Characterizing Safety of Integrated Services in Home Network System

Ben Yan[1], Masahide Nakamura[1], Lydie du Bousquet[2], Ken-ichi Matsumoto[1]

[1] Nara Institute of Science and Technology (NAIST)
8916-5, Takayama-cho, Ikoma-shi, Nara, 630-0192 Japan
{hon-e, masa-n, matumoto}@is.naist.jp
[2] LSR Laboratory, IMAG, Joseph Fourier University (Grenoble I)
BP72, F-38402, Saint-Martin d'Hères Cedex, France
Lydie.du-Bousquet@imag.fr

Abstract. This paper formalizes three kinds of safety to be satisfied by networked appliances and services in the emerging home network system (HNS). The *local safety* is defined by safety instructions of individual networked appliances. The *global safety* is specified as required properties of HNS services, which use multiple appliances simultaneously. The *environment safety* is derived from residential rules in home and surrounding environments. Based on the safety defined, we propose a modeling/validation framework for the safety. Specifically, we first introduce an object-oriented modeling technique to clarify the relationships among the appliances, the services and the home (environment) objects. We then employ the technique of *Design by Contract with JML (Java Modeling Language)*, which achieves systematic safety validation through testing.

1 Introduction

The recent ubiquitous/pervasive technologies allow general household appliances to be connected within the network at home. The *home network system* (HNS, for short) is comprised of such networked appliances to provide various services and applications for home users[7]. The great advantage of HNS lies in *integrating* (or orchestrating) features of multiple appliances, which yields more value-added and powerful services. We call such services *HNS integrated services*. For example, integrating a TV, a DVD player, lights, sound-systems and curtains implements a *DVD Theater service*, which allows a user to watch movies in a theater-like atmosphere just within a single operation.

In developing and providing a HNS integrated service, the service provider must guarantee that the service is *safe* for inhabitants, house properties and their surrounding environment. In the conventional situations where a user operates (non-networked) appliances one-by-one, the safety has been assured manually by the human user. That is, every user is supposed to follow *safety instructions* typically described in a user manual.

On the other hand, as for the HNS integrated services, we have to consider the safety more carefully. Since the service is typically implemented as a software application, appliances are often operated automatically by the application, but not by the human user. Also, one integrated service operates multiple appliances, which yields global dependencies among different appliances. Moreover, since multiple integrated services can be executed, unexpected functional conflicts may occur among the services. Thus, a single fault in the service application can cause serious accidents to the user. Unfortunately, no solid study has been reported for the safety of HNS integrated services.

In order to achieve the safety within the HNS integrated services systematically, this paper formalizes three kinds of safety: (a) *local safety*, (b) *global safety*, and (c) *environment safety*. The *local safety* is defined by the safety instructions of individual networked appliances. The *global safety* is specified as required properties of HNS services, which use multiple appliances simultaneously. The *environment safety* is prescribed as residential constraints and rules in home and surrounding environments.

Based on the safety formulated, we then propose a modeling/validation framework. Specifically, we introduce an object-oriented modeling technique to clarify the relationships among the appliances, the services and the home (environment) objects. We then employ the technique of *Design by Contract* [6] with *JML (Java Modeling Language)* [3, 9]. The properties of local, global and environment safety are represented as JML contracts, and embedded in Java source code of the appliance, the service and the home objects, respectively. Finally, the safety properties are validated through testing using related testing tools. Assuring safety is a crucial issue to guarantee high quality of life in smart home. We believe that the proposed framework provides a systematic approach to the safety assurance in the context of pervasive computing in smart home.

2 Preliminaries

2.1 Home Network System

A *home network system* (HNS) consists of one or more networked appliances connected within a LAN at home. In general, each appliance has a set of *application program interfaces* (i.e., APIs), by which the users or external software agents can control the appliance via the network. A HNS typically has a *home server*, which manages all the appliances in the HNS. Services and applications are installed on the home server. A *HNS integrated service* operates different multiple appliances together, and achieves a sophisticated and value-added service. An integrated service is implemented as a software application that invokes APIs of the appliances. It is supposed to be installed in the home server.

