HAL
open science

On the Behavioral Drift Estimation of Ubiquitous Computing Systems in Partially Known Environments

Gérald Rocher, Jean-Yves Tigli, Stéphane Lavirotte

To cite this version:

Gérald Rocher, Jean-Yves Tigli, Stéphane Lavirotte. On the Behavioral Drift Estimation of Ubiquitous Computing Systems in Partially Known Environments. 13th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Nov 2016, Hiroshima, Japan. hal-01362500

HAL Id: hal-01362500

https://hal.science/hal-01362500v1

Submitted on 13 Sep 2016

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.

On the Behavioral Drift Estimation of Ubiquitous Computing Systems in Partially Known Environments

Gérald Rocher[1,2]
gerald.rocher@gfi.fr

Jean-Yves Tigli[2]
Jean-Yves.Tigli@unice.fr

Stéphane Lavirotte[2]
Stephane.Lavirotte@unice.fr

[1]GFI Informatique, Groupe Innovation, Saint-Ouen, France

[2]Université Côte d'Azur, CNRS, Laboratoire I3S, UMR 7271, France

ABSTRACT

Background. With the recent advent of the so-called *connected objects*, today largely present in our surroundings, software applications have an open door to the physical world through *sensors* and *actuators*. However, although it offers huge opportunities in many areas (*e.g.*, smart-home, smart-cities, etc...), it poses a serious methodological challenge. Indeed, while classical software applications operate in the well known and delimited digital world, the so-called *ambient applications* operate *in* and *through* the physical world, *open* and subject to *uncertainties* that cannot be modeled accurately and entirely. These uncertainties lead the behavior of the ambient applications to potentially *drift* over time against requirements. In this paper, we propose a framework to estimate the behavioral drift of the ambient applications against requirements at runtime.

Methodology. We rely on the Moore Finite State Machines (FSM) modeling framework to specify the *ideal behavior* an ambient application is supposed to meet, irrespective of the operating environment and the underlying software infrastructure. We then appeal on the control theory and propose a framework to transform the Moore FSM to its associated Continuous Density Hidden Markov Model (CD-HMM) *state observer*. By accounting for uncertainties through probabilities, it extends Moore FSM with *viability zones*, *i.e.* zones where the behavioral requirements of the ambient applications are acceptable. The observation of the execution of a concrete ambient application together with the statistical modeling framework underlying its associated state observer allow to compute the *likelihood* of an observation sequence to have been produced by the application. The likelihood then gives direct insight into the behavioral drift of the concrete application against requirements.

Results. We validate our approach through a concrete use-case in the field of school lighting. The results demonstrate the soundness and efficiency of the proposed approach for estimating the behavioral drift of the ambient applications at runtime. In light of these results, one can envision using this estimation to support a decision-making algorithm (*e.g.*, within a self-adaptive system).

Keywords

Cyber-physical systems, ubiquitous computing, service composition, uncertainties, Hidden Markov Model, drift estimation

1. INTRODUCTION

In recent years, achievements in computer hardware miniaturization and power consumption reduction have enabled the proliferation of communicating devices integrated in everyday life physical objects (*a.k.a.* connected objects) and physical environments (*e.g.*, houses, buildings, cities, etc...). By means of software services, these so-called *ambient devices* now provide software applications with interfaces to *interact* with our physical surroundings through *sensors* and *actuators* at the heart of the Cyber-Physical Systems (CPS) [1]. The proliferation of ambient devices results in the simultaneous presence of a large number of software services offered to the users. Their *composition* allows to imagine as many applicative scenarios as relevant interconnections. The primary objective of these so-called *ambient applications* is to smartly and seamlessly assist users in their everyday lives, by off-loading them with the constraining physical and cognitive tasks [15]. This software paradigm, known as ubiquitous computing, pervasive computing or ambient computing, is at the heart of the smart-* systems (namely smart-city, smart-building, smart-home, etc...) [12].

In this context, the interaction logics between ambient devices are not hardwired and become more complex to manage. Indeed, ambient devices interact with each other within an application through the physical environment but also with the other surrounding devices and physical processes. This complexity requires ambient applications designers to ensure that the coherency and the relevancy of the interaction logics between ambient devices within the ambient applications are going to be maintained over time, irrespective of the evolutions of the environment (*stochastic dynamics*) and the interactions with the other surrounding devices (*interferences*) [7, 26]. To this end, they appeal on Model-Driven Engineering (MDE) techniques relying on: (1) models of the systems (often deterministic), their operational environments, *etc.*..., and (2) formal verification techniques. These techniques are meant to *predict* the behavioral compliancy of the applications against requirements on the long run [19] and their effectiveness depends on the accuracy of

Mobiquitous 2016, November 28–December 1, 2016, Hiroshima, JAPAN

the models or the intervention of the designers or the users on a regular basis.

However:

1. **Models are abstractions of the real world and are, by definition, incomplete**. Models flaws and runtime uncertainties [9] lead models to be continuously made consistent with the evolutions of the systems and their environments (*e.g.*, models@runtime [5]). However, these approaches assume controlled environments whose dynamics can be anticipated and modeled offline (*e.g.*, Dynamic Software Product Lines[6]) and evolutions fully observable during operation (*e.g.*, Model Identification Adaptive Control (MIAC) and Model Reference Adaptive Control (MRAC)).

2. **Models lack integration of stochastic dynamics**. Most of the modeling frameworks have discrete dynamics. Quantitative models allow, to some extent, accounting for uncertainties. However, it is still far from what would be necessary in the context of ubiquitous computing systems operating in physical environments whose dynamics is intrinsically *stochastic*[13].

Ubiquitous computing introduces a *methodological break* that challenges the classical MDE techniques. Indeed, these techniques are efficient for critical systems whose environments are whether known or at least controlled over time [14]. However, this is clearly not the case for many smart-* systems operating in partially known physical environments, subject to unpredictable changes and where indirect interactions through the physical environment may yield unexpected applications behaviors[27].

These uncertainties lead the behavior of the ambient applications to potentially *drift* over time against requirements and one need to provision ubiquitous computing systems with the ability to quantify this behavioral drift.

In this context, the main idea behind our approach is to use an *empirical statistical modeling framework* to model the behavior an ambient application must meet, irrespective of the physical environment it operates in and the underlying ambient devices it is composed with. At runtime, the *concrete* ambient application is seen as a *black box*. The *observation* of its direct and indirect impacts on the physical environment is then applied against the statistical model from which the *likelihood* of the observation is computed. The likelihood estimate gives direct insight into the behavioral deviation of the concrete application against requirements.

