Deriving Safety Properties for Home Network System Based on Goal-Oriented Hazard Analysis Model

Ben Yan[1], Masahide Nakamura[2] and Ken-ichi Matsumoto[1]
[1]*Graduate School of Information Science, Nara Institute of Science and Technology,*
NAIST8916-5 Takayama, Ikoma, Nara 630-0101, Japan
Email: hon-e, matumoto@is.naist.jp
[2]*Graduate School of Engineering, Kobe University*
1-1 Rokkodai-cho, Nada, Kobe, Hyogo 657-0013, Japan
masa-n@cs.kobe-u.ac.jp

Abstract

The home network system (HNS, for short) is comprised of networked home appliances, which achieves various value-added services for home users. Assuring safety of the HNS and the services is a crucial issue. However, safety properties to be verified against the HNS have been given in an ad-hoc manner. This paper presents a systematic method that can derive the verifiable safety properties from a given HNS model and hazard contexts. Specifically, we first define a hazard analysis model for the HNS consisting of four levels of abstractions. We then conduct a goal-oriented analysis to specify logical relations between the adjacent abstraction levels. The analysis yields cause-and-effect chains from the abstract hazard contexts to the concrete attributes and operations of HNS objects (appliances, services, environment). Finally, the safety properties and their responsible operations are derived from the complete model, which give the strong rationale of the safety of the HNS.

1. Introduction

The emerging ubiquitous technologies allow general house-hold appliances to be *networked*. The *home network system* (HNS, for short) consists of such *networked appliances*, intended to provide various value-added services to home users. Owing to the network capability, the appliances, such as TVs, DVDs, air-conditioners, lights, curtains, ventilators, kettles, gas valves, baths, hot-water systems, are monitored, controlled, and even *orchestrated* from anywhere in the house even from outside [12].

In general, each service for the HNS is implemented as a software application (i.e., program), which automatically controls multiple appliances through the network. For instance, let us consider a service, say Cooking Preparation Service. This service automatically sets up the kitchen and related appliances for preparing for cooking, within just a single user's operation. When requested, the kitchen light is turned on, the gas-valve is opened, the ventilator is turned on, the oven becomes pre-heating mode, and the kettle is turned on in a boiling mode. In developing and providing such HNS services, the service provider must guarantee *safety* of the services. In the conventional (non-networked) home appliances, the safety has been assured manually by the human user, by means of *safety instructions* [2] described in the user's manual. However, in the HNS the safety must be assured in the software applications. Despite its importance, the safety issues within the HNS have not been studied well. In general, the safety issues of hazard system are usually considered at *(A) system designing stage* or *(B) system implementing stage*. Stage (A) always proposes a method for supporting

how to design a safe specification for a system. Stage (B) always proposes a method for *watching and preventing the system into the hazard state* or considering *what measures should be took for decreasing the damage after the accident happened*. For this paper, we consider the safety issues on the stage (A) and want to propose a systemic method for helping the designers design a safe HNS specification. In our previous work [17] [16], we have proposed a method that validates the safety properties against given HNS implementations, based on the concept of *design by contract* [3] [4] [11]. The safety of the HNS was characterized as three types of properties: (1) *local safety properties* are safety instructions of each individual appliance, (2) *global safety properties* are specified over multiple appliances to operate the HNS service safely, and (3) *environment safety properties* are residential rules in home and surrounding environments, independent of appliances and services. Table 1 shows examples
of the safety properties related to CookingPreparationService.

Table 1. Examples of safety properties for HNS

Safety Type	Safety Property
Local	$L1$: Do not open the lid while the water is boiling or there is a risk of scald (for ElectricKettle).
	$L2$: Make sure that the lid was closed before boiling water (for ElectricKettle).
Global	$G1$: While the gas valve is opened, the ventilator must be turned on (for CookingPreparationService).
	$G2$: Do not change the temperature above 45 degree while the shower is open (for ShowerService).
Environment	$E1$: The total amount of current used simultaneously must not exceed 30A.
	$E2$: Do not make loud voice or sound after 9 p.m.