2.2 Example of HNS Integrated Services

We here introduce four example scenarios of HNS integrated services.

```
Public   DVD TheaterService {
         DigitalTV     tv = new DigitalTV();
         DVDPlayer  dvd = new DVDPlayer();
         SoundSystem  sound =new SoundSystem();
         Light   light = new Light();
         Curtain  curtain = new   Curtain();

         tv.on();                          /* Turn on TV */
         tv.setVisualInput('DVD');

         dvd.on();                         /* Turn on the DVD Player */
         dvd.setSoundOutput('5.1');

         sound.on();                       /* Turn on the Sound System */
         sound.setInputSource('DVD');
         sound.setVolumeLevel(25);

         curtain.closeCurtain();           /*  Close curtain  */

         light.setBrightnessLevel(1);      /*  Minimize brightness  */

         tv.playTv();                      /*  Play TV   */
         dvd.playDvd();                    /*  Play DVD  */
}
```

Fig. 1. DVD theater service

[**SS1： DVD Theater Service**] Integrating a TV, a DVD player, a sound system, a light and a curtain, this service automatically sets up the living room in a theater configuration. Upon a user's request, the TV is turned on with the DVD input, the curtains are closed, the sound system is configured for 5.1ch mode, the light becomes dark, and finally the DVD player plays back the contents.

[**SS2： Relax Service**] Integrating a DVD player, a sound system, a light, an air-conditioner, and an electric kettle, this service helps a user relax in the living room. When the user starts the service, the DVD player is turned on with a music mode, a 5.1ch speaker is selected with an appropriate sound level, the brightness of the light is adjusted, the air-conditioner is configured with a comfortable temperature, and the kettle is turned on with a boiling mode to prepare hot water for coffee.

[**SS3： Shower Service**] Integrating a gas-boiler, a shower valve and an air-conditioner in the bathroom, this service provides a comfortable setting for taking shower. Upon the service request, the gas-boiler is turned on with a preset water temperature. When the user enters the bathroom, the shower valve is automatically opened to turn on the shower. Also, the air-conditioner is turned on to keep the user warm.

[**SS4: Cooking Preparation Service**] Integrating a gas-valve, a ventilator, a roaster and a kitchen light, and an oven, this service automatically sets up the kitchen configuration of preparing for cooking. When requested, the kitchen light is turned on, the gas-valve is opened, the ventilator is turned on, and the pre-heating of the oven is started.

Fig. 1 shows a Java-like pseudo code which implements the scenario SS1 of DVD Theater service. In the figure, `X.Y()` means the invocation of API `Y()` of appliance `X`.

3 Formalizing Safety of HNS Integrated Services

3.1 Safety of HNS

For a HNS integrated service, we define the safety in the *broad sense* as follows.

Definition 1 (safety in broad sense). A HNS integrated service s is *safe* iff s is free from any condition that can cause [injury or death to home users and neighbors], or [damage to or loss of home equipments and the surrounding environment].

Our long-term goal is to establish a solid framework that guarantees the safety in Definition 1. In general however, it is quite difficult to achieve the 100% safety. Hence, the safety is often evaluated by means of *risk*. Thus, to assure the safety to a considerable extent, a set of conditions or guidelines minimizing the risk (called, *safety properties*) are usually considered [2]. In the following subsections, we investigate the safety properties specific to the domain of the HNS and the integrated services.

3.2 Local Safety Properties for Individual Appliances

For every electric appliance, the manufacturer of the appliance prescribes a set of *safety instructions* for proper and safe use of the appliance. Conventionally, these instructions have been designated for human users. However, in the HNS integrated service, the instructions must be guaranteed within the software. For instance, the following shows a safety instruction for an electric kettle.

> L1: Do not open the lid while the water is boiling, or there is a risk of scald.

Any integrated service using the kettle (e.g., SS2 in Section 2.2) must be implemented so that the service never opens the lid while the kettle is in the boiling mode. Other safety instructions include the installation issues. That is, every appliance must be installed in a proper environment described by its *specification*, including power voltage, rated current, power consumption, allowable temperature and humidity, etc.