The contributions of this paper are the following:

1. We appeal on the control theory and the notion of *state observer* (Section.4.2). Given a system whose underlying states are not directly observable, the role of a state observer is to estimate the underlying states of the real system during execution (*a.k.a.* state estimation problem). This is typically done by means of a dynamical model of the system, the *observation* of its inputs and the indirect effects of its execution on the operational environment. In this paper, we consider an ambient application as being the system whose operational environment is the physical environment. We assume that the physical environment dynamics is *non-linear* and possibly subject to *non-Gaussian noises*. Based on these assumptions, we propose to model the state observers of the ambient applications as *Continuous Density Hidden Markov Model* (CD-HMM)[17]. We then leverage the ability of this statistical modeling framework to estimate the likelihood of an observation sequence gathered from sensors buried in the physical environment to have been produced by the model. While the proposed approach is not meant to predict the behavior of the ambient applications on the long run, it provides a mechanism for estimating their behavioral drift against requirements.

2. We model the *ideal* expected behavior of the ambient applications as *Moore Finite State Machines* (Moore FSM). As such, we use Moore FSM modeling framework as a means to specify the set of possible states and transitions in which applications are not compromised, that is, the set of states and transitions where the behavioral requirements are satisfied (Section.4.1). Moreover, Moore FSM modeling framework allows designers to specify the expected observations for each state (*i.e.*, the expected state output emissions). Then, we consider this formalism from a *probabilistic* point of view (Section.5.1) and provide a framework to transform these machines to their associated CD-HMM state observers (Section.5.2). By accounting for uncertainties through probabilities, CD-HMM extends Moore FSM with *viability zones* [2], *i.e.* zones where the behavioral requirements are satisfied, without necessarily being perfect.

3. We validate our approach through a concrete use-case in the field of school lighting (Section.2). The results demonstrate the soundness and efficiency of the proposed approach at estimating the behavioral drifts of the ambient applications at runtime in the presence of environmental disturbances (Section.6.2). In light of these results, one can envision using this estimate to support a decision-making algorithm, enabling it to react smartly to unexpected environmental events.

2. CASE STUDY: LIGHTING IN SCHOOLS

Lighting, and particularly 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[16]. Thus, architects rely on standards (*e.g.* Illuminating Engineering Society of North America (IESNA[1])) and physical models to design classrooms satisfying luminosity requirements while maximizing daylighting (Figure.2). However, classrooms occupancy and unexpected physical phenomena (Figure.1) could compromise the model (*e.g.*, due to weather conditions, stickers or drawings on the windows, furniture, etc... the luminosity is not distributed in the classroom as planned by the models). Although 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 are never switched off. This results in a huge waste of energy. According to the International Energy

[1] http://www.ies.org/

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].

Figure 1: **Examples of classrooms where the luminosity is not homogeneously distributed**

Through this simple example, we show that environmental dynamics and associated uncertainties are hardly predictable and incorporable into models designed offline. Moreover, relying on users to manage these uncertainties does not allow to ensure systems effectiveness both from users and energy savings perspectives.

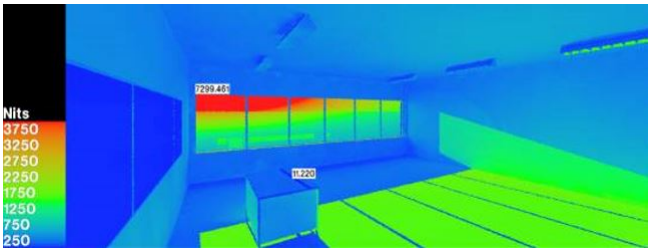


Figure 2: **Simulation of the luminance distribution within a classroom based on a 3D model**[3]

On the basis of these facts, let's imagine an ambient application whose role, in the presence of the teacher and/or students, is to maximize, on their behalf, the luminosity of a particular point in space of a classroom.

Stochastic dynamics. For instance, thanks to some semantically annotated ambient devices, the application get composed of actuators (*e.g.*, light bulbs, shutters, etc...) on which it acts to get the required luminosity at the specified point in space. From that point, numerous unexpected events may occur in the classroom and outside leading the luminosity to decrease: (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 luminosity to meet the expected level at some point in space, (4)... At any time, the behavior of the application may drift against requirements due to incomplete models (*e.g.*, semantics annotations formally describe ambient devices functionalities, irrespective of the environmental interferences which could compromise the model).

Systems multiplicity and interferences. A second ambient application runs in the same environment whose role is to minimize the lighting power consumption. Depending on the lighting power consumption, measured on a regular basis, time slots and classrooms occupancy, this application may suddenly prevent some ambient devices to be used (*e.g.*, light bulbs). By doing so, the integrity of the first application may be compromized.

Such scenarios are uncountable in the context of ubiquitous computing. The lack of comprehensive models of the systems and the physical environments dynamics calls the need for a mechanism aiming at estimating, from observations, the behavioral drifts of the ambient applications against requirements at runtime. Thereby, ubiquitous computing systems could act smartly against requirements and ambient devices availability to disqualify some ambient applications made ineffective by the evolution of the environment and deploy some others, more relevant.

3. RELATED WORKS

To the best of our knowledge, the estimation of the behavioral drift of ambient applications at runtime in the context of ubiquitous computing has not been studied before. The HMM-based technique presented in this paper to support the calculation of the likelihood that an ambient application satisfies the required behavior at runtime given observation sequences is new.

In this section, we discuss related work on *self-adaptation*, providing ubiquitous computing systems with so-called *self-** properties. Indeed, when operating in open and uncertain environments, self-adaptation becomes a must-be requirement. Self-adaptation poses new challenges in term of *assurance*, *i.e.*, the ability to provision evidence that the system satisfies its behavioral requirements, irrespective of the adaptations over time. This is witnessed by the recent research on the subject [8, 24], namely:

Runtime verification. Is a verification technique concerned by detecting at runtime, from observations, whether certain properties of a system hold. For instance, close to our work in the sense that the HMM modeling framework is used, in [21], Stoller *et al.* model a program as a HMM where the hidden states are the states of the program. The problem addressed concerns the program execution traces whose, in order to reduce the verification overhead, are often sampled by a monitor. This potentially leads some events to be missed. In this context, HMM modeling framework demonstrated good results at estimating the probability a property holds, given a potentially incomplete trace of the program execution.

