Y. Tigli, S. Lavirotte, G. Rey, and M. Riveill, "SLCA, composite services for ubiquitous computing," in *Proc. Int. Conf. Mobile Technol. Appl. Syst.*, 2008, p. 11:1–11:8. [Online]. Available: http://doi.acm.org/10.1145/1506270.1506284

[49] S.-Z. Yu, "Hidden semi-Markov models," *Artif. Intell.*, vol. 174, no. 2, pp. 215–243, Feb. 2010.

[50] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Sep. 2015.

GÉRALD ROCHER received the M.S. degree in electronics and computer engineering from the Grenoble Institute of Technology, France, in 1995, and the M.S. degree in computer science from the University Côte d'Azur (UCA), France, in 2015. He is currently pursuing the Ph.D. degree in computer science with UCA. He is currently a Research Engineer with the Innovation Group, GFI Informatique, Saint-Ouen, France. His research interests include stochastic modeling and management of software applications that operate within physical environments.

JEAN-YVES TIGLI received the M.S. degree from the School of Computer Sciences, University of Nice-Sophia Antipolis, France, and the Ph.D. degree in computer science from the University of Nice-Sophia Antipolis in 1991 and 1996, respectively. He was a Postdoctoral Fellow with the Ecole des Mines de Paris and CNRS, I3S Laboratory. He contributed to two European Projects (MAST and NARVAL), focusing on software architecture for intelligent and autonomous robots. Since 1998, he has been an Associate Professor of Computer Science with the University Côte d'Azur, France. He has been the Scientific Leader of various national and international research projects (e.g., CONTINUUM Project on seamless services for users based on software adaptation, funded by the National Research Agency (2008–2012), and UBIFLOOD with the National University of Singapore in a common research program). He has broad expertise in ubiquitous computing, Internet of Things, Cyber Physical Systems, algorithms and architectures for opportunistic and dynamic software, and service composition. His research interests include stochastic modeling and management of software that physically interacts with environments and semantic service matching as a means to improve interoperability in Semantic Web of Things. In 2002, Prof. Tigli received a grant from Microsoft Research to develop WComp, a middleware for ubiquitous computing being currently transferred in various industries (Healthcare, Ambient Assisted Living, and Factory 4.0).

STÉPHANE LAVIROTTE received the M.S. degree from the Engineering School of Computer Sciences, University of Nice-Sophia Antipolis, France, in 1995, and the Ph.D. degree in computer science from the University of Nice-Sophia Antipolis and the National Institute for Computer Science and Applied Mathematics in 2000. In 2001, he was a Postdoctoral Fellow with CNRS, I3S Laboratory, France, and contributed to two European projects: Trial Solution and OpenMath Thematic Network. Since 2001, he has been an Associate Professor of Computer Science with the University Côte d'Azur, France. He co-led the CONTINUUM Project on seamless services for users based on software adaptation, funded by the National Research Agency (2008–2012) and contributed to numerous national and international projects. He is one of the main contributors of WComp, a middleware for ubiquitous computing. He is involved in several European projects and collaborates with various universities and international companies. His research interests include ubiquitous computing and ambient intelligence.