Note that the safety instructions are a set of properties that are *locally* specified for each appliance. Thus we regard them as *local safety properties*. We assume that the local safety properties for an appliance are *determined by the vendor of the appliance*.

3.3 Global Safety Properties for Integrated Services

Since an integrated service orchestrates different multiple appliances simultaneously, it is necessary to consider *global properties* over the multiple appliances. For instance, SS3:Shower Service in Section 2.2 should guarantee the following safety property to prevent the user from getting scald or heart attack.

G1: When the service turns on the shower valve, the water temperature of the gas-boiler must be between 35 and 45 degree.

The next example shows a safety property for SS4:Cooking Preparation Service, which avoids carbon monoxide poisoning.

G2: While the gas valve is opened, the ventilator must be turned on.

Note that each of the properties is *globally* specified over multiple appliances. These *global safety properties* are usually service-specific, and are not covered by the local safety properties of individual appliances. Therefore, we suppose that the global safety properties are carefully *specified by the provider of the integrated service.*

3.4 Environment Safety Properties for House

In general, each house has a set of residential rules for inhabitants and neighbors to make a safe living. Since the integrated services give various impacts against the surrounding *environment* (including the room, the building, the neighbors, etc), the services must be safe against the environment by conforming to the residential rules. For instance, most house has a capacity of electricity, which yields the following safety property.

E1: The total amount of current used simultaneously must not exceed 30A.

Also for emergency, the following safety property should be concerned.

E2: Do not lock doors and windows in case of fire.

The following property may be derived from community rules.

E3: Do not make loud voice or sound after 9 p.m.

We assume that these safety properties are derived from the residential rules, including the house manual, the emergency procedure, community rules and policies, etc. We call such properties *environment safety properties.* Note that the environment safety properties are specified *independently* of appliances or services deployed in the HNS.

3.5 Safety Definition of HNS Integrated Services

Based on the discussion above, we define three kinds of safety as follows.

Definition 2 (safety of integrated service). Let s be a given integrated service, and

- let $App(s) = \{d_1, d_2, ..., d_n\}$ be a set of networked appliances used by s,

- let $LocalProp(d_i) = \{lp_{i1}, lp_{i2}, ..., lp_{im}\}$ be a set of local safety properties derived by the safety instructions of d_i,
- let $LocalProp(s) = \cup_{d_i \in App(s)} LocalProp(d_i)$,
- let $GlobalProp(s) = \{gp_1, gp_2, ..., gp_k\}$ be a set of global safety properties prescribed by s,
- let $EnvProp(s) = \{ep_1, ep_2, ..., ep_l\}$ be a set of environment safety properties derived from the environment where s is provided. Then,

Local Safety: s is *locally safe* iff s satisfies all properties in $LocalProp(s)$.
Global Safety: s is *globally safe* iff s satisfies all properties in $GlobalProp(s)$.
Environment Safety: s is *environmentally safe* iff s satisfies all properties in $EnvProp(s)$.
Safety: We say that s is *safe* iff s is locally, globally and environmentally safe.

We are now ready to formulate the safety validation problem.

Definition 3 (safety validation problem).

Input: A HNS h, an integrated service s, $LocalProp(s)$, $GlobalProp(s)$, and $EnvProp(s)$.
Output: A verdict whether s is safe or not within h.

4 Object-Oriented Modeling for Safety Validation

To conduct the safety validation problem systematically, this section presents an object-oriented model for the HNS and integrated services. Every networked appliance has the internal state (power status, driving mode, etc.) and the operational interfaces (i.e., APIs). Hence, it is reasonable to model each appliance as an *object* consisting of *attributes* and *methods*. Previously [5, 8], has developed an object-oriented model for networked appliances. In addition to the appliance object, here we newly introduce a *service object* for the integrated service and a *home object* for the home environment.

Fig. 2 shows the overview of the proposed model described as a UML class diagram. The model mainly consists of three kinds of objects (classes): `Appliance`, `Service`, and `Home`. As specified in the diagram, these classes forms the following relationships: [R1: a `Home` has multiple `Appliance`s], [R2: a `Home` has multiple `Service`s], and [R3: a `Service` uses multiple `Appliance`s]. These relationships match well the intuition of the HNS and integrated services.

