HAL
open science

Probabilistic Models Towards Controlling Smart-* Environments

Gérald Rocher, Jean-Yves Tigli, Stéphane Lavirotte

To cite this version:

Gérald Rocher, Jean-Yves Tigli, Stéphane Lavirotte. Probabilistic Models Towards Controlling Smart-* Environments. 2017. hal-01515088v1

HAL Id: hal-01515088

https://hal.science/hal-01515088v1

Preprint submitted on 27 Apr 2017 (v1), last revised 13 Jul 2017 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Probabilistic Models Towards Controlling Smart-* Environments

Gérald Rocher[1,2], Jean-Yves Tigli[2] and Stéphane Lavirotte[2]
[1]GFI Informatique, Groupe Innovation, Saint-Ouen, France,
[2]Université Côte d'Azur, CNRS, Laboratoire I3S, UMR 7271, France.

Email: gerald.rocher@gfi.fr, jean-yves.tigli@unice.fr, stephane.lavirotte@unice.fr

[Pre-print version]

***Abstract*—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 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, as opposed to the classical distributed software applications operating 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 their behavior 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 to specify the *ideal behavior* a smart-* application is supposed to meet in terms of the *effects* it is expected to produce within the physical environment. 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 allows to compute the probability of the observed effects to match the expected ones, *i.e.* it gives direct insight into the behavioral drift of the concrete application. We validate our approach on a real dataset. 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 to support a decision-making algorithm (*e.g.*, within a self-adaptive system).**

***Index Terms*—Ubiquitous computing, Ambient Intelligence, Smart-* systems and applications, Uncertainties, Probabilistic modeling, Input/Output Hidden Markov Model.**

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 [45] to context recognition [31], health-care [35] and energy analytics [25], just to name a few.
2) **Automation systems and applications**. Beside 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 [30], managing resources efficiently hereby drastically reducing the economical and ecological impacts [6][37]. 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 [20]. 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 [49]. 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 [22]:

```
rule 'LightOn'
when
  Item Presence_office changed from 0 to 100
then
  sendCommand(Light_office_dimmer, 200)
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 the background whose role is the 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* [20] (a defective light bulb, a newly added furniture conceals the light bulb, etc. . .).

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 [21] and lack the integration of stochastic dynamics that would be necessary in the context of smart-* systems and applications operating in the physical environment [28]. Therefore, although efficient for critical systems whose environments are whether known or at least controlled over time [27], 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 [12]. A direct consequence of this limitation is that 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 [4] (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 [20], 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 to specify the effects an application is expected to produce within the environment as it executes. Here, the point is that

Fig. 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 [4][3].

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 to match the expected ones 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 to model the ideal behavior an application must meet in terms of the effects it is expected to produce within the environment as it executes [43] (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 [43].
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 is expected to produce 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 [34] (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 [36] we proposed to model the state observer through the *Continuous Density Hidden Markov Model* (CD-HMM)

[Pre-print version]

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 [10] 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, given its parameters, 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 [47] (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 using this estimate to support a decision-making algorithm, enabling applications to smartly react to unexpected environmental events.

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 [13], 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 lighting in the streets, street lights in smart-cities being connected, they can be monitored and managed wirelessly [37]. Adaptive 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 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).

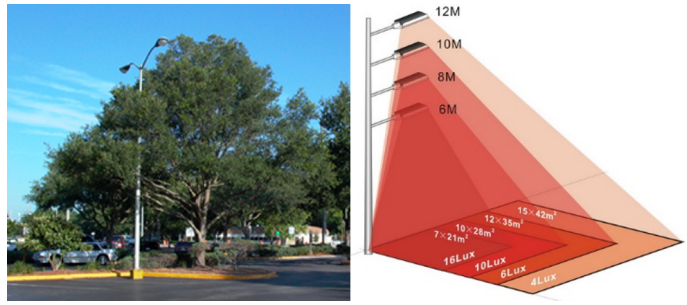

Fig. 2: The tree conflicts with the street light and jeopardizes the model [32]

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 [33]. Consequently, architects rely on standards (*e.g.* Illuminating Engineering Society of North America [1]) and physical models to design classrooms satisfying luminosity requirements while 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"[2].
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 drift unexpectedly over time, creates a serious limitation in their capability to safely act on the behalf of the users. This

[Pre-print version]

Fig. 3: The luminosity in the classrooms is not homogeneously distributed [15]

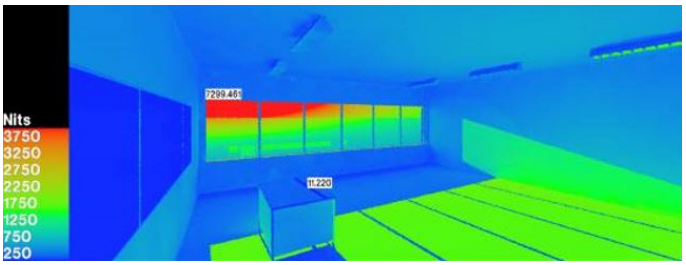

Fig. 4: The distribution of the luminosity in a classroom based on a 3D model [15]

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 [16], [46]. Close to the problem addressed in the present paper, the authors in [8] 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** [9]. Design-time *beliefs* for specific decisions are specified using *Bayesian Dynamic Decision Networks* (DDNs) [9]. 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** [19]. 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 [44] 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** [7] 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 [41]. 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 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 [40], 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 [14]). In this context, the estimation of the behavioral drift of these applications is intimately linked into the **Quality of Experience** (QoE) [29] 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 to measure the QoE. In [17], 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 through the Moore Finite State Machine modeling framework (Moore FSM)[43]. The expected ideal behavior is defined in terms of effects an application is expected to produce within the physical environment as it executes. Moore FSM models systems dynamics

[Pre-print version]

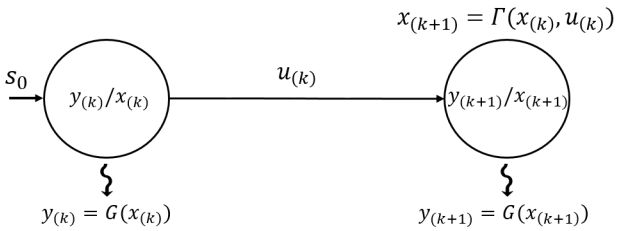

Fig. 5: Moore Finite State Machine (FSM) in a nutshell

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:

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

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 [43]. For instance, let's consider the two following BAS behavioral rules:

