2015 IEEE 39th Annual International Computers, Software & Applications Conference

# Evaluating Energy Consumption for Cyber-Physical Energy System: an Environment Ontology-Based Approach

Xiaohong Chen, Fan Gu, Mingsong Chen*, Dehui Du, Jing Liu, and Haiying Sun
Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, 200062, China
{xhchen, fgu, mschen, dhdu, jliu, hysun}@sei.ecnu.edu.cn
* Corresponding author

***Abstract*—Energy consumption evaluation is one of the most important steps in Cyber-Physical Energy System (CPES) development. However, due to the lack of accurate and effective modeling and evaluation approaches considering the uncertainty of environment, it is hard to conduct the quantitative analysis for the energy consumption of CPESs. To address the above issue, this paper proposes an environment-aware energy consumption evaluation framework based on the Statistical Model Checking (SMC). In our framework, the environment uncertainty of CPESs is modeled using the Stochastic Hybrid Automata (SHA). In order to describe various environment modeling patterns, we create a collection of parameterized SHA models and save them to a domain specific environment ontology. Based on the domain environment ontology and user designs in the form of UML sequence diagrams and activity diagrams, our framework can automatically guide the construction of CPES models using networks of SHA and conduct the corresponding energy consumption evaluation. A case study based on an energy-aware building design demonstrates that our approach can not only support the accurate environment modeling with various uncertain factors, but also can be used to reason the relations between the energy consumption and environment uncertainties of CPES designs.**

***Keywords*-Cyber-Physical Energy Systems; Statistical Model Checking; Stochastic Hybrid Automata (SHA); Environment Ontology; Uncertainty of Environment.**

## I. Introduction

Cyber-Physical Energy Systems (CPESs) [1], [2] are a kind of modern energy system (e.g., smart grid, smart building [3]) that combines both information technology and physical thermal infrastructures to enable the efficient management and control of energy consumption. While taking external physical environment into account, the design of CPESs targets to find a control solution that can minimize the overall energy consumption. Since there may exist multiple control design candidates during the search of optimal solutions, the energy consumption evaluation of these candidates is becoming an important step in the design of CPESs. A wise design of CPES controllers can not only save the overall electrical energy cost, but also can enhance the quality of services [2], [4].

While considering various uncertain factors in real environment, the energy consumption evaluation for CPESs is very complex. As defined in Physics, energy equals the integral of power over time. However, the unpredictable events in environment may inevitably interfere with the real-time power usage of devices, which makes the accurate energy consumption calculation extremely hard. Due to the lack of accurate modeling techniques and automated tools, how to achieve a variation-aware controller considering the uncertainties of environment is becoming a major challenge in CPES design.

Since dealing with the uncertainties of environment in an early stage can enhance the probability of a successful implementation in a cost-effective manner, there are many research works presented to handle the uncertainties of environment during the requirement phase [5]–[7]. However, most existing approaches focus on the discrete change of physical world rather than the continuous time model. As a promising Statistical Model Checking (SMC) based approach for modeling continuous and stochastic behaviors of systems, Stochastic Hybrid Automata (SHA) [8], [11] has been widely investigated to conduct the quantitatively evaluation of system performance [19]–[21]. Based on the timed automata together with the extended stochastic semantics, SHA are quite suitable for the modeling of external uncertain environment as well as internal controllers for CPESs. By composing the SHA of both environment and controllers, we can obtain a Network of SHA (NSHA), which enables the comprehensive evaluation of the whole CPES. However, due to the diversity among different CPESs, the SHA-based modeling of different kinds of uncertain environment and controllers requires significant expertise and human efforts. To guarantee the quality of the generated NSHA models and automate the evaluation process for CPESs, ontology-based approaches [9] can be used to guide the generation of domain models.

In this paper, we propose to model the physical world (also environment of the software) of CPES with SHA. In order to guide people to get better SHA model, we give an environment ontology consisting of both upper ontology and domain ontologies. The upper ontology offers basic concepts for modeling the environment using SHA, while the domain ontology provides pattern models in the domain. In our approach, the physical world of CPESs is constructed by the environment models. The cyber world of CPESs is generated based on the input design strategies in the form of UML sequence diagrams and activity diagrams. By composing both the physical world and cyber world using an NSHA, we can conduct the automated energy consumption evaluation based on the tool UPPAL-SMC [12]. To demonstrate the efficacy of our approach, an experiment on a smart building based example is presented to investigate the relations between the

0730-3157/15 $31.00 © 2015 IEEE
DOI 10.1109/COMPSAC.2015.114

energy consumption and uncertain environment as well as the relations between the energy consumption and software controller. Experimental results show that our approach can not only be used to automatically and accurately generate the CPES models considering various kinds of uncertain environments, but also can be used to compare the energy consumption of CPESs with different design strategies.

The rest of this paper is organized as follows. Section II introduces the notations of SHA and NSHA. Section III gives the details of our framework. Based on a smart building example, Section IV presents how to construct the upper ontology and the domain environment ontology. Section V takes an energy-aware building as an example to show how to construct the NSHA models to enable the energy consumption evaluation. After the introduction of related works in Section VI, Section VII concludes the paper.

## II. Preliminaries

In our approach, we use NSHA to model CPESs where the uncertain environment is modeled using SHA. We adopt the tool UPPAAL-SMC [12] to perform the quantitative energy consumption analysis of CPESs.

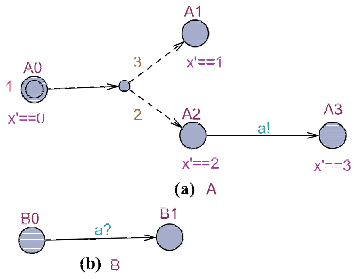


Fig. 1. An NSHA, (A|B)

SHA are timed automata whose clock rates can be different at different locations (i.e., states) [8]. Figure 1 shows a graphical representation of an NSHA consisting of two SHA, i.e., $A$ and $B$. The SHA $A$ has four states $A_0$, $A_1$, $A_2$ and $A_3$, and has a clock variable $x$. In $A_0$, $x'$==0 means that the value of $x$ has no change. The label "1" near location $A_0$ indicates that the the delay of SHA $A$ at location $A_0$ follows an exponential distribution with $\lambda = 1$. SHA supports the nondeterministic execution. For example, when SHA $A$ exits the location $A_0$, it has two choices of the target location, i.e., $A_1$ and $A_2$. In SHA, we use the dashed arrow line labeled with the weight information to denote the possibility of the entering a target location. For example, the probability of going to location $A_1$ from $A_0$ is $3/(3+2)$, while the probability of going to location $A_2$ from $A_0$ is $2/(3+2)$. To conduct synchronization between SHA in an NSHA, SHA communicate each other via broadcast channels and shared variables. As an example shown in Figure 1, the two SHA communicate each other using the channel $a$. After the synchronization using the send operation (denoted by !) and receive operation (denoted by ?), both SHA will go to their next locations (i.e, $A_3$ and $B_1$ in this example) simultaneously.