Bayesian surprise. Close to the problem addressed in the present paper, in [3], Bencomo is concerned by the quantification of the deviation gap from the original specified behavior of a Self-Adaptive System (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* [4]. Design-time *beliefs* for specific decisions are specified using *Bayesian Dynamic Decision Networks* (DDNs) [4]. A Bayesian surprise then quantifies how observations affect beliefs at runtime by measuring the *distance* between posterior and prior belief distributions. The distance is calculated by using the *Kullback-Leibler divergence* (KL). This work in progress mainly focuses on the non-functional requirements. The approach we propose in the present paper addresses both functional (what an application is supposed *to do*, described with Moore FSM modeling framework) and

[2]http://www.iea.org/topics/energyefficiency/subtopics/lighting/

[3]Source:http://lightinglab.fi/IEAAnnex45/publications/Technical_reports/lighting_in_schools.pdf

non-functional requirements (how an application is supposed *to be*, specified through the expected observations).

Viability zones The *viability zone* of a system is the set of possible states in which the system operation is not compromised (here a parallel can be done with the *solutions space* of a dynamical system). In other words, this is the set of states where the system behavior is satisfied [2]. Viability zones are characterized in terms of relevant attributes and associated values measured from the system or the environment. Managing viability zones at runtime is crucial for the assurance of SAS and is still an open problem [22]. Along with Moore FSM and HMM modeling frameworks, our approach allows, to some extent, defining viability zones of an ambient application through state transition probabilities and state output emission probability density functions (pdf). Specifically, the HMM parameters can be *learnt* during operation [21], allowing the viability zones to be refined at runtime.

4. BACKGROUND

The theoritical fundations of this paper are based on two modeling frameworks, namely Moore Finite State Machines (FSM) and Hidden Markov Models (HMM). We provide hereafter a brief overview of these modeling frameworks.

4.1 Moore Finite State Machine (FSM)

In this paper we consider ambient applications whose ideal expected behavior can be modeled as Moore Finite State Machines (Moore FSM). Moore FSM models systems dynamics as *deterministic processes* where each state is associated with an output emission [23].

More formally, a discrete-time Moore FSM is defined by the tuple $M = \langle S, S_0, I, Y, \Gamma, G \rangle$ 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 outputs 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 ($y_{(k)} = G(x_{(k)})$). The outputs of a Moore FSM depend only on the underlying states. Thus, as it executes, a Moore FSM produces an observation sequence $y_{1:K} = \{y_1, y_2, \ldots, y_K\}$.

Moore FSM equations can be stated as follow (Figure.3):

$$\begin{cases} x_{(k+1)} & = \Gamma(x_{(k)}, u_{(k)}) \quad \text{(State equation)} \\ y_{(k)} & = G(x_{(k)}) \quad \text{(Observation equation)} \end{cases} \tag{4.1.1}$$

The state transition function Γ and the state space S reflect the paths an ambient application can go through as it executes. The function G defines the outputs one can expect while being in a particular state.

4.2 Hidden Markov Model (HMM)

The absence of reliable models of the underlying ambient devices the ambient applications are composed with and the physical environment they interact with, requires ambient applications to be *observed* during operation in order to quantify their potential behavioral drift against requirements. To this end, we appeal on the control theory and the notion of *state observer*. In most practical cases, a system underlying state $x_{(k)}$ cannot be completely determined by direct observations (system states are said *hidden*). Instead, a state observer is used to estimate the underlying state $\hat{x}_{(k)}$ of the real system by means of its dynamical model, the observation of its inputs and the indirect effects of its execution on the operational environment (Figure.4). In this

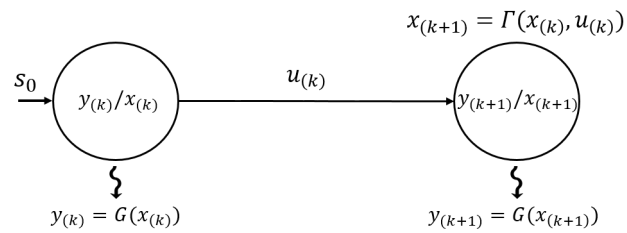


Figure 3: **Moore Finite State Machine (FSM) in a nutshell**

paper, we consider an ambient application as being the system whose operational environment is the physical environment. The physical environment dynamics is intrinsically *stochastic* and physical phenomena are most of the time *nonlinear*. Ambient applications, therefore, when operating within the physical environment through sensors and actuators, 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 ambient application buried in the physical environment, can be described by the following discrete-time stochastic dynamical model [11]:

$$\begin{cases} x_{(k+1)} & = \Phi(x_{(k)}, \omega_{(k)}) \quad \text{(State process)} \\ y_{(k)} & = \psi(x_{(k)}, v_{(k)}) \quad \text{(Observation process)} \end{cases} \tag{4.2.1}$$

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.4.2.1 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 ambient 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 Model (HMM) modeling framework [11]. 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 com-

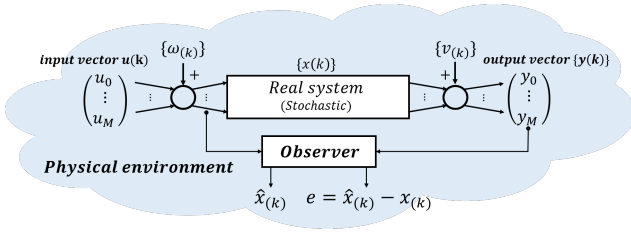


Figure 4: **Discrete-time Stochastic Dynamical System Observer**

pletely 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 x_j at time $k+1$ given we are in state x_i at time k; $\sum_{j=1}^{N} A_{ij} = 1$.
- B is the observation probability density function matrix. In this paper we consider *multivariate continuous observations*, where the emission probabilities are expressed, without loss of generality, as multivariate normal density functions.

$$B = diag(p(y_{(k)}|x_{(k)} = 1), \ldots, p(y_{(k)}|x_{(k)} = N) \quad (4.2.2)$$

Indeed, in the context of ubiquitous computing systems, states are characterized by observations inherently *multi-dimensional* and *continuous*. This type of HMM 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 observing $y_{(k)}$.