```
rule 'LightOn'
when
  Item A1 changed from 0 to 100
then
  sendCommand(Light_office_dimmer, 200)
end

rule 'LightOff'
when
  Item A1 changed from 100 to 0
then
  sendCommand(Light_office_dimmer, 10)
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

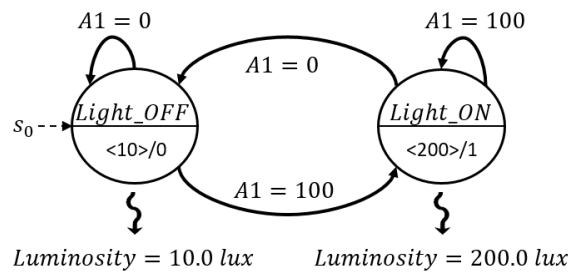

Fig. 6: BAS rules transformed to equivalent Moore FSM

is convenient at 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 gathered values,
2) Secondly, output values are idealized values. However, it is unlikely that, for a given state, the observed output 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 [12].

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 [26]:

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

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

[Pre-print version]

[Pre-print version]

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) [26]. 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.

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* [40].
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

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

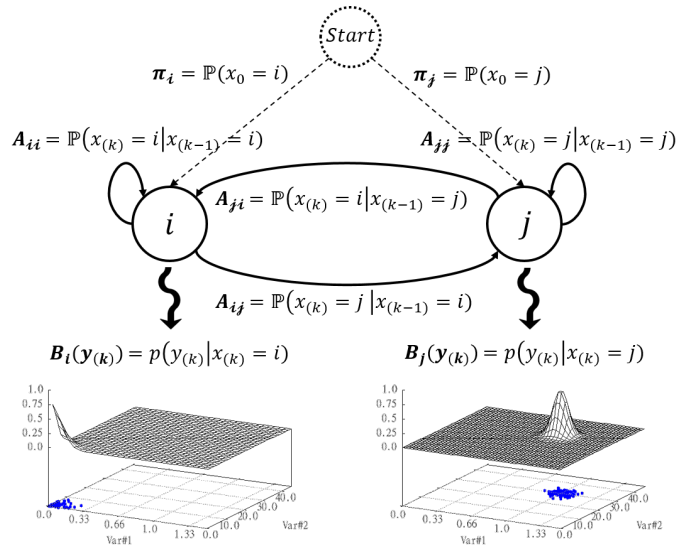

Fig. 7: Multivariate CD-HMM in a nutshell

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 $\widehat{\Theta} = \left\langle \widehat{A}, \widehat{B}, \widehat{\pi} \right\rangle$. 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 capability of the Moore FSM modeling framework for specifying, in an intuitive manner, the

behavioral rules and the expected effects an application must meet as it executes, (2) the capability of the HMM modeling framework for reasoning about the conformity of the behavior of an application as it executes.

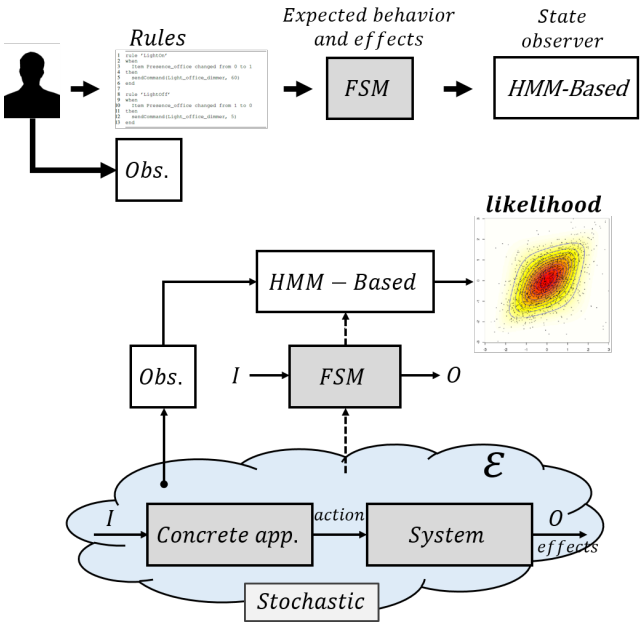

Fig. 8: Approach overview. The model of the ideal expected behavior of an application is specified in terms of the effects it is expected to produce within the physical environment as it executes (Moore FSM). This model is further projected into its probabilistic HMM-Based state observer counterpart.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