As an extension of the model checker UPPAAL, UPPAL-SMC [12] allows the statistical model checking of the given NSHA models. By monitoring the randomly simulated system behaviors, UPPAL-SMC provides a user-friendly GUI framework that enables the visualization of the values of expressions along runs as well as the statistical model checking for the specified properties. To perform the evaluation of energy consumption, our approach adopt the properties in the format of $Pr[attr <= bound](<> expr)$ which tries to figure out the probability that $expr$ happens finally (denoted by $<>$) with a given constraint $attr$ whose value is bounded by a constant $bound$. For example, the query $Pr[energy <= 1000000](<> time >= 2 * day)$ tries to check the probability that the system can last for more than two days with an energy quota of 1000000.

## III. Our approach

CPES integrates the cyber world and physical world, aiming at better controlling physical world using cyber world. It contains two parts: cyber world and physical world. The cyber world is usually software system (including network). As it aims to control physical world, we name it *controller*. The physical world is actually the environment of controller. We name it *environment*.

In our approach, both the environment and the controller must be modeled in SHA, and they form an NSHA to enable the energy consumption evaluation. Therefore, we design a framework as shown in Figure 2. There are mainly four parts, i.e., environment modeling, controller modeling, system construction, and analysis and evaluation. We use environment ontology to save patterns for environment modeling (named as parameter models), and use configuration table to instantiate parameters. As to the controller modeling, we use context diagram [10] to support the synchronization of environment and message exchange, and system sequence diagram and activity diagrams as an input to automatically generate controller model in SHA. The system can be constructed by instantiating environment and controller. Then the system could run on UPPAL-SMC which could execute specified properties.

### A. Environment modeling

Similar to our previous work [13], [14], this paper assumes the environment to be a set of *environment entities*. We classify these entities, and analyze their characteristics for modeling uncertainty in SHA. The uncertainty finally turns out to be parameters in the model. We call them parameter models, and obtain parameter models for entities in a specific domain. The upper environment ontology is used to provide basic modeling terms of SHA, and domain ontology to record the parameter models for environment entities in SHA in that domain. Details about ontologies are given in the later section.

Based on the environment ontology, there are two steps to obtain an environment model. (1) Ask the requirement providers to choose environment entities from the environment ontology as well as entity classifications and parameters; and (2) set parameter values using configuration table.

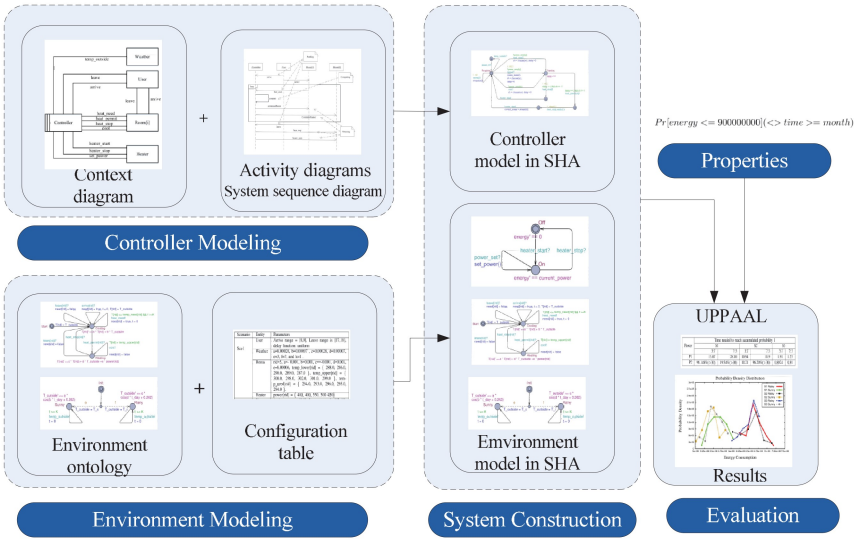


Fig. 2. Framework of our approach

The parameters are replaced with values from requirements providers. Then SHA description would be obtained by transforming descriptions from ontology. The transformation are done automatically by mapping the concepts in the ontology to SHA modeling concepts. The only exception is the parameter variables. In order to distinguish parameter variables from other variables, we declare parameters to special data type, for instance, *int*.

### B. Controller Modeling

Firstly we would explicitly express a controller by giving it a name. As a controller in CPES systems, it needs to read information from all kinds of environment entities, and making decisions to control some controlled entities. So basically, there should be at least two locations in the controller: *Reading* and *Deciding*. All these locations are interacting with the outside world, so we need to know the interactions between the controller and environment. We use the *context diagram* to capture these interactions.

However, the context diagram is only a static diagram. We need to know the temporal relations among these interactions, that is, sequences. So the *system sequence diagram* is used for showing when these interactions happen. In the following, we classify these interactions. All the interactions could be classified into 3 parts, i.e., *Reading*, *Computing*, and *Deciding*. The *Reading* includes two types, i.e., reading driven by time, and reading by events. The *Deciding* parts should give commands to controlled entities. Limited by expression of SHA (each location sends one command), for n commands that must be done one by one we must add n-1 urgent locations to express the relation between these n commands. The *Computing* parts presents conditions for location transitions. But they can not be given in the system sequence diagram. We suggest using *activity diagrams* for *Computing* in the controller. From the system sequence diagram and activity diagrams, we give transformation rules to obtain SHA models for controllers. The interactions belonging to different parts suggest different locations, updates in the SHA model. The detailed rules are shown in Table I.