The model equations can be stated as follow (Figure.5):

$$\begin{cases} x_{(k+1)} & = \mathbb{P}(x_{(k+1)}|x_{(k)}) & \Rightarrow A_{x_{(k+1)}, x_{(k)}} \\ y_{(k)} & = p(y_{(k)}|x_{(k)}) & \Rightarrow B_{x_{(k)}}(y_{(k)}) \end{cases} \quad (4.2.3)$$

HMM is mainly used to solve the following problems:

1. **Hidden state estimation problem**. Given a CD-HMM with parameters $\Theta = \langle A, B, \pi \rangle$ and an observation sequence $y_{1:K} = \{y_1, y_2, \ldots, y_K\}$, evaluate the probability that the HMM ended in a particular state ($\mathbb{P}(x_{(K)}|y_{1:K})$). In other words, one obtains the likelihood of a given observation sequence $y_{1:K}$ to have been produced by the model. In the CD-HMM context, this computation is achieved by the classical recursive *forward algorithm* [21]. Let $\alpha_{(K)}(i) = \mathbb{P}(y_{1:K}, x_{(K)} = i)$ the probability that $x_{(K)} = i$ given the observation sequence $y_{1:K}$. Then, given

$$\alpha_{(1)}(i) = \pi_{(i)} b_{(i)}(y_{(1)}), 1 \leq i \leq N \quad (4.2.4)$$

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

$$\alpha_{(k+1)}(i) = (\sum_{j=1}^{N} \alpha_{(k)} A_{ji}) b_{(i)}(y_{(k+1)}) \quad (4.2.5)$$

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 a CD-HMM with parameters $\Theta = \langle A, B, \pi \rangle$ and an observation sequence $y_{1:K} = \{y_1, y_2, \ldots, y_K\}$, decode the most probable underlying hidden state sequence Q that has been ran through to produce $y_{1:K}$. *Viterbi algorithm* is mainly used for this purpose.

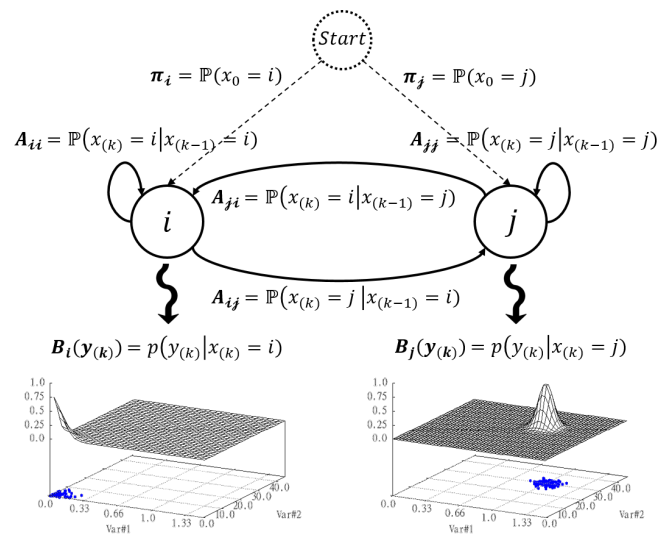


Figure 5: **Multivariate Continuous Density HMM (CD-HMM) in a nutshell**

3. **Model parameters learning**. Given an observation sequence $y_{1:K} = \{y_1, y_2, \ldots, y_K\}$, estimate the CD-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.*...).

5. PROPOSED APPROACH

In this paper, Moore FSM modeling framework is used to model the ideal expected behavior of the ambient applications, irrespective of the physical environment uncertainties and the underlying software infrastructure. As such, Moore FSM specifies the set of possible states and transitions where the behavioral requirements are satisfied, along with the expected observations for each state. A step further, CD-HMM modeling framework is used to model the corresponding state observers. CD-HMM adds a probabilistic dimension on both state observations and state transitions to account for uncertainties and thereby extends Moore

FSM with viability zones [2], *i.e.* zones where the behavioral requirements are satisfied, without necessarily being perfect (Figure.11). Therefore, Moore FSM dynamics $\subset$ CD-HMM dynamics. The problem is then, given an ambient application whose ideal expected behavior is specified through a Moore FSM, to transform this model to its associated CD-HMM state observer further used to estimate the behavioral drift of the concrete application against requirements from observations.

5.1 Moore FSM: A Probabilistic point of view

The state transition function Γ maps a state $x_{(k)} \in S$ and an input vector $u_{(k)} \in I$ to a next state $x_{(k+1)} \in S$. This can be traduced by (1) the probability of the input $u_{(k)}$ occurrence and (2) the probability that this occurrence leads effectively the state transition (joint probability). The output function G maps each state to an output vector. Here again,

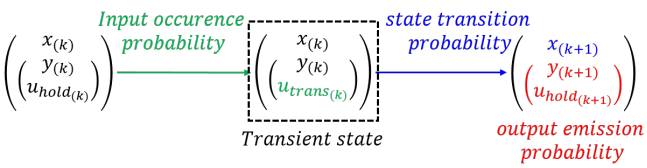


Figure 6: **DFSA from a probabilistic point of view.** *$u_{hold_{(k)}}$ is the input leading the state to loop back on itself, and $u_{trans_{(k)}}$ is the input leading a transition from the state $x_{(k)}$ to the state $x_{(k+1)}$*

this can be traduced by the probability of observing an output vector given the system is in a particular state (Figure6). Taking into account the input occurrence probability leads the apparition of a *transient* state where the output vector $y_{(k)}$ is partitioned with the input vector ($Y_{(k)} = (y_{(k)}, u_{(k)})$). It means that inputs have to be observed as part of the state output vector. For a given Moore FSM, the probabilities discussed above are implicit and values constrained as follow:

1. **Input occurrence probability**. Moore FSM doesn't define an input occurrence function. The occurrence of an input is *exogenous* to the model. So, from a probabilistic point of view, given a state $x_{(k)} \in S$ any input $u_{(k)} \in I$ may occur at time k. This can be formulated as follow:

$$\forall x_{(k)} \in S, \sum_{u_{(k)}=u_1}^{u_M} \mathbb{P}_{occurrence}(x_{(k)}, u_{(k)}) = 1 \qquad (5.1.1)$$

2. **State transition probability**. The state transition function Γ maps a state $x_{(k)} \in S$ and an input $u_{(k)} \in I$ to the next state $x_{(k+1)}$ ($x_{(k+1)} = \Gamma(x_{(k)}, u_{(k)})$). From a probabilistic standpoint, the Moore FSM implicit constraint on state transitions can be stated as follow:
$\forall x_{(k)} \in S, u_{(k)} \in I$:

$$\exists! x_{(k+1)} \in S \backslash \mathbb{P}_{trans}(x_{(k)}, x_{(k+1)} | u_{(k)}) = 1, \qquad (5.1.2a)$$

$$\sum_{x_{(k+1)}=x_1}^{x_N} \mathbb{P}_{trans}(x_{(k)}, x_{(k+1)} | u_{(k)}) = 1, \qquad (5.1.2b)$$

$$\sum_{x_{(k+1)}=x_1}^{x_N} \mathbb{P}_{trans}(x_{(k)}, x_{(k+1)}) = 1 \qquad (5.1.2c)$$

