New Definition of Environment Feature Interactions in Home Network System

Kousuke Ikegami
Kobe University, Kobe, Japan
Email: ikegami@ws.cs.kobe-u.ac.jp

Shinsuke Matsumoto
Kobe University, Kobe, Japan
Email: shinsuke@cs.kobe-u.ac.jp

Masahide Nakamura
Kobe University, Kobe, Japan
Email: masa-n@cs.kobe-u.ac.jp

***Abstract*—The environment feature interactions in the home network system refer to functional conflicts among features of home appliances, occurring on certain environment properties. Due to lack of the degree of environmental impact and explicit consideration of requirements, the previous formalization tends to overestimate many acceptable cases as feature interactions. To capture the environment interaction more precisely, this paper introduces an *environment impact model*, describing how much impact is given to the environment by each appliance operation within a service. We also describe user requirements by environment properties evaluated within the model. Then, the environment interaction is formalized as the unfulfilled requirement caused by the use of multiple services.**

***Keywords*-home network system, integrated services, feature interactions, environment**

I. Feature Interactions in HNS

The recent ICT technology enables house-hold appliances (e.g., TV, air-conditioner, light, etc.) to be connected in a local home network. A system managing such networked appliances is called *home network system* (HNS, for short). The HNS integrates various appliances via network, to provide value-added *integrated services* for home users. Examples of the integrated services are listed as follows.

- **S1: TVTheaterService (TVT)** configures a living room for watching a TV in a theater-like atmosphere. When activated, a curtain is closed, lights are turned off, and the TV is turned on.
- **S2: ComingHomeService (CH)** turns on a light in a lobby and activates an aroma diffuser for relaxation fragrance, when a user comes home.
- **S3: BGMService (BGM)** plays a back ground music using a music player in a living room.
- **S4: AirCleaningService (AC)** cleans and deodorizes the air in the room with an air-cleaner.

Even if individual services are implemented correctly, using multiple services together may cause a functional conflict leading to malfunction or unexpected behaviors. The problem is generally called *feature interactions* [1], and can be observed in the HNS as well. For example, suppose that a user A comes home while another user B is watching a TV by TVT. Then, the lights of CH brighten the room, which may ruin the atmosphere of TVT.

In [2], we originally defined two kinds of feature interactions in the HNS: *appliance interactions* and *environment interactions*. Matsuo et al. [3] elaborated our definition for a model checking framework. The appliance interaction occurs when two services request incompatible operations on the same appliance. A simple example is that one service turns on a TV while another service turns off the same TV. The appliance interaction can be characterized by satisfiability of goals and premises of appliance operations [2][3].

On the other hand, the environment interaction occurs when two operations of different appliances interfere indirectly via environment. The interaction between CT and TVT explained above is an environment interaction, since `LobbyLight.on()` interfere to `LivingLight.off()` indirectly via an environment property `Brightness`. Compared to the appliance interaction, the environment interaction is much more difficult to formalize, since there are many ways to capture the "undesirable" interference on the environment.

II. Limitation of Previous Methods

Our original method [2] characterized the environment interactions simply by incompatible *READ* or *WRITE* operations to an environment property. This definition was a bit extended in [3] by introducing the *direction* of the impact. However, we found that these formulations were so coarse that they tend to misjudge many "acceptable" or "undesirable" interactions. Specifically, they lacked the *degree of impact* to the environment. Also, they did not count *user requirements* to the environment explicitly.

III. Proposed Method

A. Modeling Impacts to Environment

To overcome the limitations, we first propose an *environment impact model*. The environment of the HNS is characterized by a set of *environment properties*. In our model, we say that a property e is *numeric* if e can be measured in number form. Typical numeric properties include `brightness` and `temperature`, whereas non-numeric ones are `fragrance` and `sound_content`.

An appliance method may give an *impact* to a certain property. For instance, turning on a light increases `brightness` by 100 lux. The proposed environment impact model defines such dynamics of impacts by a set of FSMs. Each FSM describes a single appliance associating the impacts with every state transition. Figure 1 shows an example of the impact model.