TABLE I
TRANSFORMATION RULES

| Interaction classification | Component of diagram | SHA |
|---|---|---|
| Reading | None | Location *Reading* |
| | Activity | Synchronization<br>Transition: *Reading* to *Reading* |
| | Message | Synchronization<br>Transition: *Reading* to *Reading* |
| Computing | Condition | Guard<br>Transition: *Reading* to *Deciding* |
| | Activity | Update<br>Transition: *Reading* to *Deciding* |
| | Message | Update<br>Transition: *Reading* to *Deciding* |
| Deciding | None | Location *Deciding* |
| | Condition | Guard<br>Transition: *Deciding* to *Urgent* Location |
| | Activity | Synchronization<br>Transition: *Deciding* to *Urgent* Location |
| | Message | Synchronization<br>Transition: *Deciding* to *Urgent* Location |

### C. Evaluation

In order to support quantitative evaluation, we provide two types of energy related queries. They are:

- $Pr[energy <= EC](<> time >= X)$ meaning to return the probability distribution that total energy consumption less than EC unit within X time.
- $Pr[<= X](<> range\_out >= Y)$ meaning to return within X time, the possibility that ranges out Y time unit.

The users are asked to choose the energy related queries, and fill in these parameters. The system will run on the UPPAL-SMC. Visual and data results could be displayed.

## IV. Environment Ontology

### A. Upper Ontology

By summarizing the basic modeling concepts of SHA, we present an upper environment ontology. Figure 3 shows its conceptual model. The meanings of these terms are given in Table II. Their associations are shown in Table III.

As we said earlier, the environment of a controller is assumed to be a set of *environment entities* which will interact with this controller. An environment entity may has many attributes. In terms of attributes, the environment entities could be classified into *monitored entities* and *controlled entities*. The monitored entities are entities whose attributes could only be monitored rather than controlled. For example, the weather could influence the outside temperature but could not be influenced by a controller. The controller could only monitor the temperature of outside. The controlled entities are entities whose attributes could be controlled by a controller.

Each environment entity could be modeled by an SHA. An SHA has many *attributes*, *locations* and *transitions*. A location has *invariants* on attributes, and a *delay function*. A transition sources from one location, and sinks to another. It has *guards* which mean transiting conditions and updates and *synchronization*. We distinguish a special kind of transition: *virtual transition* for expressing probability of transiting direction. It only has updates and *probability weight*.

Because the monitored entities can not be influenced by controllers, only the monitored entity model has uncertainty. The uncertainty is embodied in delay functions of location or probability weights of transitions. From the perspective of controllers, the transitions of monitored entities are triggered by time or chance. For example, user. You can not predicate when the user arrives. By observation, the entity may transit from one location to another at any time. On the contrary, the controlled entity has deterministic inputs and outputs. The inputs and outputs have specific causal relations. It is actually modeled by common timed automata. No uncertainty exists.

### B. Smart Building Domain Environment Ontology

This subsection aims to take smart building domain as an example to show how to build a domain environment ontology. Firstly, we need to find the environment entities in the domain. Consider a scenario proposed by [15] as a benchmark for smart building. The benchmark consists of a building layout with temperature dynamics, autonomous heaters and a central controller deciding which room gets a heater. We get 3 environment entities, i.e., *User, Heater*, and *Room*. The other entities that relate to these 3 entities are also in consideration. As the temperature in the room are influenced by outside temperature, we add one more entity: *Weather*.

Secondly, ask the provider to judge whether these entities are monitored or controlled entities. Among these entities, *User* and *Weather* could not be influenced by *Controller*, so they are monitored entities. *Room* and *Heater* could be influenced by *Controller*, so they are controlled entities.

Thirdly, for the monitored entities, ask the expert to give their SHAs. Users come at different times. Assume that the

TABLE II
Hierarchy of the Concept Categories

| Concept Category | Meaning |
|---|---|
| Environment Entity | a relevant real world entity that will interact with the software |
| Monitored Entity | whose properties can be observed but can not be controlled by a software |
| Controlled Entity | whose properties can be controlled by a software, its properties have predictable causal relationship |
| SHA | timed automata whose clock rates can be changed to be constants or expressions depending on other clocks |
| Location | a state that keeps |
| Transition | used to express the entity location change |
| Attribute | the stable value property of the environment entity in a location |
| Invariant | the expression that holds of attributes in the location |
| Virtual Transition | a transition that has probability weight |
| Guard | triggering condition |
| Synchronization | message sent or received |
| Update | something to be done |
| Probability Weight | a weight that indicates the happening possibility |

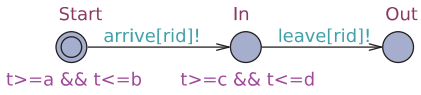


Fig. 4. User model: uniform

users get to work in a time interval [a, b] (e.g. a=8:00am, b=9:00 am), and get off within a time interval [c,d] (e.g., c=5:00pm, d=6:00 pm). They could come at any time between 8:00-9:00 am, and leave between 5:00-6:00 pm. So we could model them as a uniform distribution as shown in Figure 4. In this figure, there are three locations, Start, In, and Out. They have an attribute: time $t\_day$. In location In, there is an invariant $t\_day >= a \&\& t\_day <= b$. From Start to In, the entity sends a message '*arrive*'. From In to Out, it sends '*leave*'.

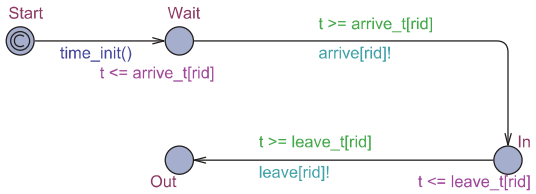


Fig. 5. User model: normal

The time interval of users coming could be modeled as other distributions. For example, it may be a normal distribution. We model them in Figure 5 where $time\_init()$ gives the normal distribution. We add one more location 'wait' for arranging $time\_init()$. In $time\_init()$, according to the distribution function, it allocates the $arrive\_t$ and $leave\_t$. Generally speaking, we adopt the second model. The parameters in this model includes *a, b, c, d,* and *distribution function*.