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

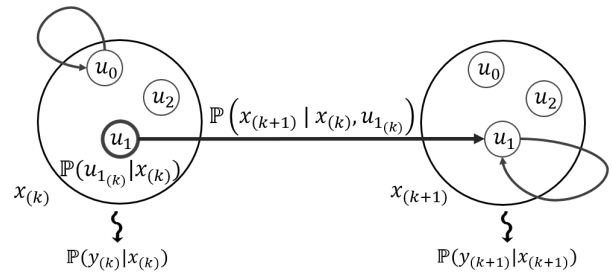

Fig. 9: Moore FSM from a probabilistic point of view. Being in the state $x_{(k)}$ there is a probability of the occurrence of the input $u_{1_{(k)}} \in I$ ($\mathbb{P}(u_{1_{(k)}}|x_{(k)})$). Then $\mathbb{P}(x_{k+1}|x_{(k)}, u_{1_{(k)}})$ is the probability of the transition $x_{(k)} \rightarrow x_{(k+1)}$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

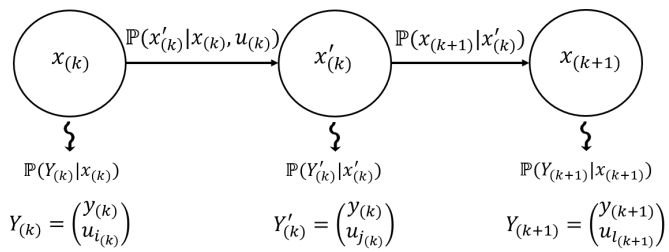

Fig. 10: Considering Moore FSM from a probabilistic point of view leads (1) a transient state $x'_{(k)}$ to appear in the HMM state observer and (2) the output vector $y_{(k)}$ to be partitioned with the input vector $u_{(k)}$ ($Y_{(k)} = (y_{(k)}, u_{(k)})$).

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.

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

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

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

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

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

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:

[Pre-print version]

$\forall x_{(k)} \in S:$

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

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

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 have 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).

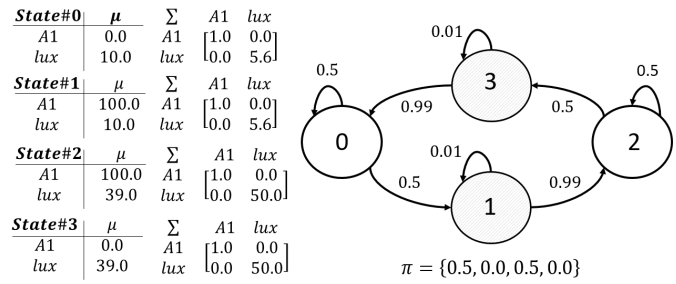

Fig. 11: CD-HMM state observer corresponding to the Moore FSM depicted in Fig.6. We assume that the probability density functions in the matrix B are multivariate normal distributions. Here, the parameter values of the distributions have been populated from the dataset used for the validation of the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

This approach, although showing good results at estimating the behavioral drift of applications [36] has several drawbacks: (1) it requires transient states to be added to take into account input events, (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

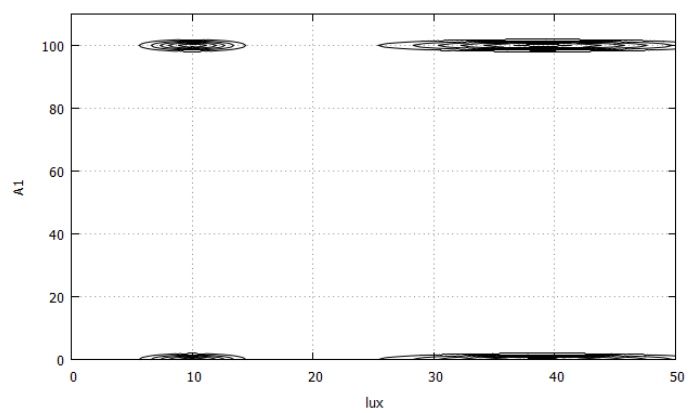

Fig. 12: The parameters of the emission probability functions (matrix B), expressed as multivariate normal distributions, define viability zones, *i.e.* zones where the effects of an application within the physical environment are satisfactory without necessarily being perfect. This figure represents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

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 [7], respectively controls that are not identified (and thus not part of the model) or disturbances under 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 [10] is an HMM for which the state transition and emission probabilities can be conditionally dependent on an input sequence (for this reason this model can also be found in the literature as Non-Homogeneous HMM (NHMM)

[Pre-print version]

[Pre-print version]

[24]). In this paper, we consider *input-independent state-emitting* model variant [18], *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 of the input $u_{(k)} \in I$ of triggering the state transition $x_{(k)} \rightarrow x_{(k+1)}$. Hence:

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

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* [40].