However, the previous method assumes that all the safety properties are *given* by the analyst beforehand. Therefore, we had to specify the safety properties (including the ones in Table 1) manually in an ad-hoc manner. Some critical safety properties might be overlooked, which results in serious accidents in the HNS. To cope with this problem, this paper presents a requirement-engineering approach that can systematically derive the verifiable safety properties. Specifically, we first propose a *hazard analysis model* for the HNS, consisting of four levels of abstractions: (1) hazard context, (2) hazardous state, (3) object attribute and (4) object method. For a given HNS model and the hazard context, we then conduct a goal-oriented analysis to specify logical relations between the adjacent abstraction levels. The analysis yields cause-and-effect chains from the abstract hazard contexts to the concrete attributes and operations of HNS objects (appliances, services, and environment). Finally, the safety properties and their responsible operations are derived from the complete model, which give the strong rationale of the safety of the HNS.

2. Preliminaries

2.1. Home Network System (HNS)

A HNS consists of one or more *networked appliances* connected to a LAN at home. In general, each appliance has a set of *application program interfaces* (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 service* provides a sophisticated and value-added service by using multiple appliances together. The HNS service is implemented as a software application that invokes the APIs of the appliances. The appliances and services are deployed in a *home*, which is characterized by environmental attributes (e.g., temperature, humidity, brightness, current, sound, space).

2.2. Object-Oriented Model for HNS

Every component in HNS (i.e., appliance, service, or home) can be regarded as an *object* consisting of *attributes* and *methods*. Therefore, we have previously proposed an object-oriented model for the HNS [5] [13], which can be represented by a UML class diagram in Figure 1. The model consists of three kinds of objects (classes): Appliance, Service, and Home. These classes have relationships such that (a) a Home has multiple Appliances, (b) a Home has multiple Services, and (c) a Service uses multiple Appliances. These relationships reasonably characterize the structure of the HNS. For every object in the model, the attributes characterize the (internal) state of the object, while the methods represent operations (i.e., APIs) of the object. Executing a method may refer or update the values of some attributes. For instance, for an object ElectricKettle, the method on() updates the value of the attribute power to ON. Similarly openLid()updates the value of lid to OPEN.

2.3. Describing HNS Specification

To capture the given HNS model more clearly, we here introduce a language for describing the *specification* of the HNS. The language is originally defined in our previous research [5] [13], so full definition can be referred to the papers. Since the language aims to specify the appliance and the service models at the design (or requirements) level, no implementation specific information is contained.

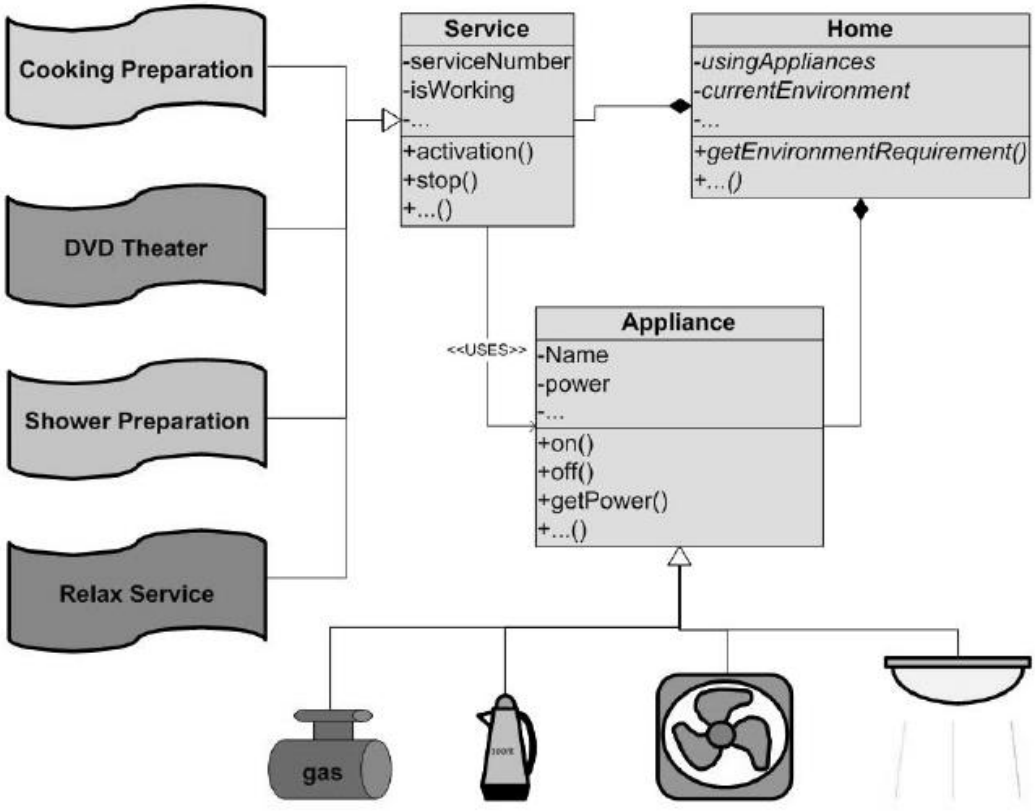

Figure 1. Object-oriented model of HNS

2.3.1. Appliance. As mentioned in Section 2.2, every appliance is characterized as an object consisting of *attributes* and *methods*. Figure 2 represents a specification of an electric kettle.