Then we consider *Weather*. Assume that the outside temperature in Shanghai is within a range. Basically, there are

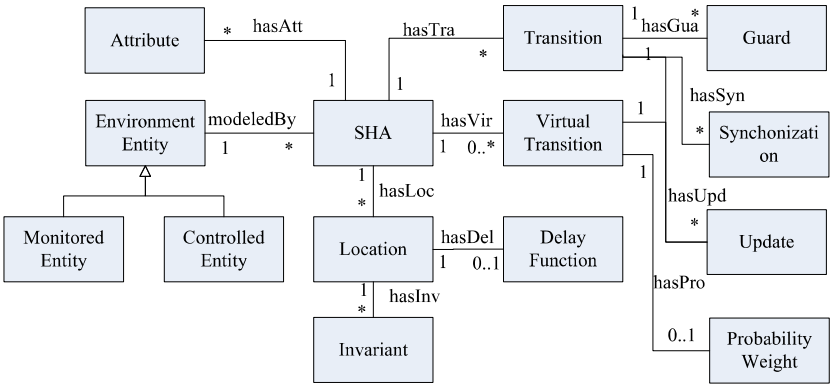


Fig. 3. Conceptual model

TABLE III
ASSOCIATIONS AMONG THE CONCEPTS OF THE ENVIRONMENT ONTOLOGY

| Association | Formation | Meaning |
|---|---|---|
| modelledBy | Environment Entity → SHA | Each environment entity can be modelled by one or more SHA |
| hasLoc | SHA → Location | Each SHA has one or more locations |
| hasAtt | SHA → Attribute | Each SHA has one or more attributes |
| hasInv | Location → Invariant | Each location may have many invariants |
| hasDel | Location → Delay Function | Each location has at most a delay function |
| hasTra | SHA → Transition | Each SHA has one or more transitions |
| hasVir | SHA → Virtual Transition | Each SHA may have many virtual transitions |
| hasTra | SHA → Transition | Each SHA has one or more transitions |
| hasGua | Transition → Guard | Each transition has one or more guards |
| hasSyn | Transition → Synchronization | Each transition has one or more synchronization |
| hasUpd | Transition → Update | Each transition has one or more updates |
| hasUpd | Transition → Update | Each transition has one or more updates |
| hasUpd | Virtual Transition → Update | Each virtual transition has one or more updates |
| hasPro | Virtual Transition → Probability Weight | Each virtual transition has at most 1 probability weight |

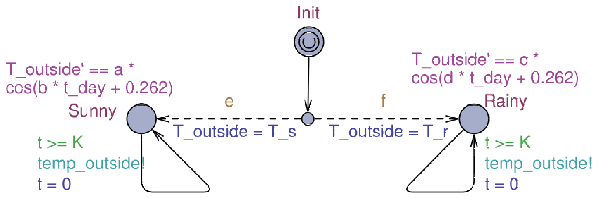


Fig. 6. Weather model

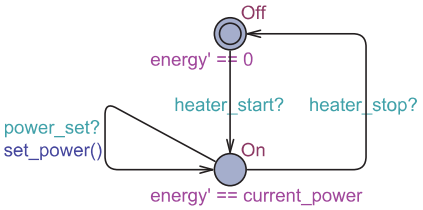


Fig. 7. Heater model

two kinds of weather: rainy and sunny. In a month, there are possibilities of rainy days. We use both $sin$ method to simulate the temperature change. Both in rainy days and sunny days they conform to the same dynamics: $T\_outside' == a * cos(b * t\_day + 0.262)$. They differ in parameters and offsets. Figure 6 shows the weather model. There are three locations: *init*, *rainy* and *sunny*. From *init* to *rainy* and *sunny*, there are uncertain. The possibility weights are *e* and *f*. It sends $Temp\_outside$ every $K$ seconds. The parameters in this model includes *a, b, c, d, e, f* and *K*.

Finally, for the controlled entities, ask the expert to give their SHAs. The controlled entities also have configuration parameters. First we consider *Heater*. A Heater has two locations: *On* and *Off*. They consume different amount of energies. In the *Off* location, no energy consumes. So *energy'==0*. In the *On* location, *energy'==current_power*. In addition, power is set at the *On* location. Figure 7 shows the heater model which moves between locations *Off* and *On*. In this model, the configure parameter is *power*.

Then we consider *Room*. The SHA model of a room with an identifier *rid* is shown in Figure 8. There are three locations: *Start*, *Cooling*, and *Heating*. In *Cooling* and *Heating*, the room temperature dynamics can be described by different differential equations, for example,
$T[rid]' == -0.001 * T[rid] + 0.001 * T\_outside(Cooling)$
and
$T[rid]' == -0.001 * T[rid] + 0.001 * T\_outside + 0.00006 * power[rid](Heating)$

Temperature has three thresholds: *temp_upper*, *temp_lower*, and *temp_need* (*temp_lower* may be the same as *tem-*

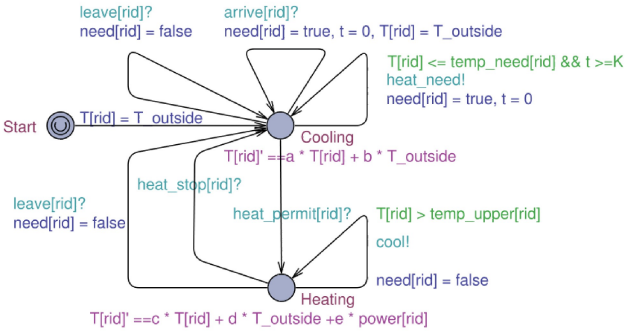


Fig. 8. Room model

*p_need*). From *Cooling*, if the user arrives, and temperature *T[rid]<=temp_need* (waiting for *K* seconds), Room *rid* sends *heat_need*. If it gets message *heat_permit*, it transits to Heating. In *Heating*, if *T[rid]>=temp_upper*, it sends request *cool*. If it gets message *heat_stop*, it transits to *Cooling*. If the user leaves, it transits to *Cooling*. In this model, the parameters include *temp_upper*, *temp_lower*, *temp_need* and wait time *K*.

## V. Case Study

We will use a smart building provided in [15] as a case study. In this smart building, suppose there are five rooms. These rooms have different wall thicknesses and sizes. They share one heater which have many different powers. A central controller decides which room gets a heater.