Eq.5.1.2a is the probability for a transition to occur from the state $x_{(k)} \in S$ at time k to the state $x_{(k+1)} \in S$ at time $k+1$, given the occurrence of the input $u_{(k)} \in I$ at time k (joint probability). Eq.5.1.2a and Eq.5.1.2b put together stipulate that a couple $x_{(k)}, u_{(k)}$ at time k is associated with a unique state $x_{(k+1)}$ at time $k+1$ (determinism). This constraint can be represented by a tri-dimensional matrix $N \times N \times L$. Interestingly, the probabilistic approach describes all state transitions, even those not explicitly described in the model (*i.e.*, state transitions with probability set to 0 in the $N \times N \times L$ matrix are implicitly not accepted by the model). Eq.5.1.2c stipulates that state outgoing transitions are equiprobable. Indeed, inputs being exogenous to the model, one cannot presume the probability of the inputs occurrence to each other.

3. **Output emission probability**. Each state output vector is given by $y_{(k)} = G(x_{(k)})$. From a probabilistic standpoint, the Moore FSM implicit constraints on outputs can be stated as follow:
$\forall x_{(k)} \in S$:

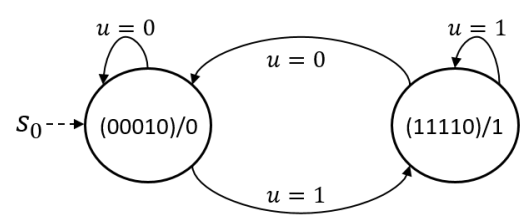
$$\exists! y_{(k)} \in Y \backslash \mathbb{P}_{emission}(y_{(k)} | x_{(k)}) = 1, \qquad (5.1.3a)$$

$$\sum_{y_{(k)}=y_1}^{y_N} \mathbb{P}_{emission}(y_{(k)} | x_{(k)}) = 1, \qquad (5.1.3b)$$

Eq.5.1.3a is the probability to emit $y_{(k)} \in Y$ given we are in the state $x_{(k)} \in S$ at time k. Eq.5.1.3a and Eq.5.1.3b put together stipulate that each state is associated with a unique output vector.

5.2 Moore FSM to CD-HMM projection

Based on the case-study described in section 2, let's consider an ambient application whose required behavior is defined by the Moore FSM depicted in Figure.7. The model describes the ideal behavior of the ambient application that can be stated as follow:
"While a human is present in the classroom, the luminosity of a point in space in the classroom must be maintained at 30 lux. Otherwise, the luminosity must be maintained at 2 lux".



input $u_{(k)}$	Low luminosity (State 0)	High luminosity (State 1)
No presence ($u = 0$)	Conform	Non-conform
Presence ($u = 1$)	Non-conform	Conform

Figure 7: **Moore FSM Example**. *The luminosity of a point in space must be set to 30 lux if human presence is detected, 2 lux otherwise*

This behavior can be modeled with two states: (1) while being in the first state, *i.e.*, no human presence is detected ($u = 0$), the expected luminosity is 2 lux (low luminosity,

represented by the output vector (00010)); (2) while being in the second state, *i.e.*, human presence is detected ($u = 1$), the expected luminosity is 30 lux (high luminosity, represented by the output vector (11110)). Let's transform these requirements to a CD-HMM state observer (defined by $\Theta = \langle A, B, \pi \rangle$).

Construction of the state transition matrix (A). We consider the FSM model depicted in Figure.7 from a probabilistic point of view. First, we only consider the state transitions probabilities and apply the constraints given by Eqs.(5.1.2a, 5.1.2b and 5.1.2c). The resulting model is given in Figure.8. For the time being, the state transitions are defined irrespective of the input. Without input information, the non-conform situations described in Figure.7 (*i.e.*, (1) human presence is detected but the luminosity remains at a low level, (2) no human is detected but the luminosity is at a high level) cannot be detected because both states (*i.e.*, low and high luminosity) are defined by the model. Although both states are defined by the model, they are conditioned by an input value. So, let's now consider the input occurrences probabilities as depicted in Figure.6. The resulting

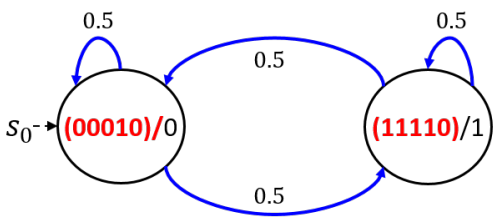


Figure 8: **Moore FSM from a probabilistic point of view taking into account only the state transitions probabilities**

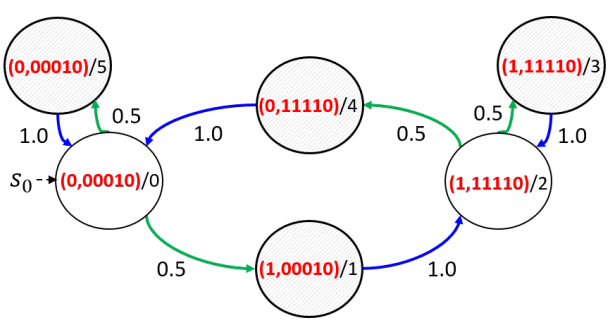


Figure 9: **Moore FSM from a probabilistic point of view taking into account both state transitions and inputs occurrence probabilities**

model is given in Figure.9. Transient states denote the fact that state transitions given in Figure.8 are conditioned by an input occurrence represented by the transient states in Figure.6.

Note that in Figure.9, the state transitions from states $1 \rightarrow 2$ and $4 \rightarrow 0$ are supposed to be instantaneous. Depending on the observation sampling rate, there is a non-negligible probability for a transient state to loop back on itself (*i.e.*, the observation values lead CD-HMM to consider that the application is still in the same state due to the observation probability density functions defined in matrix B). One need to integrate this probability in the model, computed based on the sensor data acquisition sampling rate. Also, some states being equivalent (states 2/3 and 0/5), a model reduction can be applied. The resulting model is given in Figure.10.