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

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

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

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

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:

```
rule 'LightOn'
when
  Item A1 changed from 0 to 100
  measureState(A1, Sensor_1)
then
  sendCommand(Light_office_dimmer, 200)
  measureValue(Light_office_dimmer, Sensor_2)
end
```

VI. Evaluation

A. Dataset

To evaluate our approach, we experimented with the MavPad published datasets from the MavHome (Managing an Adaptive Versatile Home) project [47]. MavPad is an on-campus apartment at the University of Texas at Arlington (TX,

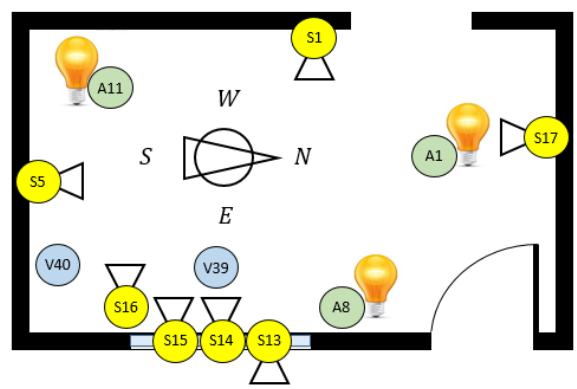

Fig. 13: MavPad living room actuators and sensors location

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

TABLE I: MavPad living room actuators and sensors description

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

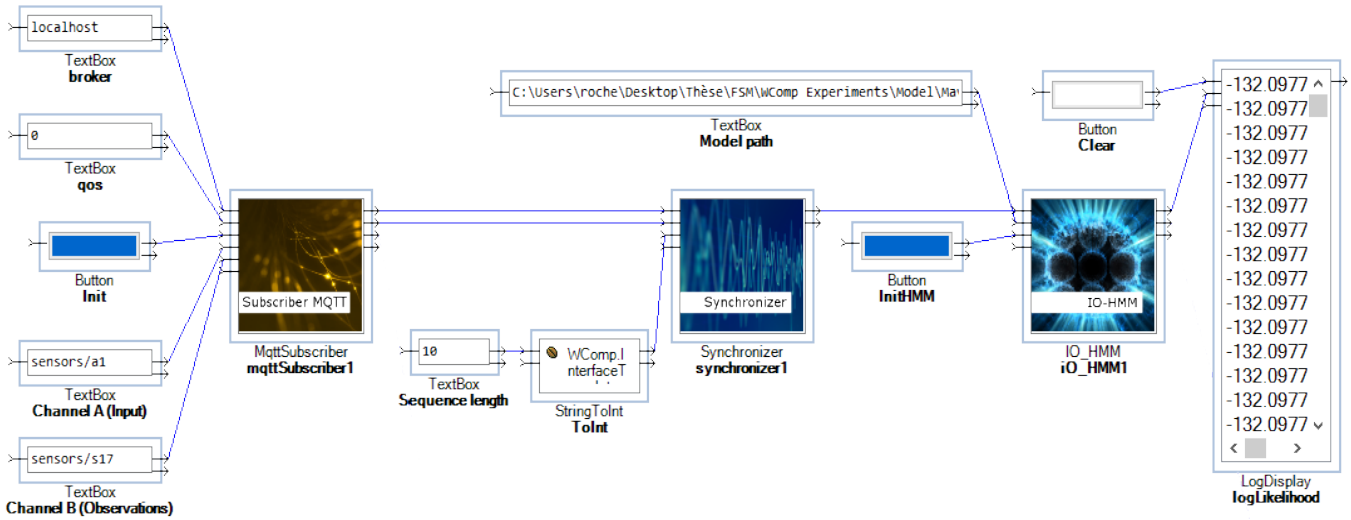

Fig. 14: Data acquisition and treatment chain implemented as lightweight components instantiated within WComp [42], a middleware for ubiquitous computing. To fit as much as possible with a real scenario, each measure from the dataset is transmitted through MQTT, a lightweight publish/subscribe protocol 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 [5].

[Pre-print version]

status. For the evaluation we focused on the lighting actuators and sensors in the living/dining room (Fig.13, Fig15) described in Table.I.

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

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 [50]. 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 [39]. 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 [42] allowing service composition by assembling lightweight components. It implements the SLCA model (Lightweight Service Component Architecture) [23] 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).