### A. Construction of the simulating system

**Step 1. Environment modeling**

First we choose the name of environment entities from the smart building domain environment ontology. Heater, Room, User, and Weather are four chosen environment entities. Among these entities, Heater and Room are controlled entities, and User and Weather are monitored entities. From the environment ontology, we find parameters for each monitored entity. By setting these parameters, we obtain a scenario. In Table IV, we give two scenarios. Sce1 has uniform distribution in User, a 7/10 Sunny possibility in Weather, lower power in each room, while Sce2 has normal distribution in User, a 3/10 Sunny possibility in Weather, and higher power in each room.

**Step 2. Controller modeling**

Firstly we add a controller, and name it to be *Controller.* For each environment entity and the controller, we add an association. For instance, the Weather sends *Temp_Outside* every 1 second. Then the controller receives *Temp_Outside* every 1 second. We records this in a context diagram as shown in Figure 9.

Next, we analyze interactions between the Controller and its environment entities. Monitored entities should pay special attention. For instance, the weather. It sends message every 1 seconds. The interactions between the Controller and the Weather is shown in Figure 10.

Figure 11 shows the interactions between other entities. The user arrives. It notifies Room and the Controller. Then Room

TABLE IV
Configuration Table

| Scenario | Entity | Parameters |
|---|---|---|
| Sce1 | User | Arrive range = [8,9], Leave range is [17,18], delay function: uniform |
| | Weather | a=0.00028, b=0.00007 , c=0.00028, d=0.00007, e=3, f=7, and k=1 . |
| | Room | rid=5, a=- 0.001, b=0.001, c==-0.001, d=0.001, e=0.00006,temp_lower[rid] = { 288.0, 286.0, 290.0, 289.0, 287.0 },temp_upper[rid] = { 300.0, 298.0, 302.0, 301.0, 299.0 },temp_need[rid] = { 294.0, 293.0, 296.0, 295.0, 294.0 }. |
| | Heater | power[rid] = { 480, 400, 550, 500, 450} |
| Sce2 | User | Arrive range = [8,9], Leave range is [17,18], delay function: normal |
| | Weather | a=0.00028, b=0.00007 , c=0.00028, d=0.00007, e=7, f=3, and k=1 . |
| | Room | rid=5, a=- 0.001, b=0.001, c==-0.001, d=0.001, e=0.00006,temp_lower[rid] = { 288.0, 286.0, 290.0, 289.0, 287.0 }, temp_upper[rid] = { 300.0, 298.0, 302.0, 301.0, 299.0 }, temp_need[rid] = { 294.0, 293.0, 296.0, 295.0, 294.0 }. |
| | Heater | power[rid] = { 780, 700,850,800,750 } |

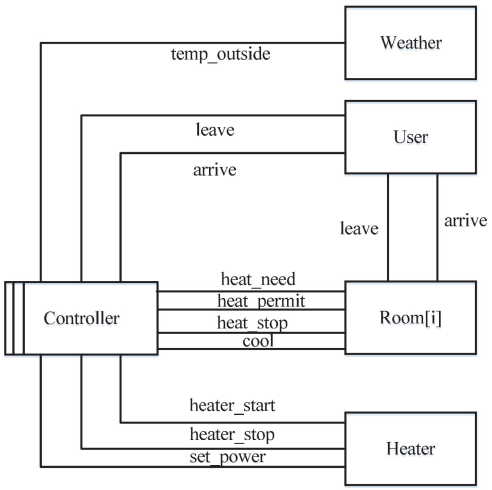


Fig. 9. Context diagram for EAB

sends heat request to the Controller. Through computation, it makes decisions to inform Room and the Heater. When the User leaves, the Controller sends a message to stop heating.

Based on this, we get the Controller model as shown in Figure 12. It has two locations: *Reading* and *Deciding*. In *Reading*, the Controller reads messages from Weather, Room, and User. In Sending, it sends commands to rooms according to the queue which lists rooms needing heating. In this diagram, we explicitly denote which interactions/ messages belong to Reading part, Computing part, and Deciding part.

In the sequence diagram, the computing part is the actual


Fig. 10. Interaction between the Controller and the Weather

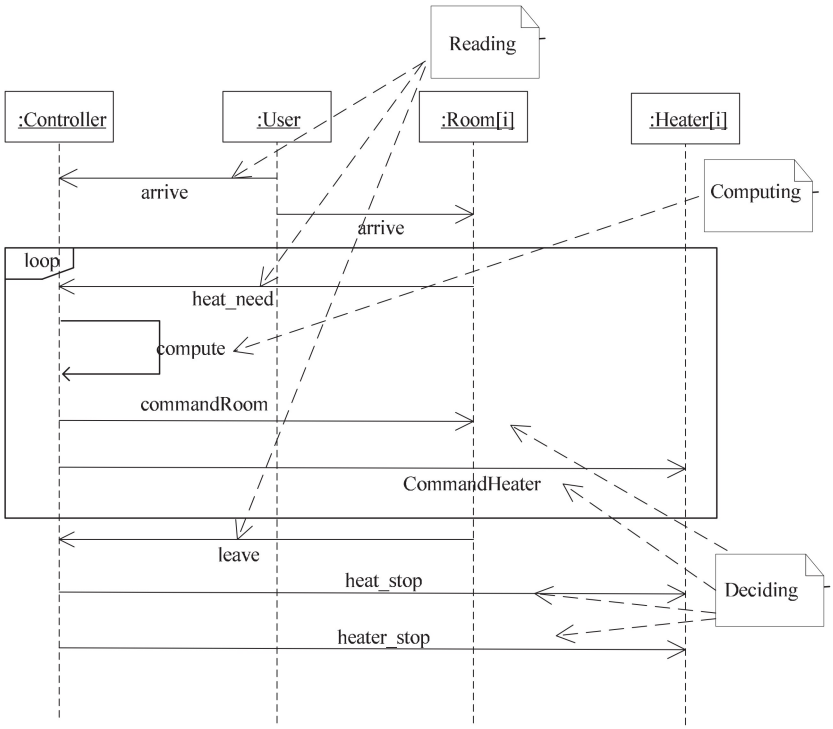


Fig. 11. System sequence diagram of EAB

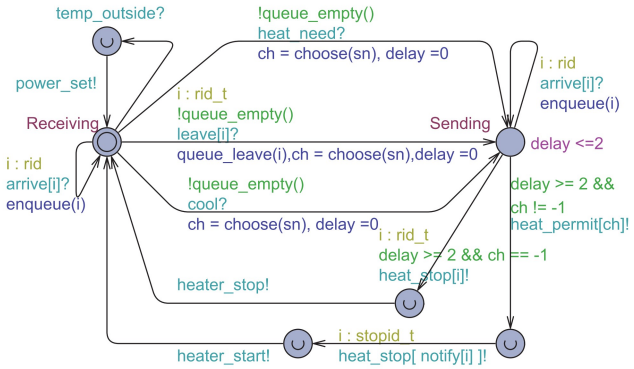


Fig. 12. Controller model

Controller. It also means the queue part. Different strategies may lead to different decisions. We consider three strategies:

- Strategy 1: first come first service
- Strategy 2: lowest room temperature comes first
- Strategy 3: lowest room temperature comes first, but must wait for current heating room reaching comfort temperature $temp\_lower$

These three strategies will influence enqueue(), queue_leave(), and choose(sn). Figure 13 shows how to do these.

From the context diagram, finally, we get a system declaration.

$$system\ Controller,\ Heater,\ Weather,\ Room,\ User;$$

**Step 3. Simulation conduction and result analysis**

This step provides some energy related queries. For instance, $Pr[energy <= 900000000](<> time >= month)$ means to find the probability distribution that total energy consumption of a heater less than 900000000 unit within a month. Then we run the *system* and verify these energy queries in the UPPAL-SMC, which in turn displays the results.

## B. Experiment

The aim of the experiment is to explore the relation between energy consumption and the environment uncertainty. We use two scenarios set in Table IV. We denote Power in Sce1 to be P1, and in Sce2 to be P2. The first weather called Sunny, and the second weather called Rainy. We compare S1, S2, and S3 under two different scenarios.

(1) *Weather influence*

Figure 14 shows the energy consumption probability density function with X position to be energy consumption and Y position to be probability density. In this figure, the area of diagram enclosed by the curve and X position is 1. There is a rule says that the higher in the curve near the left part of the figure, the higher probability of smaller value of air conditioning energy consumption.

In this figure, we first compare energy consumptions of the different strategy behaves under the same weather. Under Sunny day, S2's left part is higher than S1 and S3, so S2 consume less energy, it is better. Under Rainy day, firstly S2 is better than the other two. After some time S3 is better. In the

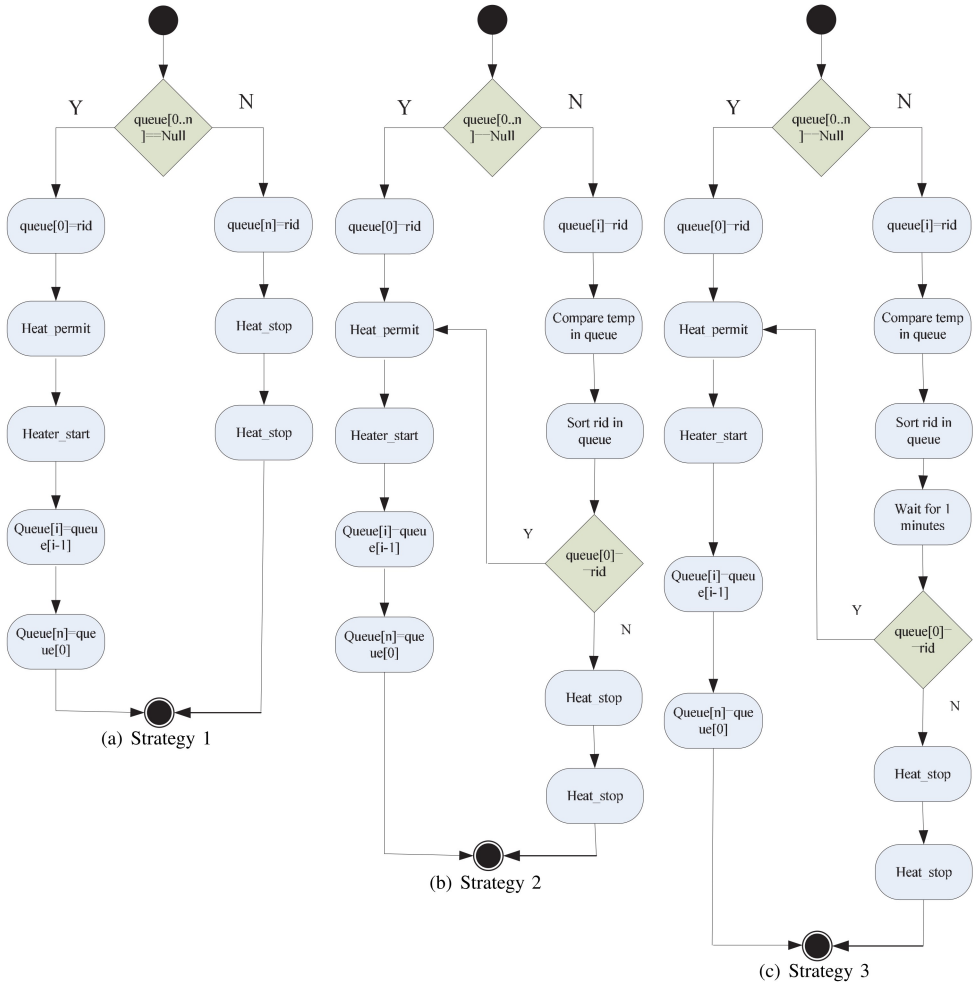


Fig. 13. Activity diagrams of strategies

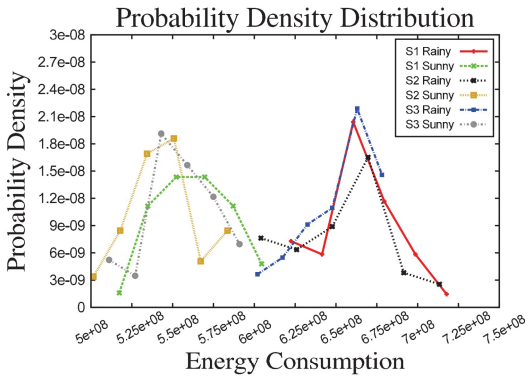


Fig. 14. Energy consumption: weather influence

long run, we should select S3. To sum up, different weathers suggest choosing different strategies.