4.1 Appliance Object

An appliance object models a networked appliance. The proposed model involves a super class `Appliance` and concrete appliance classes that inherit `Appliance`. The `Appliance` aggregates attributes and methods commonly contained in any kinds of electric appliances. It also has a `Specification`, which stores static specification information such as power voltage, rated current, size, allowable

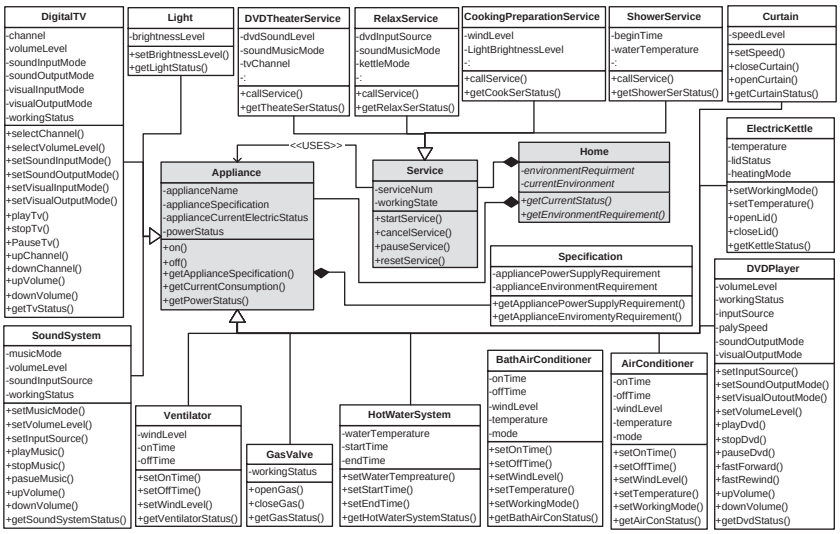

Fig. 2. Object-oriented model of HNS

temperature and humidity. Typical methods involve the power switch (`on()`, `off()`), acquisition of current power consumption (`getCurrentConsumption()`), and query for the specification (`getApplianceSpecification()`).

On the other hand, operations specific to each kind of appliance are specified in the individual sub-classes. Such methods include `TV.selectChannel()`, `DVD.playDvd()` and `Kettle.openLid()`. When a method of an appliance is executed, values of some attributes are changed, which updates the *current state* (i.e., the tuple of the current values of all attributes) of the appliance. Preferably, every appliance should have a mothod such as `TV.getTvStatus()` so that the current state can be referred by external objects.

4.2 Service Object

A service object models an integrated service, which uses several appliance objects depending on contents of the service. Similar to `Appliance`, there is a super class `Service` which aggregates common operations such as `startService()` and `cancelService()`. The concrete service scenarios are specified in sub-classes that inherit `Service`. For instance, `DVDTheaterService` (see SS1 in Section 2.2) uses appliances `DigitalTV`, `DVDPlayer`, `SoundSystem`, `Light`, and `Curtain`.

4.3 Home Object

A home object models the house that involves environmental attributes, which is is represented as a singleton object `Home`. The attributes of `Home` include the current energy consumption, sound level, brightness, temperature and humidity.

We assume that these attributes can be obtained or computed from the current states of appliances and services. For instance, the current temperature is suppose to be obtained via `Home.currentEnvironment.getTemperature()`. The current electricity consumption is supposed to be computed from specifications and states of appliances that are currently on.

5 Safety Validation Framework with Design by Contract

5.1 Key Idea: Using Design by Contract (DbC)

In order to achieve the safety validation, we apply a software design strategy, called *design by contract* (DbC, for short) [3, 6], to the object-oriented model presented in Section 4. For a given program, the DbC describes properties, conditions and invariants as a set of *contracts between calling and called objects*. The contracts are verified during runtime of the program under testing. During the execution, if a contract is violated, an exception is thrown or an error is reported. There are three kinds of contracts in the DbC.

Pre-Condition: A pre-condition of a method m is a condition that must be satisfied *before* executing m, which characterizes a premise of m.