This specification says that the object ElectricKettle has four attributes and six methods. Each attribute is defined by associated type and initial value. In our language, integer (with allowable values), boolean, or enumerated type can be used. For instance, power can take two values ON or OFF, which is initialized to OFF. The attribute temperature holds a temperature setting value, to which the kettle warms up the water. Each method in the specification is simply characterized by a pair of logical formula over the attributes, namely, *precondition* and *post-condition*. The pre-condition of a method is a guard condition that must be satisfied *before* the method is executed. On the other hand, the post-condition is a resultant condition that must be satisfied *after* the method is executed. In our model, ordinary comparison and logical operators (==, !=, >, >=, <, <=, &&, ||, !) 1 are used. For instance, let us take openLid() method in Figure 2. To execute openLid(), the lid must be closed and the power must be on. After executing the method, the lid will be opened. The method setMode() takes a formal parameter md, intended that the working mode will be updated to the given md in the post-condition. Our specification language also can specify *invariants*. The invariant is a condition that must be satisfied all the time no matter which method is executed. In Figure 2, no specific invariant is given.

2.3.2. Service. The specification for HNS service can be specified in the almost same way. In addition to the case of the appliance, a service has a set of appliances used by the service. Figure 3 shows a specification of CookingPrepara tionService, which was introduced in Section 1. The specification says that this service uses five appliances — a ventilator, a gas system, a light, an oven, and an electric kettle. The attribute attr of each appliance app is denoted by app.attr. As specified in the post condition of activation(), when the service is activated, the light is turned on, gas valve is opened, ventilator is turned on, the oven becomes the pre-heating mode, and kettle is turned to the boiling mode.

2.4. Safety Property and Validation

An appliance (or a service) in the HNS is called *safe*, iff the appliance (or the service, respectively) is free from any condition that can cause (a) injury or death to home users and neighbors, or (b) damage to or loss of home equipments and the surrounding environment.

1. The operator notation follows those of the C (or Java) language

```
APPLIANCE ElectricKettle
 ATTRIBUTES
   power       : {ON, OFF} = OFF;
   mode        : {IDLE, BOILING, WARMING} = IDLE;
   lid         : {OPEN, CLOSE} = CLOSE;
   temperature : {60,80,90,95,98} = 98;
 METHODS
   on()  {
     PRE:  power == OFF;
     POST: power == ON;
   }
   off()  {
     PRE:  power == ON;
     POST: power == OFF;
   }
   openLid() {
     PRE:  lid == CLOSE && power == ON;
     POST: lid == OPEN;
   }
   closeLid() {
     PRE:  lid == OPEN && power == ON;
     POST: lid == CLOSE;
   }
   setMode(md) {
     PRE:  power == ON;
     POST: mode == md;
   }
   setTemperature(temp) {
     PRE:  power == ON;
     POST: temperature == temp;
   }
 INVARIANTS
    true;
```

Figure 2. Specification of ElectricKettle

```
SERVICE CookingPreparationService
  APPLIANCES
     vent      : Ventilator;
     gas       : GasSystem;
     light     : Light;
     oven      : Oven;
     kettle    : ElectricKettle;
  ATTRIBUTES
     isWorking : {true, false} = false;
     :
  METHODS
     activation() { //activate the cooking preparation
      PRE:  isWorking==false;
      POST: light.power==ON && gas.valve==OPEN &&
            vent.power==ON &&  vent.windLevel==3 &&
            oven.mode==PREHEAT && kettle.mode==BOILING &&
            kettle.temperature==100
     }
     stop() { //shutdown the preparation
       :
     }
  INVARIANTS
     true;
```

Figure 3. Specification of CookingPreparetionService

However, it is generally quite difficult to achieve 100% safety in any system. Hence, the safety is often evaluated by means of *risk*. To assure the safety to a considerable extent, a set of conditions or guidelines minimizing the risk are considered [2]. These conditions are so-called *safety properties*. In the context of the HNS, the safety properties are specified over the three kinds of HNS objects in the object-oriented model, considering potential hazards caused by the objects. The safety properties are often written in the natural language, as shown in Table 1. In our previous work [17] [16], we have defined the notion of *local*, *global*, and *environment* safety properties for the HNS. Specifically, **Local Safety Property:** A safety property *lp* is called a *local safety property* iff *lp* is defined within a single appliance *d* in the HNS. Typically, *lp* is derived as a safety instruction for using *d*.