(2) *User influence*

Figure 15 shows energy consumptions of different users. Under uniform distribution, S1 is better than S2 and S3. As to the normal distribution, S1 is still better than S2 and S3. To sum up, different users do not suggest choosing different strategies in terms of energy.

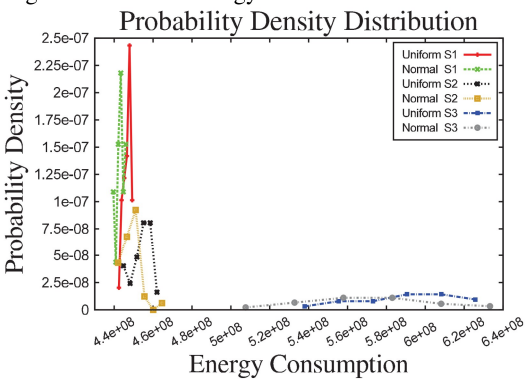


Fig. 15. Energy consumption: user influence

(3) *Power influence*

Figure 16 shows energy consumptions in different powers under rainy days. From this figure, we could see that different powers of a heater seems do something to the energy consumption. For P1, we could see S1 is better than S1 and S3 at first. Then after some time S3 is better. So it sounds that S3 is a better choice. For P2, S2 is better. P1 and P2 make a difference on energy consumption. Different device changes will influence strategy choice in terms of energy consumption.

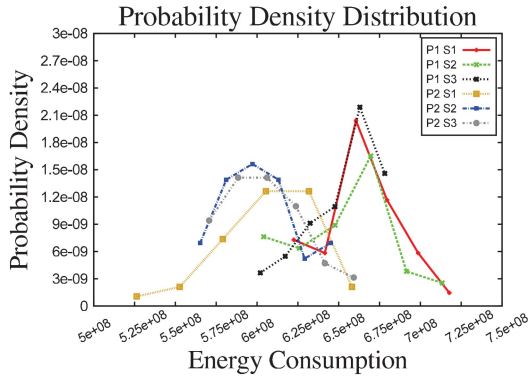


Fig. 16. Energy consumption: power influence

**Discussion** From the above results, we can see that the monitored entities will influence energy consumption. Different controlled entities marked by different parameters will also influence energy consumption. The selection of strategies can not rely on energies only. The comfort time must be considered. By verifying

$$Pr[<= month](<> range\_out[1] >= 2500)$$

within a month, the possibility of Room number 1 ranging out 2500s is obtained.

Table V gives the time to reach accumulated probability 1 (User: uniform). From this table, we could see that S1 and S2 behaves as expected. It means, the higher the power, the easier to over temperature limits and reach 1. But S3 needs more time to reach 1 by increasing more power. Comparing data under different weathers, S1 costs more days to achieve 1. So S1 is better than S2 and S3 in terms of comfort days. This choice is quite different from only considering energy consumption.

## VI. Related Work

Cyber-physical energy systems are complex systems that involve both physical and cyber variables like battery life, power flow, computing processes, network limitations, etc. [2]. Since energy efficiency plays a great role in CPES design [3], so far, majority of the studies on CPES modeling are focused on the perspective of power. For example, in [16], Ilic utilized the Differential Algebraic Equations (DAE) as a more general way to model the electric power systems. Their proposed DAE-based models are simplified and divided by the time span to cover the phenomena of interest, ranging from milliseconds to years. In [17], Ilic et al. presented a dynamic model for future CPESs. The underlying mathematical description of the models depends on the cyber technologies considering the physical world. Unlike the above researches which conduct the CPES modeling and energy efficiency analysis in the design phase, our work focuses on the CPES modeling in the requirement phase and early design phase. Besides investigating various continuous properties of the physical world, our approach takes the uncertainty of the physical world into account.

Uncertainty has been widely studied in the requirement modeling and analysis. In [5], Ali et al. presented a goal-based framework for contextual requirement modeling and analysis. In this work, the changing environment in the requirements is modeled as a varying discrete context. In [6], Qureshi and Perini proposed an approach that can support system analysts to engineer adaptive requirements at requirements-time by combing the goal-based approaches and domain techniques. They extended the goal model in Tropos with domain modeling where the knowledge is represented by ontology. In [7], Cheng et al. introduced a goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. Based on the KAOS model, the uncertainty of environment was modeled by the threats that the system will confront. Although the above approaches are promising in exploring how various uncertainty factors impact requirements, most of them only investigated the discrete aspects of the physical world rather than the continuous aspects (e.g., time). Consequently, the generated models and analysis results are inaccurate. Compared with these uncertainty-aware approaches, our method supports both the modeling and queries of various continuous aspects of CPESs based on the clocks provided by SHA.

Due to the efficacy and scalability, statistical modeling checking-based approaches have been widely used in quantitatively evaluating the performance of system designs under variations. For example, David et al. [8], [18] extended the stochastic semantics of UPPAAL-SMC to enable the modeling and evaluation of objects in various research domains, including biology and energy-aware buildings. In [19], Chen et al. proposed an approach based on UPPAAL-SMC to enable the quantitative analysis on the resource allocation strategies in Cloud computing. In [20], Chen et al. adopted an SMC-based approach to quantitatively reason the performance of the task allocation and scheduling strategies. In [21], Du et al. adopted UPPAAL-SMC to perform the quantitative evaluation on project schedules considering the variations of human resources. Although our framework employs a similar SMC-based approach for the performance evaluation, the main purpose of our approach is to guide the uncertainty-aware environment modeling and controller synthesis for CPESs based on the provided environment ontology in the requirement phase. In our approach, designers only need to choose the configuration parameters. The modeling and evaluation work will be conducted by our framework automatically, which can drastically save the design efforts. To the best of our knowledge, our work is the first one that combines both the ontology and SMC to enable the uncertain environment construction and

TABLE V
TIME NEEDED TO REACH ACCUMULATED PROBABILITY 1

| Power | Time needed to reach accumulated probability 1 | | | | | |
|---|---|---|---|---|---|---|
| | S1 | | S2 | | S3 | |
| | 3:7 | 7:3 | 3:7 | 7:3 | 3:7 | 7:3 |
| P1 | 15.07 | 28.10 | 10.94 | 18.9 | 1.91 | 1.25 |
| P2 | 98.148%(>30) | 19.54%(>30) | 10.21 | 96.20%(>30) | 0.8924 | 0.91 |

quantitative evaluation of energy consumption for CPESs.