Post-Condition: A post-condition of a method m is a condition that must be satisfied *after* executing m, which characterizes a consequence of m.

Class Invariant: A class invariant of a class c is a condition that must be guaranteed (i.e., kept unchanged) no matter which methods in c are executed.

Our key idea is to cope with the safety validation problem (see Definition 3) by first describing $LocalProp(s)$, $GlobalProp(s)$ and $EnvProp(s)$ as the DbC contracts, and then embedding them into the proposed object-oriented model. For this, we must consider carefully which object (`Appliance`, `Service`, or `Home`) should be responsible for $LocalProp(s)$, $GlobalProp(s)$ and $EnvProp(s)$.

5.2 Describing Local Safety Properties

Since the local safety properties are defined for individual appliance, `Appliance` or its sub classes should be responsible for $LocalProp(s)$. For instance, the local safety property L1 in Section 3.2 should be specified in `ElectricKettle` class, which can be encoded as the following contract:

```
Contractor:     ElectricKettle.openLid() method
Pre-condition:  heatingMode != 'boiling'
Post-condition: lidStatus == 'open' && heatingMode!='boiling'
```

The pre-condition is saying that; any service that executes the method `ElectricKettle.openLid()` must assure the kettle is not in the boiling status before executing the method. On the other hand, the post-condition prescribes that; the method must be implemented so that when completed, the lid is opened and the status does not change to boiling mode. Once the contract is broken, an exception is thrown to the HNS to conduct an appropriate action.

5.3 Describing Global Safety Properties

The global safety properties depend on the contents of each integrated service. Hence, it is reasonable to specify $GlobalProp(s)$ as DbC contracts in `Service` or its sub classes. For example, the global safety property G2 in Section 3.3 can be encoded as the following contract embedded in `CookingPrepatationService`:

```
Contractor:      CookingPreparationService class
Class Invariant: GasValve.workingStatus=='open'
                            ->Ventilator.powerStatus=='ON'
```

The above contract prescribes a condition that; at any time when the gas valve is opened, the ventilator must be turned on. This contract is a class invariant, which must be guaranteed no matter what operations are executed within the integrated service.

5.4 Describing Environment Safety Properties

Since the environment safety properties are derived from residential issues, `Home` class should be in charge of $EnvProp(s)$. For instance, the environment safety property E1 in Section 3.4 can be encoded as follows:

```
Contractor:      Home class
Class Invariant:
     home.currentEnvironment.getTotalConsumption()<=30
```

The method `getTotalConsumption()` is supposed to return the current total consumption of electricity, which is computed from the appliances that are being turned on. This contract is also an invariant, which must be assured whatever services or appliances are operated.

5.5 Implementing Safety Validation by JML

If the proposed model is implemented in the Java language, we can use the *JML* (*Java Modeling Language*) [3, 9] extensively for implementing the safety validation. The JML is a specification language that can be used to describe the DbC contracts in the form of Java comments, called *JML annotations*. The source code with the JML annotations is compiled by the JML compiler into *instrumented bytecode* implementing assertion-based checking routines of the DbC contracts.

Fig. 3 shows a JML annotation describing the contract mentioned in Section 5.2. In the figure, the contract is described as the annotation just above the method `openLid()`. The line starting with `requires` (or `ensures`) represents the pre-condition (or post-condition, respectively) of the contract. The word `spec_public` is for exporting the subsequent attribute to be used in the JML annotation.

Fig. 4 depicts the proposed framework of the safety validation. For given implementations of the appliance, service and home objects, a validator first

```
public class ElectricKettle {
  private /*@spec_public@*/ LidStatus  lid;
  private /*@spec_public@*/ HeatingMode hstate;
...
// JML Contract for openLid() [L1]
/*@ requires hstate != “boiling” ;
  @ ensures  lid = “open” && hstate != “boiling”;
  @*/
  public void openLid() {
    ... // Implementation
  }
...
}
```

Fig. 3. Safety description as JML contract (L1)