Construction of the observation probabilities (B). Recall the definition of the matix B given by 4.2.2 where $p(y_{(k)}|x_{(k)})$ are probability density functions expressed, without loss of generality, as multivariate normal density functions. Based on Figure.6, the observation vector is given by

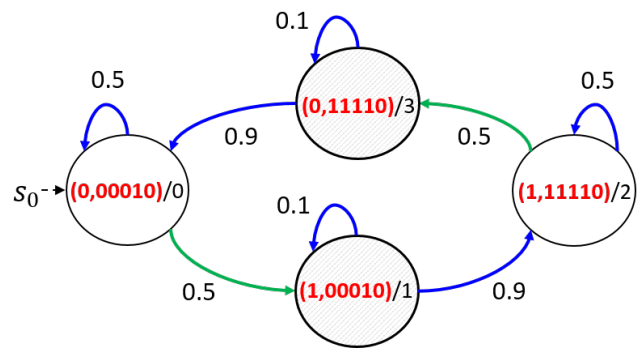


Figure 10: **Moore FSM from a probabilistic point of view after states reduction**

$Y_{(k)} = (y_{(k)}, u_{(k)})$, where $y_{(k)}$ is the luminosity sensor value $\in \mathbb{N}$ and $u_{(k)}$ the human presence sensor value $\in [0, 1]$. Thus for each $x_{(k)}$:

$$b_{x_{(k)}}(Y_{(k)}) = p(Y_{(k)}|x_{(k)}) \tag{5.2.1a}$$

$$= \frac{1}{\sqrt{(2\pi)^n}} \frac{1}{\sqrt{det\Sigma}} e^{\frac{1}{2}(Y_{(k)}-\mu)'\Sigma^{-1}(Y_{(k)}-\mu)} \tag{5.2.1b}$$

where $n = 2$ ($Y_{(k)}$ comprises two random variables, $y_{(k)}$ and $u_{(k)}$), $\mu = \mathbb{E}(Y_{(k)})$ an n-dimensional vector and Σ is the $n \times n$ positive definite variance-covariance matrix of $Y_{(k)}$. μ can directly be retrieved from the Moore FSM model output vectors. Variance and covariance values cannot be retrieved from the Moore FSM model directly and have to be defined offline by the designer of the ambient application.

Construction of the initial state distribution vector (π). The initial state distribution vector π is initialized with equiprobable values ($\pi_i = \frac{1}{N}, 1 \leq i \leq N$). Without setting equiprobable values, the CD-HMM initial state distribution vector obtained from the Moore FSM depicted in Figure.10 would be $\pi = (\pi_0 = 1.0, \pi_1 = 0.0, \pi_2 = 0.0, \pi_3 = 0.0)$. Then, based on Eqs.(4.2.4 and 4.2.5), any observation sequence whose initial state is not x_0 would yield $\mathbb{P}(x_{(K)}|y_{1:K}) \approx 0.0$. Actually, one do not know the initial state of an observation sequence gathered at any time from sensors. In other words, the starting state of an observation sequence can be any state $x \in S$.

CD-HMM through parameters $\Theta = \langle A, B, \pi \rangle$ adds a probabilistic dimension on both state emissions and state transitions to account for uncertainties and thereby extends Moore FSM with viability zones [2], *i.e.* zones where the behavioral requirements of the ambient application are satisfied, without necessarily being perfect (*e.g.*, the Figure.11 depicts viability zones defined from the observation probability density functions defined in matrix B).

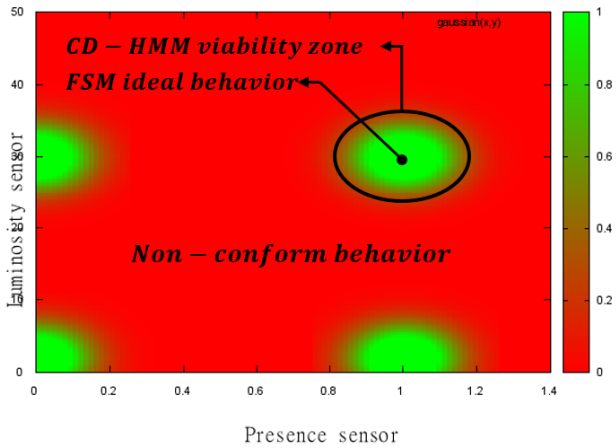


Figure 11: **The CD-HMM observation matrix B defines viability zones through pdf**

6. EVALUATION

6.1 Evaluation Methodology

The following methodology has been used to evaluate the validity of our approach:

1. **Specify the ideal expected behavior of an ambient application as a Moore FSM**. To this end, the finite state machine designer tool Qfsm[4] has been used.

2. **Simulate the Moore FSM to obtain a set of traces**. From the model, we developed Verilog test benches to produce a set of traces as Value Change Dump (VCD) files (Figure.12). The length of the generated traces is limited to 350 events.

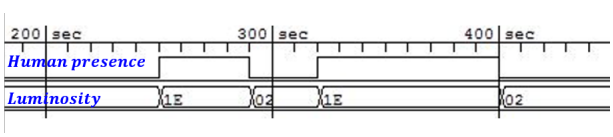


Figure 12: **Example of a Moore FSM simulation waveforms**

3. **Apply an additive random noise to the set of traces**. We wrote a program that reads the set of traces obtained from the simulation (.vcd) and transforms it to a Comma-Separated Values (.csv) dataset. A user defined additive pseudo-random noise uniformly distributed has been added to the traces.

4. **Transform the Moore FSM to its associated CD-HMM state observer**, defined by $\Theta = \langle A, B, \pi \rangle$. We implemented the algorithm described in section 5.2 to transform the Moore FSM model to its associated CD-HMM state observer. We used Accord.NET framework [20] to implement the CD-HMM engine.

5. **Verify the CD-HMM state observer dynamics**. From the CD-HMM parameters $\Theta = \langle A, B, \pi \rangle$, we generated a probable observation sequence $\widehat{Y_{1:K}}$ (Figure.13). This step is necessary to ensure, offline, that the observer dynamics is conform to the behavioral requirements of the ambient application. Some parameters are then adjusted as necessary (variance-covariance matrix values and transient states probabilities given the inputs acquisition sampling rate).

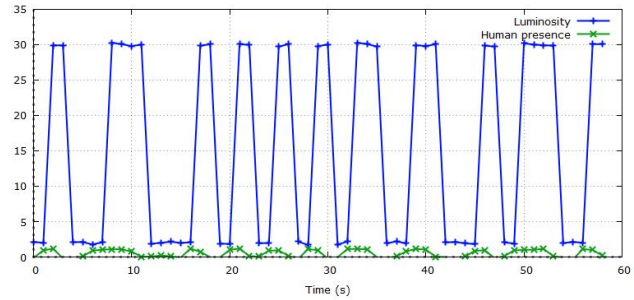


Figure 13: **Observation sequence generated from the CD-HMM state observer depicted in Figure.10**

6. **Apply the set of traces as input to the CD-HMM state observer**. Given a CD-HMM state observer with parameters $\Theta = \langle A, B, \pi \rangle$ and an observation sequence $Y_{1:K} = \{Y_1, Y_2, \ldots, Y_K\}$, one get the probability that the CD-HMM ended in a particular state ($\mathbb{P}(x_{(K)}|y_{1:K})$). In other words, one obtains the likelihood of a given input sequence $Y_{1:K}$ to have been produced by the model $\Theta = \langle A, B, \pi \rangle$.