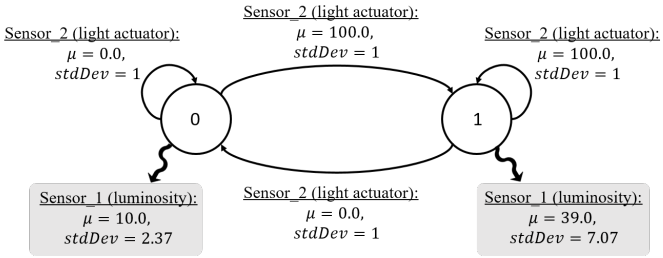

Fig. 16: CD-IOHMM state observer corresponding to the Moore FSM depicted in Fig.6. One sensor is used to observe the input event (light actuator A1) and one sensor is used to observe expected emissions for each state (luminosity sensor S17). Probability density functions are defined as univariate normal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

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

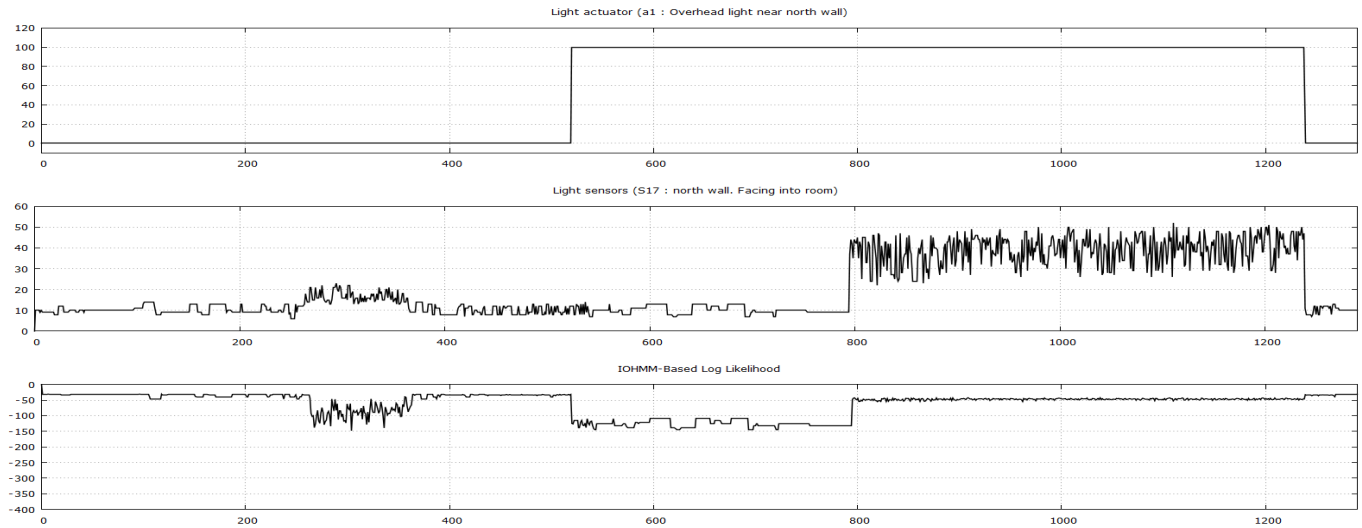

Fig. 17: Results for the computation of the log-likelihood of an observation sequence $y_{1:K}$ (here the luminosity sensor S17 is used) given an input sequence $u_{1:K}$ and the CD-IOHMM state observer depicted in Fig.16.

[Pre-print version]

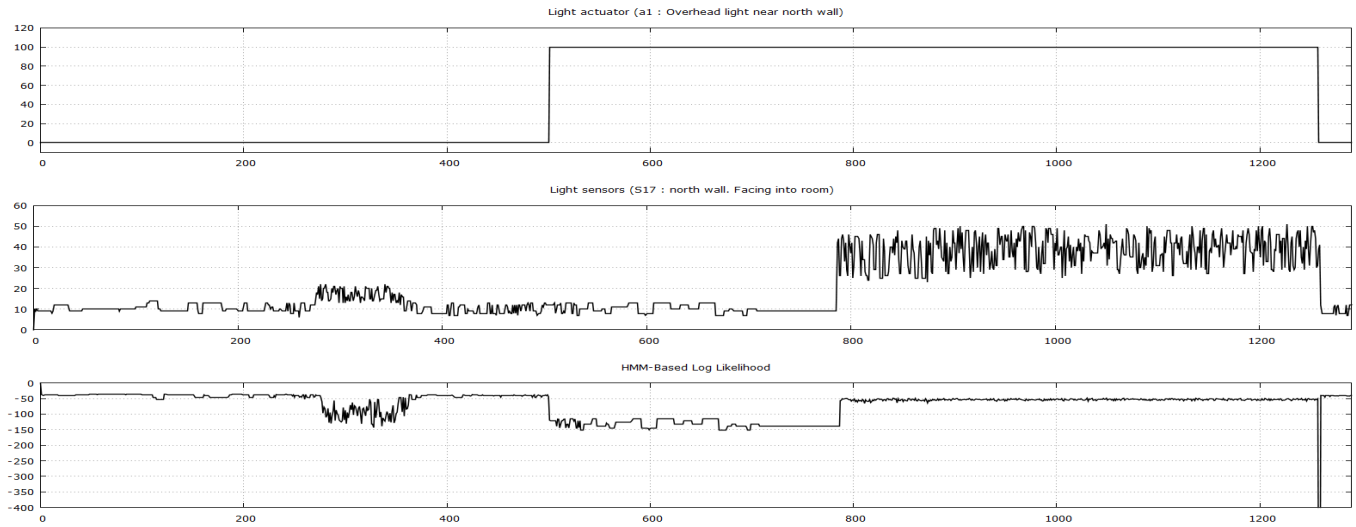

Fig. 18: Results for the computation of the log-likelihood of an observation sequence $y_{1:K}$ (here the luminosity sensor S17 is used) given an input sequence $u_{1:K}$ and the CD-HMM state observer depicted in Fig.11.