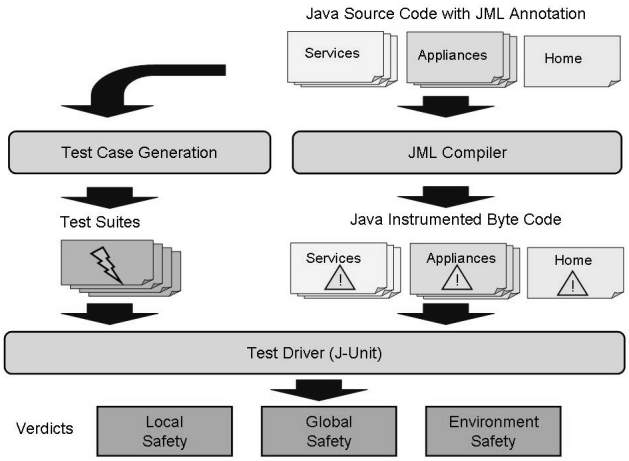

Fig. 4. Procedure of safety validation for HNS

describes safety properties to be validated in the JML annotation. Next, the annotated source codes are automatically compiled into instrumented bytecodes with the JML compiler. The validator also generates *test suites* against the HNS and the integrated services. For this, the validator develops the suites based on the logic and parameter values either manually or using test-case generation tools (e.g., TOBIAS [1, 4]). Finally, the instrumented bytecodes are validated against the generated test suites by the test driver. During the test, if any JML contract is broken, the test fails. The testing can be automated using any testing framework, e.g., JUnit [10]. Thus, the safety validation for the local, global and environment properties is achieved. Note that quality and efficiency of the validation process deeply depend on the test suites. Generating *good* test suites is beyond this paper and left for our future work.

6 Conclusion

In this paper, we have formalized the concept of safety in the context of HNS integrated services. Three kinds of safety are defined: local safety, global safety and environment safety. We have then proposed an object-oriented model and safety validation framework based on the DbC with JML. The properties of the local, global and environment safety are described as DbC contracts embedded in `Appliance`, `Service` and `Home` objects, respectively. Thus, the proposed framework can assist the HNS service vendors to develop safe integrated services, systematically.

In the future work, we will conduct experimental evaluation of the safety validation against the actual HNS. We also plan to examine effective test case generation techniques in the context of the HNS integrated services. The safety validation counting the feature interaction problem is also a challenging issue for our future research.

References

1. L. du Bousquet, Y. Ledru, O. Maury, and P. Bontron, "A case study in JML-based software validation, " Proceedings of 19th Int. IEEE Conf. on Automated Software Engineering (ASE'04), Linz, pages 294-297. IEEE Computer Society Press, Sep. 2004.
2. International Electrotechnical Commission, "Household and similar electrical appliances — Safety," IEC 60335-1, Sep.2006.
3. G. T. Leavens and Y. Cheon,"Design by Contract with JML, "Java Modeling Language Project,Internet: http://www.jmlspecs.org, 2003.
4. Y. Ledru, L. du Bousquet, O. Maury, and P. Bontron, "Filtering TOBIAS combinatorial test suites," In Proceedings of ETAPS/FASE'04 - Fundamental Approaches to Software Engineering, LNCS 2984, Springer-Verlag, Mar.2004.
5. P. Leelaprute, M. Nakamura, T. Tsuchiya, K. Matsumoto, and T. Kikuno, "Describing and Verifying Integrated Services of Home Network Systems," In Proc of 12th Asia-Pacific Software Engineering Conference (APSEC 2005), pp.549-558, Dec.2005.
6. B. Meyer, "Applying Design by Contract,"IEEE Computer,vol.25, no.10, pp.40-51, Oct.1992.
7. M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, K. Matsumoto, "Adapting Legacy Home Appliances to Home Network Systems Using Web Services," Proc. of Int'l Conf. on Web Services (ICWS 2006), pp.849-858, Sep.2006.
8. M. Nakamura, H. Igaki, and K. Matsumoto, "Feature Interactions in Integrated Services of Networked Home Appliances -An Object-Oriented Approach-," Proc. of Int'l. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI'05), pp.236-251, Jul. 2005
9. "The Java Modeling Language - JML," http://www.cs.iastate.edu/ leavens/JML/
10. "JUnit, Testing Resources for Extreme Programming," http://www.junit.org/