6.2 Experimental Results

We applied the aforementioned methodology to the use-case described in the Figure.7. We used MQTT (Message Queuing Telemetry Transport) [10], a lightweight messaging protocol, to transmit each event of the set of traces to the CD-HMM state observer on a regular basis (we set the emission rate to one event per second). We added a pseudo-random noise uniformly distributed between 0 and 0.2 to the traces. Then, we set the observation window of the state observer to 10 events ($Y_{1:K} = \{Y_1, Y_2, \ldots, Y_K\}$, where $K = 10$ and $Y_{(k)} = (y_{(k)}, u_{(k)})$, where $y_{(k)}$ is the luminosity sensor value $\in \mathbb{N}$ and $u_{(k)}$ the human presence sensor value $\in [0, 1]$. The CD-HMM state observer is generated with a probability for transient states to loop back on themselves set to 0.1. μ values (Eq.5.2.1) are set from the Moore FSM output vectors for each state while the variance values of the matrix Σ are set to 0.02 and the covariance set to 0.0 (We consider independent variables).
We modified the set of traces to randomly introduce some violations: (1) the luminosity in the classroom is degraded by some stochastic dynamics, (2) a second system prevents the utilization of some devices (systems multiplicity and interferences).
(1) Stochastic dynamics. In the first case, we simulated a situation where, while the observed ambient application is running, the weather turns cloudy. The drop in luminosity is detected by the luminosity sensor. The behavior of the ambient application starts drifting against requirements (blue curve in Figure.14). As the CD-HMM state observer receives the observation sequence $Y_{1:K} = \{Y_1, Y_2, \ldots, Y_K\}$ where $K = 10$, it detects the drift and returns negative log-likelihood values (red curve).

[4]urlhttp://qfsm.sourceforge.net/

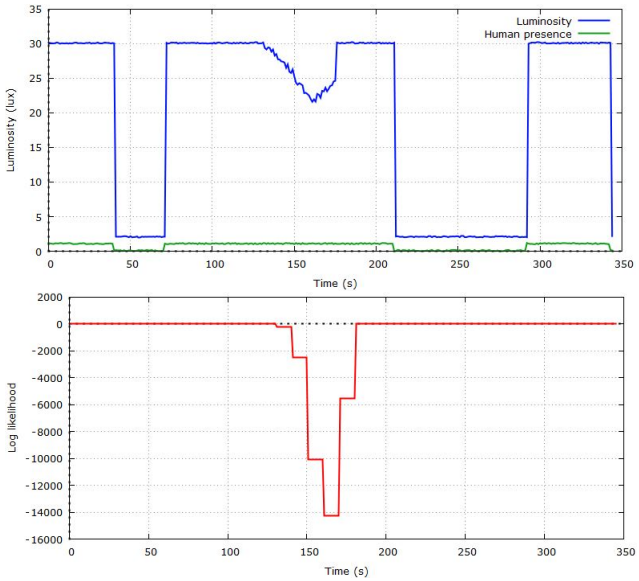


Figure 14: **While the application is running, the weather turns cloudy. This leads a luminosity drop**

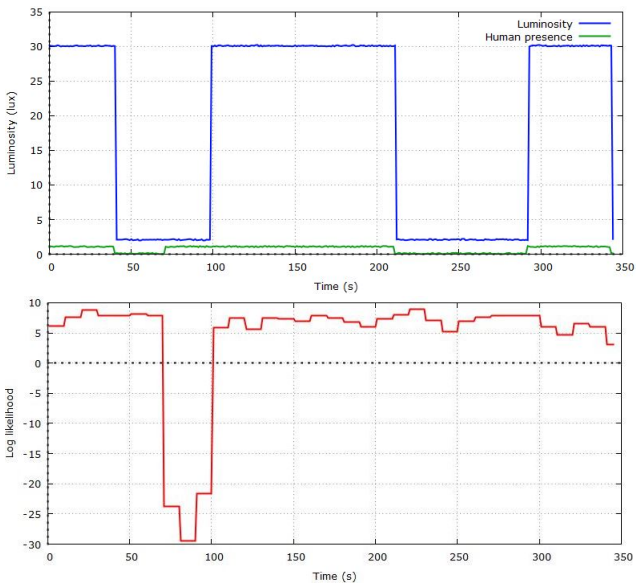


Figure 15: **Although a human is detected in the classroom, the luminosity remains at a low level**

(2) Systems multiplicity and interferences. In the second case we simulated a situation where, although a human presence is detected in the classroom (green curve in Figure.15), the luminosity remains at a low level (blue curve). For instance, the ambient devices used in the first situation (some light bulbs) are disqualified by a second application managing the lighting power consumption. Here again, the non-conformity of the behavior of the ambient application against requirements is detected by the CD-HMM state observer (red curve).

The magnitude of the quantification of the behavioral drift against the required behavior depends on the HMM parameters $\Theta = \langle A, B, \pi \rangle$. Typically, if we compare the behavioral violations previously depicted and associated drift quantifications, in the first case (stochastic dynamics) the behavior is not expected at all by the model while in the second case (Systems multiplicity and interferences), the behavior is expected but with low probability.

7. CONCLUSION AND FUTURE WORK

The proliferation of ambient devices in our physical surroundings (*a.k.a.* connected objects) results in the simultaneous presence of a large number of software services offered to the users. Their composition yields the so-called ambient applications whose primary objective is to smartly and seamlessly assist users in their everyday lives, by offloading them with the constraining physical and cognitive tasks. However, this vision, at the heart of *smart-** systems, is seriously hampered by the physical nature of the operational environment the ambient applications operate in. Indeed, ambient devices interact in and through the physical environment whose nature is intrinsically stochastic and subject to uncertainties that cannot be modeled entirely offline. Hence, apart from the case where the operational environment is controlled or its future evolutions are known, any unforeseen environmental change (*i.e.*, not modeled) can jeopardize the models and lead the behavior of the ambient applications to drift over time.