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.

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. 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

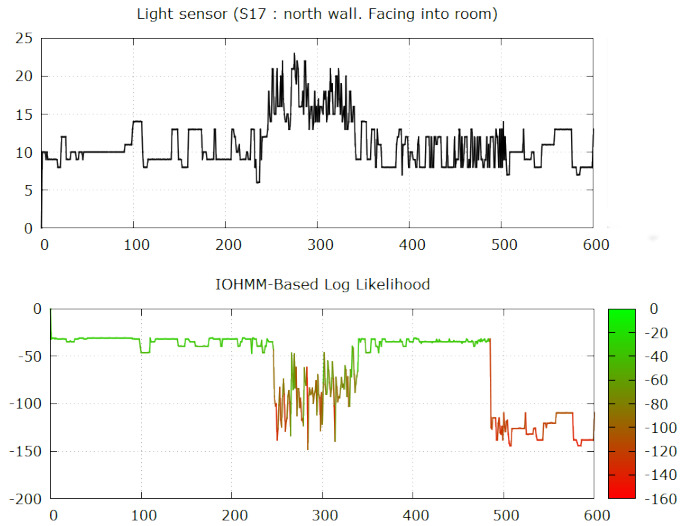

Fig. 19: The CD-IOHMM-based state observer approach, through the log-likelihood ($]-\infty;0]$) estimation, allows to quantify, at runtime, the magnitude of the behavioral drift of a concrete application beyond a discrete PASS/FAIL result.

[Pre-print version]

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 (Fig.19).

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, we envision using this estimation to support a decision-making algorithm within a self-adaptive system. 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 automata further projected into its Hidden Semi-Markov Model (HSMM) state observer [48]. 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 [14]). 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) [29] 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 [38].

VIII. Acknowledgment

This work was supported by GFI Informatique, Innovation group, Saint-Ouen, France. 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] Illuminating engineering society. http://www.iesna.org/ [Online; accessed 3-Sept-2016].

[2] International energy agency. http://www.iea.org/topics/energyefficiency/subtopics/lighting/ [Online; accessed 3-Sept-2016].

[3] Light control systems, fox domotics. http://www.foxdomotics.com/light-control-mobile-apps.html [Online; accessed 17-Feb-2017].

[4] Schloss dagstuhl: Ambient notification environments. http://www.dagstuhl.de/17161 [Online; accessed 2-Feb-2017].

[5] Thingspeak, the open iot platform with matlab analytics. https://thingspeak.com/ [Online; accessed 17-Feb-2017].

[6] Yuvraj Agarwal, Bharathan Balaji, Rajesh E. Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *BuildSys'10, Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, Zurich, Switzerland, November 3-5, 2010*, pages 1–6, 2010.

[7] Jean-Pierre Aubin, Alexandre Bayen, and Patrick Saint-Pierre. *Viability Theory: New Directions*. Springer, 2011.

[8] Nelly Bencomo. Quantun: Quantification of uncertainty for the reassessment of requirements. In *23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*, pages 236–240, 2015.

[9] Nelly Bencomo and Amel Belaggoun. A world full of surprises: bayesian theory of surprise to quantify degrees of uncertainty. In *36th International Conference on Software Engineering, ICSE '14, Companion Proceedings, Hyderabad, India, May 31 - June 07, 2014*, pages 460–463, 2014.

[10] Yoshua Bengio and Paolo Frasconi. Input/output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7:1231–1249, 1996.

[11] Gérôme Bovet, Antonio Ridi, and Jean Hennebert. Toward web enhanced building automation systems. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 259–283. 2014.

[12] Tomas Bures, Danny Weyns, Christian Berger, Stefan Biffl, Marian Daun, Thomas Gabor, David Garlan, Ilias Gerostathopoulos, Christine Julien, Filip Krikava, 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 Software Engineering Notes*, 40(6):28–32, 2015.

[13] European commission. Lighting the cities, 2013. http://cordis.europa.eu/fp7/ict/photonics/docs/ssl-cip/lighting-the-cities_en.pdf [Online; accessed 3-Sept-2016].

[14] Joëlle Coutaz, Sybille Caffiau, Alexandre Demeure, and James L. Crowley. Early lessons from the development of spok, an end-user development environment for smart homes. In *The 2014 ACM Conference on Ubiquitous Computing, UbiComp '14 Adjunct, Seattle, WA, USA - September 13 - 17, 2014*, pages 895–902, 2014.

[15] Truus de Bruin-Hordijk and Ellie de Groor. Lighting in schools. Climate Design/Building Physics, Faculty of Architecture, TU Delft, The Netherlands, 2012. http://lightinglab.fi/IEAAnnex45/publications/Technical_reports/lighting_in_schools.pdf [Online; accessed 3-Sept-2016].