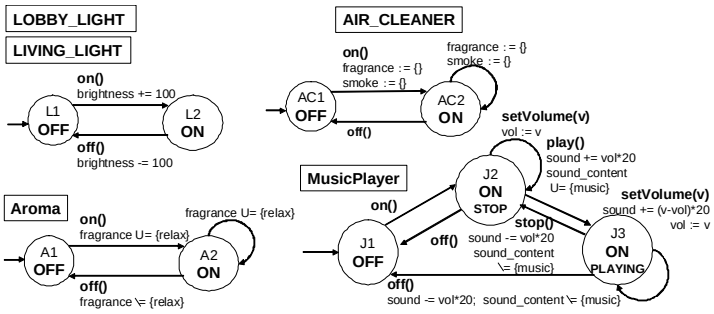

Figure 1. Environment Impact Model

We consider two kinds of impacts: *cumulative* and *immediate*. The cumulative impact represents an addition (or subtraction) to the present value (e.g., `brightness+=100`). The immediate impact specifies an direct value to be assigned (e.g., `temperature:=28`) In our model, a non-numeric property is regarded as a *set*. Hence, `sound_content U= {music}` represents a cumulative impact that adds a music to the present sound contents, while `fragrance :={}` is an immediate impact that removes any fragrance in the room.

An *environment state* is defined as a collection of current values of all environment properties given. When an integrated service s is executed, the environment state is updated according to the impact model. For a service s, we write $Estate(s)$ to represent an environment state obtained by the execution of s. In this paper, we assume that the execution of s is *atomic*, that is, no other service can be interleaved until s executes all the appliance methods.

B. Describing Requirements for Environment

In general, a user of an integrated service s *expects* some effects on the environment (as well as on appliances) provided by s. Usually such expectation has a range of acceptance. In the proposed method, we define the bottom line of the expectation for s to be *environment requirement*, denoted by $Ereq(s)$. A requirement is supposed to be given as a logical formula over environment properties.

For example, the user of BGM service requires that a music is played as a sound content of a room, represented by: `Ereq(BGM): music∈sound_content`. A requirement of TVT is that the room is enough dark and no other movie or sound is allowed: `Ereq(TVT):[brightness<80] &&[movie_content=={tv}]&&[sound_content=={tv}]`.

An environment requirement is evaluated to be true or false under a certain environment state. Since $Ereq(s)$ is the minimal expectation to s, so $Ereq(s)$ is supposed to be true under $EState(s)$, denoted by $EState(s) \vdash Ereq(s)$.

C. New Definition of Environment Interaction

We define that an environment interaction occurs when the execution of multiple services leads to a state where an environment requirement is not fulfilled.

[Environment Interaction:] Let s_1 and s_2 be integrated services. We say that s_1 and s_2 cause an environment interaction iff both of the following conditions hold:
(Condition B1:) $EState(s_i) \vdash Ereq(s_i)\ (i = 1, 2)$
(Condition B2:) $EState(s_1; s_2) \nvdash Ereq(s_1) \wedge Ereq(s_2)$,
where ';' denotes a successive execution of services.

IV. Example

For any pair of services in Section I, we detect environment interactions using the proposed method. By definition, interactions depend on execution order of services. We suppose the following environment requirements.

```
EReq(TVT): [brightness<80]&&movie_content=={tv}]&&
           [sound_content=={tv}]
EReq(CH):  fragrance.contains(relax)
EReq(BGM): sound_content.contains(music)
EReq(AC):  fragnance=={}
```

Then the following environment interactions are detected by the proposed method. Note that these interactions cannot be explained well by the previous methods, since they did not count the requirements, explicitly.

Interaction1 (TVT vs CH) CH turns on `LOBBY_LIGHT`, which violates `brightness<80` of `Ereq(TVT)`. It formally explains the interaction explained in Section I.

Interaction2 (TVT vs BGM) BGM adds `music` to `sound_content` of the room, which violates `sound_content=={tv}` of `Ereq(TVT)`. The interaction illustrates the fact that the BGM service disturbs the user watching the TV.

Interaction3 (Coming Home vs Air Cleaning) CH adds `relax` to `fragrance` of the room, but the fragrance is absorbed by AC. On the other hand, although AC tries to clean the air, CH keep adding the fragrance. Thus, AC violates `Ereq(CH)`, also CH violates `Ereq(AC)`.

V. Conclusion

This paper proposed a new definition of environment interaction in the HNS. The proposed method introduced the environment impact model, to represent cumulative or immediate impacts of the appliances to numeric or non-numeric environment properties. We also considered requirements to the environment to characterize the desirable or undesirable interactions, explicitly.

References

[1] M. Calder, E. Magill, M. Kolberg, and S. Reiff-Marganiec, "Feature Interaction: A Critical Review and Considered Forecast," *Computer Networks*, Vol. 41, No. 1, pp. 115–141, 2003.

[2] M. Nakamura, H. Igaki, and K. Matsumoto, "Feature Interactions in Integrated Services of Networked Home Appliances," *Proc. of ICFI'05*, pp. 236–251, 2005.

[3] T. Matsuo, P. Leelaprute, T. Tsuchiya, and T. Kikuno "Verifying Feature Interactions in Home Network Systems," IPSJ Journal, Vol. 49, No. 6, pp. 2129–2143, 2008.