In this context, we have presented an approach providing smart-* systems with the ability to quantify, at runtime, the behavioral drift of the ambient applications against requirements. The main idea of the proposed approach is twofold: (1) We modeled the ideal behavior of the ambient applications as deterministic Moore Finite State Machines (Moore FSM). This modeling framework allows ambient applications designers to specify: (i) the set of possible states and state transitions in which the behavior of the ambient applications is not compromised and (ii) the expected observations for each state. (2) We considered the Moore FSM modeling framework from a statistical point of view, and presented a framework to transform Moore FSM models to their associated Continuous Density Hidden Markov Model (CD-HMM) state observers. As such, HMM modeling framework allows to take into account unertainties and extends Moore FSM with *viability zones* in which the behavior of the ambient applications is not compromised. We then leveraged the ability of this modeling framework to estimate the *likelihood* of an observation sequence gathered from sensors buried in the physical environment to have been produced by the model (*i.e.*, the quantification of the behavioral drift against requirements).

We validated our approach through a concrete use-case in the field of school lighting. The results obtained demonstrate the soundness and efficiency of the proposed approach for estimating the behavioral drifts of the ambient applications at runtime in the presence of environmental disturbances. In light of these results, we envision using this estimation to support a decision-making algorithm within a self-adaptive system.

To extend the application areas of the proposed approach, we plan to investigate cases where behavioral requirements have to integrate *temporal properties*. For instance, we target situations where ambient devices interactions could be subject to some temporal inertia (*e.g.*, (1) in order to reduce energy consumption, recent light bulbs reach their op-

timal luminosity after a given period of time; (2) a user requiring ambient temperature to be increased cannot expect this requirement to be instantaneously executed). On that front, we plan to specify the required behaviors of the ambient applications as discrete-time *Finite State Time Machine* (FSTM) [18] further transformed to *Hidden Semi-Markov Model* (HSMM) state observer [25]. 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.

Finally, the lack of experimental datasets covering the functional and non-functional behavioral aspects of the ambient applications, makes difficult the benchmarking of different approaches. To fill the gap, we plan to gather datasets from multiple fields ranging from assisted living to industry 4.0.

8. ACKNOWLEDGMENTS

This research was supported by GFI Informatique, Innovation Group.

9. REFERENCES

[1] R. Alur. *Principles of cyber-physical systems*. MIT Press, 2015.

[2] J.-P. Aubin, A. Bayen, and P. Saint-Pierre. *Viability Theory: New Directions*. Springer, 2011.

[3] N. Bencomo. Quantun: Quantification of uncertainty for the reassessment of requirements. In *23rd International Requirements Engineering Conference (RE)*, pages 236–240. IEEE, 2015.

[4] N. Bencomo and A. Belaggoun. A world full of surprises: Bayesian theory of surprise to quantify degrees of uncertainty. In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 460–463. ACM, 2014.

[5] N. Bencomo, R. France, B. Cheng, and U. Aßmann. *Models@runtime: foundations, applications, and roadmaps*, volume 8378. Springer, 2014.

[6] R. Capilla, J. Bosch, P. Trinidad, A. Ruiz-Cortès, and A. Hinchey. An overview of dynamic software product line architectures and techniques: Observations from research and industry. *Journal of Systems and Software*, 91:3 – 23, 2014.

[7] W. Chipman, C. Grimm, and C. Radojicic. Coverage of uncertainties in cyber-physical systems. *GMM-Fachbericht-ZuE*, 2015.

[8] R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese. Software engineering for self-adaptive systems: Assurances (dagstuhl seminar 13511). *Dagstuhl Reports*, 3(12), 2014.

[9] H. Giese, N. Bencomo, L. Pasquale, A. J. Ramirez, P. Inverardi, S. Wätzoldt, and S. Clarke. Living with uncertainty in the age of runtime models. In *Models@runtime*, pages 47–100. Springer, 2014.

[10] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. MQTT- S A publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*.

[11] V. Krishnamurthy. *Partially Observed Markov Decision Processes*. Cambridge University Press, 2016.

[12] J. Krumm. *Ubiquitous computing fundamentals*. CRC Press, 2016.

[13] M. Kwiatkowska. Advances in quantitative verification for ubiquitous computing. In *International Colloquium on Theoretical Aspects of Computing*, pages 42–58. Springer, 2013.

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

[15] J. Nehmer, M. Becker, A. Karshmer, and R. Lamm. Living assistance systems: an ambient intelligence approach. In *Proceedings of the 28th international conference on Software engineering*, pages 43–50. ACM, 2006.

[16] M. Pinto, R. Almeida, P. Pinho, and L. Lemos. Daylighting in classrooms-the daylight factor as a performance criterion. In *ICEH-3rd International Congress on Environmental Health*, 2014.

[17] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[18] K. Sacha. Translatable finite state time machine. In *International SDL Forum*, pages 117–132. Springer, 2007.

[19] I. Sarray, A. Ressouche, D. Gaffé, J.-Y. Tigli, and S. Lavirotte. Safe composition in middleware for the internet of things. In *Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT*, pages 7–12. ACM, 2015.

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

[21] S. D. Stoller, E. Bartocci, J. Seyster, R. Grosu, K. Havelund, S. A. Smolka, and E. Zadok. Runtime verification with state estimation. In *International Conference on Runtime Verification*, pages 193–207. Springer, 2011.

[22] G. Tamura, N. M. Villegas, H. A. Müller, J. P. Sousa, B. Becker, G. Karsai, S. Mankovskii, M. Pezzè, W. Schäfer, L. Tahvildari, et al. Towards practical runtime verification and validation of self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II*, pages 108–132. Springer, 2013.

[23] F. Wagner, R. Schmuki, T. Wagner, and P. Wolstenholme. *Modeling software with finite state machines: a practical approach*. CRC Press, 2006.

[24] D. Weyns, N. Bencomo, R. Calinescu, J. Camara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek, et al. Perpetual assurances in self-adaptive systems. In *Assurances for Self-Adaptive Systems, Dagstuhl Seminar*, volume 13511, 2014.

[25] S.-Z. Yu. Hidden semi-markov models. In *Journal of Artificial Intelligence*, volume 174, pages 215–243, 2010.

[26] M. Zhang, B. , Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren. Understanding uncertainty in cyber-physical systems: A conceptual model. In *ECMFA*, 2016.

[27] X. Zheng, C. Julien, M. Kim, and S. Khurshid. On the state of the art in verification and validation in cyber physical systems. *The University of Texas at Austin, The Center for Advanced Research in Software Engineering, Tech. Rep. TR-ARiSE-2014-001*, 2014.