[16] Rogério de Lemos, David Garlan, Carlo Ghezzi, and Holger Giese. Software engineering for self-adaptive systems: Assurances (dagstuhl seminar 13511). *Dagstuhl Reports*, 3(12):67–96, 2013.

[17] Klaus-Peter Engelbrecht, Florian Gödde, Felix Hartard, Hamed Ketabdar, and Sebastian Möller. Modeling user satisfaction with hidden markov models. In *Proceedings of the SIGDIAL 2009 Conference, The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 11-12 September 2009, London, UK*, pages 170–177, 2009.

[18] Paolo Frasconi. *Input/Output HMMs: A Recurrent Bayesian Network View*, pages 63–79. Springer London, London, 1997.

[19] João Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[20] David Garlan. Software engineering in an uncertain world. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 125–128. ACM, 2010.

[21] Holger Giese, Nelly Bencomo, Liliana Pasquale, Andres J. Ramirez, Paola Inverardi, Sebastian Wätzoldt, and Siobhán Clarke. Living with uncertainty in the age of runtime models. In *Models@run.time - Foundations, Applications, and Roadmaps [Dagstuhl Seminar 11481, November 27 - December 2, 2011]*, pages 47–100, 2011.

[22] Luis Ignacio Lopera Gonzalez and Oliver Amft. Mining relations and physical grouping of building-embedded sensors and actuators. In *2015 IEEE International Conference on Pervasive Computing and Communications, PerCom 2015, St. Louis, MO, USA, 23-27 March, 2015*, pages 2–10, 2015.

[23] Vincent Hourdin, Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, and Michel Riveill. Slca, composite services for ubiquitous computing. In *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, Mobility '08, pages 11:1–11:8. ACM, 2008.

[24] James P. Hughes, Peter Guttorp, and Stephen P. Charles. A non-homogeneous hidden markov model for precipitation occurrence. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 48(1):15–30, 1999.

[25] Milan Jain, Amarjeet Singh, and Vikas Chandan. Non-intrusive estimation and prediction of residential AC energy consumption. In *2016 IEEE International Conference on Pervasive Computing and Communications, PerCom 2016, Sydney, Australia, March 14-19, 2016*, pages 1–9, 2016.

[26] Vikram Krishnamurthy. *Partially Observed Markov Decision Processes*. Cambridge University Press, 2016.

[27] Marta Kwiatkowska. From software verification to 'everyware' verification. *Computer Science-Research and Development*, 28(4):295–310, 2013.

[28] Marta Z. Kwiatkowska. Advances in quantitative verification for ubiquitous computing. In *Theoretical Aspects of Computing - ICTAC 2013 - 10th International Colloquium, Shanghai, China, September 4-6, 2013. Proceedings*, pages 42–58, 2013.

[29] Karan Mitra, Arkady B. Zaslavsky, and Christer Åhlund. Qoe modelling, measurement and prediction: A review. *CoRR*, abs/1410.6952, 2014.

[30] Jürgen Nehmer, Martin Becker, Arthur I. Karshmer, and Rosemarie Lamm. Living assistance systems: an ambient intelligence approach. In *28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 20-28, 2006*, pages 43–50, 2006.

[31] Thuong Nguyen, Vu Nguyen, Flora Dilys Salim, and Dinh Q. Phung. SECC: simultaneous extraction of context and community from pervasive signals. In *2016 IEEE International Conference on Pervasive Computing and Communications, PerCom 2016, Sydney, Australia, March 14-19, 2016*, pages 1–9, 2016.

[32] University of Florida. Landscape plants. http://hort.ifas.ufl.edu/woody/street-lights.shtml [Online; accessed 3-Sept-2016].

[33] Manuel Pinto, Ricardo Almeida, Paulo Pinho, and Luís Lemos. Daylighting in classrooms-the daylight factor as a performance criterion. In *ICEH-3rd International Congress on Environmental Health*, 2014.

[34] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[35] Daniele Riboni, Claudio Bettini, Gabriele Civitarese, Zaffar Haider Janjua, and Rim Helaoui. Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment. In *2015 IEEE International Conference on Pervasive Computing and Communications, PerCom 2015, St. Louis, MO, USA, 23-27 March, 2015*, pages 149–154, 2015.

[36] Gérald Rocher, Jean-Yves Tigli, and Stéphane Lavirotte. On the behavioral drift estimation of ubiquitous computing systems in partially known environments. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MOBIQUITOUS 2016, pages 264–273, New York, NY, USA, 2016. ACM.

[Pre-print version]

[Pre-print version]

[37] Gul Shahzad, Heekwon Yang, Arbab Waheed Ahmad, and Chankil Lee. Energy-efficient intelligent street lighting system using traffic-adaptive control. *IEEE Sensors Journal*, 16(13):5397–5405, 2016.

[38] Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, and Alberto Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164, 2015.

[39] C. R. Souza. The accord.net framework, Dec 2014. http://accord-framework.net. São Carlos, Brazil.

[40] Scott D. Stoller, Ezio Bartocci, Justin Seyster, Radu Grosu, Klaus Havelund, Scott A. Smolka, and Erez Zadok. Runtime verification with state estimation. In *Runtime Verification - Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers*, pages 193–207, 2011.