## VII. CONCLUSIONS

Energy consumption is of great importance for CPESs. Aiming at obtaining a proper CPES controller with low energy consumption under uncertain environment, this paper presents an evaluation framework based on UPPAAL-SMC that can accurately model and query the energy consumption considering various environment uncertainties. To enable the comprehensive modeling of environment uncertainties, our framework employs an environment ontology which can describe various patterns of uncertain environment using a collection of parameterized SHA models. Based on our proposed guiding process, the CPES controllers in SHA can be automatically obtained from the user design outputs in the form of UML sequence diagrams and activity diagrams. Combined with instantiated environment models with specified parameter configurations, an overall system model in the form of NSHA can be constructed for the purpose of energy consumption evaluation. The experimental results of an energy-aware smart building design show the significant energy consumption variation for CPES designs with different controllers and uncertain environment instances, which demonstrates the necessity and importance of our proposed approach. In our future work, we plan to conduct more experiments on larger CPES designs to check the scalability of approach. Moreover, tool support for automating the modeling and evaluation based on our framework is another our future work.

## ACKNOWLEDGEMENT

The authors would like to thank Professor Zhi Jin from Peking University for her insightful comments and suggestions. This work was partially supported by the National Nature Science Foundation of China (Grant No. 61202104, 91318301, 91418203, 61472140, 61332008), Innovation Program of Shanghai Municipal Education Commission 14ZZ047, Shanghai Project 13511503100, and NSF of Shanghai (Grant No. 14ZR1412500), and Shanghai Knowledge Service Platform ZF1213.

## REFERENCES

[1] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. LeeSeshia.org, 2011.

[2] C. Macana, N. Quijano, and E. Mojica-Nava, "A survey on cyber physical energy systems and their applications on smart grids," in *Proc. of IEEE PES Conference on Innovative Smart Grid Technologies (ISGT)*, pp. 1–7, 2011.

[3] J. Kleissl and Y. Agarwal, "Cyber-physical energy systems: Focus on smart buildings," in *Proc. of ACM/IEEE Design Automation Conference (DAC)*, 2010, pp. 749–754.

[4] X. Chen, and M. Chen, "Extending the Four-Variable Model for Cyber-Physical Systems," In *proceedings of Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, pp.31–36, 2012.

[5] R. Ali, F. Dalpiaz, and P. Giorgini, "A goal-based framework for contextual requirements modeling and analysis," *Requirements Engineering*, vol. 15, no. 4, pp. 439–458, 2010.

[6] N. A. Qureshi and A. Perini, "Engineering adaptive requirements," in *Prof. of ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2009, pp. 126–131.

[7] B. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, "A goal-based modeling approach to develop requirements for adaptive systems with environmental uncertainty," Dept. of Computer Science and Engineering, Michigan State University, Technical Report MSU-CSE-09-22, 2009.

[8] A. David, D. Du, K. G. Larsen, A. Legay, M. Mikucionis, D. Poulsen, and S. Sedwards, "Statistical model checking for stochastic hybrid systems," in *Proc. of International Workshop on Hybrid Systems and Biology (HSB)*, 2012, pp. 122–136.

[9] X. Chen and Z.Jin and L. Yi, "An Ontology of Problem Frames for Guiding Problem Frame Specification," in *Proc. of KSEM*, Volume 4798, 2007, pp. 384–395.

[10] M.Jackson, "Problem Frames: Analyzing and Structuring Software Development Problems," Harlow, England:Addison-Wesley, 2001.

[11] A. David, K. G. Larsen, A. Legay, M. Mikucionis, D. B. Poulsen, J. Vliet, and Z. Wang, "Statistical model checking for networks of priced timed automata," in *Proc. of International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2011, pp. 21–23.

[12] P. Bulychev, A. David, K. Larsen, A. Legay, M. Mikucionis and D. Poulsen:, "Checking and distributing statistical model checking," in *Proc. of NASA Formal Methods*, 2012, pp. 449–463.

[13] X. Chen, and J. Liu, and Z.Ding, "On Constructing Software Environment Ontology for Time-Continuous Environment," in *Proc. of International Conference on Knowledge Science, Engineering and Management (KSEM2011)*, pp.148–159, 2011.

[14] X. Chen, and Z.Jin, "An Ontology-guided Process for Developing Problem Frame Specification: An Example," in *Proceedings of the 3rd International Workshop on Applications and Advances of Problem Frames (IWAAPF'08)*, pp.36–39, 2008.

[15] A. Fehnker and F. Ivancic , "Benchmarks for hybrid systems verification," in *Proc. of Hybrid Systems: Computation and Control (HSCC)*, pp. 326–341, 2004.

[16] M. Ilic, "From hierarchical to open access electric power systems," in *Proceedings of the IEEE*, vol. 95, no. 5, pp. 1060–1084, 2007.

[17] M. Ilic, L. Xie, U. Khan, and J. Moura, "Modeling of future cyber physical energy systems for distributed sensing and control," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 40, pp. 825–838, 2010.

[18] A. David, D. Du, and L. G. Kim, "An evaluation framework for energy-aware building using statistical model checking," *SCIENCE CHINA Information Sciences*, vol. 55, no. 12, pp. 2694–2707, 2012.

[19] S. Huang, M. Chen, X. Liu, D. Du and X. Chen, "Variation-aware resource allocation evaluation for Cloud workflows using statistical model checking," in *Proc. of International Conference on Big Data and Cloud Computing (BDCloud)*, 2014, pp. 201–208.

[20] M. Chen, D. Yue, X. Qin, X. Fu and P. Mishra, "Variation-aware evaluation for MPSoC task allocation and scheduling strategies using statistical model checking," in *Proc. of Design, Automation and Test in Europe (DATE)*, 2015, pp. 199–204.

[21] D. Du, M. Chen, X. Liu and Y. Yang, "A novel quantitative evaluation approach for software project schedules using statistical model checking," in *Proc. of International Conference on Software Engineering (ICSE) Companion*, 2014, pp. 476–479.