[41] Gabriel Tamura, Norha M. Villegas, Hausi A. Müller, João Pedro Sousa, Basil Becker, Gabor Karsai, Serge Mankovski, Mauro Pezzè, Wilhelm Schäfer, Ladan Tahvildari, and Kenny Wong. Towards practical runtime verification and validation of self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, pages 108–132, 2010.

[42] Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, Vincent Hourdin, Daniel Cheung-Foo-Wo, Eric Callegari, and Michel Riveill. Wcomp middleware for ubiquitous computing: Aspects and composite event-based web services. *annals of telecommunications - annales des télécommunications*, 64(3):197–214, 2009.

[43] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, and Peter Wolstenholme. *Modeling software with finite state machines: a practical approach*. CRC Press, 2006.

[44] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai-Long Nguyen, and François Petitjean. Characterizing concept drift. *Data Min. Knowl. Discov.*, 30(4):964–994, 2016.

[45] Jiaqi Wen, Jiannong Cao, and Xuefeng Liu. We help you watch your steps: Unobtrusive alertness system for pedestrian mobile phone users. In *2015 IEEE International Conference on Pervasive Computing and Communications, PerCom 2015, St. Louis, MO, USA, 23-27 March, 2015*, pages 105–113, 2015.

[46] Danny Weyns, Nelly Bencomo, Radu Calinescu, Javier Camara, Carlo Ghezzi, Vincenzo Grassi, Larse Grunske, Paola Inverardi, Jean-Marc Jezequel, Sam Malek, et al. Perpetual assurances in self-adaptive systems. In *Assurances for Self-Adaptive Systems, Dagstuhl Seminar*, volume 13511, 2014.

[47] G. Michael Youngblood and Diane J. Cook. Data mining for hierarchical model creation. volume 37, pages 561–572. IEEE Trans. Systems, Man, and Cybernetics, Part C, 2007.

[48] Shun-Zheng Yu. Hidden semi-markov models. In *Journal of Artificial Intelligence*, volume 174, pages 215–243, 2010.

[49] Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. Understanding uncertainty in cyber-physical systems: A conceptual model. In *Modelling Foundations and Applications - 12th European Conference, ECMFA 2016, Held as Part of STAF 2016, Vienna, Austria, July 6-7, 2016, Proceedings*, pages 247–264, 2016.

[50] Paul Zimmer. Fizzim, the free fsm design tool. http://www.fizzim.com/ [Online; accessed 17-Feb-2017].

Gérald Rocher received the M.S. degree in computer science from the University Côte d'Azur (UCA), France, in 2015, and the M.S. degree in electronics and computer engineering from Grenoble Institute of Technology, France, in 1995. He is currently pursuing the Ph.D. degree in computer science at UCA. His current research interests include stochastic modeling and management of software physically interacting with environments. He is a research engineer working within the innovation group at GFI Informatique, Saint-Ouen, France.

Jean-Yves Tigli received the M.S. degree from the engineering school of computer sciences from the University of Nice-Sophia Antipolis (ESSI), France, and the Ph.D. degree in computer science from the same university in 1991 and 1996 respectively. He was a postdoctoral fellow at Ecole des Mines de Paris and at CNRS, I3S laboratory. He contributed on two European projects (MAST and NARVAL) focusing on software architecture for intelligent and autonomous robots. Since 1998, he is Associate Professor of computer science at University Côte d'Azur (UCA), France (formerly University of Nice-Sophia Antipolis). He has been the scientific leader of various national and international research projects (*i.e.* CONTINUUM project on seamless services for users based on software adaptation, funded by the National Research Agency (2008-2012), UBIFLOOD with National University of Singapore in a common research program,. . . just to name a few). In 2002, he received a grant from Microsoft Research to develop WComp, a middleware for ubiquitous computing being currently transferred in various industries (Healthcare, Ambient Assisted Living, Factory 4.0, . . .). He has broad expertise in ubiquitous computing, Internet of Things (IoT), Cyber Physical Systems (CPS), algorithms and architectures for opportunistic and dynamic software and service composition. His current research interests include stochastic modeling and management of software physically interacting with environments and semantic service matching as a means to improve interoperability in SWoT (Semantic Web of Things).

Stéphane Lavirotte received the M.S. degree from the engineering school of computer sciences from the University of Nice-Sophia Antipolis (ESSI), France, in 1995, and the Ph.D degree in computer science from the same university and the national institute for computer science and applied mathematics (INRIA) in 2000. In 2001, he was a postdoctoral fellow at CNRS, I3S laboratory, France, and contributed on two European projects: Trial Solution and OpenMath Thematic Network. Since then, he is Associate Professor of computer science at University Côte d'Azur (UCA), France (formerly University of Nice-Sophia Antipolis). He co-led CONTINUUM project on seamless services for users based on software adaptation, funded by the National Research Agency (2008-2012) and contributed in numerous national and international projects. His current research interests include ubiquitous computing and ambient intelligence. He is one of the main contributors of WComp, a middleware for ubiquitous computing. He is involved in several European projects and collaborate with various universities and international companies.