Global Safety Property: A safety property *gp* is called a *global safety property* iff *gp* is defined over multiple appliances *d*1, *d*2, ..., *dn*. Typically, *gp* is defined as a functional (or non-functional) requirement of a HNS service that uses *d*1, ..., *dn*,simultaneously.

Environment Safety Property: A safety property *ep* is called an *environment safety property* iff *ep* is defined as the environmental or residential constraints, which exist independently of any appliances or services. The *safety validation* is to verify if the target system satisfies the *given* safety properties. The safety validation would be implemented by some V&V techniques, such as testing [16], model checking [5], theorem proving, etc. No matter which approach is taken, the correctness and completeness of the safety properties are a key factor for the successful safety validation. In our previous work, we assumed that all the safety properties are *manually given* by the analyst. Thus, how to give *correct* and *complete* properties is still an open issue, although it is quite a challenging problem.

3. Research Goal and Approach

The main problem in this paper is considering how to derive safety properties (such as the ones in Table 1) for given HNS specifications (in Figures 2 and 3), systematically. Moreover, the derived safety properties should be reflected in the (original) specifications, so that the safety properties are explicitly considered at the design level. Among the local, the global and the environment safety properties, we do not consider the environment property in this paper. By definition, every environment property heavily depends on the environmental factors, which cannot be captured
by the HNS model and specification. After all, the problem is formulated as follows:

Input:
- (I) *ASpec*: a set of appliance specifications, and
- (II) *SSpec*: a set of service specifications.

Output:
- (a) *LProp*: a set of local safety properties,
- (b) *Safe-ASpec*: a set of (safe) appliance specifications, where *Safe-ASpec* is a revision of *ASpec* with considering *LProp*
- (c) *GProp*: a set of global safety properties,
- (d) *Safe-SSpec*: a set of (safe) service specifications, where *Safe-SSpec* is a revision of *SSpec* with considering *GProp*.

To achieve the goal, we conduct a *hazard analysis*, which investigates potentially dangerous situations under the given HNS. To perform the analysis efficiently, we propose a *HNS hazard analysis model (HNS-HAM, for short)*. Using the HNS-HAM, we then derive the safety properties in a goal-oriented way.

4. Proposed Method

4.1. HNS Hazard Analysis Model

We propose a unique hazard analysis model, called *HNS-HAM*, consisting of four levels of abstractions. Figure 4 depicts the overview of the proposed model. The model starts with abstract types of hazards (we call *hazard contexts*), which are independent of specific HNS configuration. The hazard contexts (Level 1) are refined to *hazardous states* (Level 2), and then mapped to concrete *attributes* (Level 3) and *methods* (Level 4) of the HNS objects. The adjacent levels are linked by *logic relations*. The HNS-HAM specifies *cause-and-effect chains* [9] from the abstract hazard contexts to the concrete attributes
and operations of the HNS model. We explain the details of each level as follows. **(A) *Level1: Hazard Context Level.*** The top level of the HNS-HAM defines abstract types of hazards to be considered in the hazard analysis, which we call *hazard contexts*. Each hazard context must be independent of the specific HNS instances. Typical hazard contexts for the HNS include burn, scald, explosion, gas poisoning, flood, deficiency of oxygen, noise, etc. The purpose of Level 1 is to determine the scope of the hazard analysis. **(B) *Level2: Hazardous State Level.*** For each hazard context *hc* defined in Level 1, this level defines possible states in the given HNS, where the hazard *hc* is realized. We call such dangerous states *hazardous states*. In general, a hazard occurs due to several related factors. Also a hazardous state can be composed of fine-grained sub-states. So we characterize a hazard context *hc* by several hazardous states *hs*1, *hs*2, ..., *hsn* connected by logical operaters (AND, OR, NOT). Moreover, a hazardous state *hsi* can be decomposed into several sub-states *hsi*1, ..., *hsik*. For a hazardous state *hs*, if there is no more sub-state into which *hs* is refined, we call *hs* an *atomic hazardous state*.

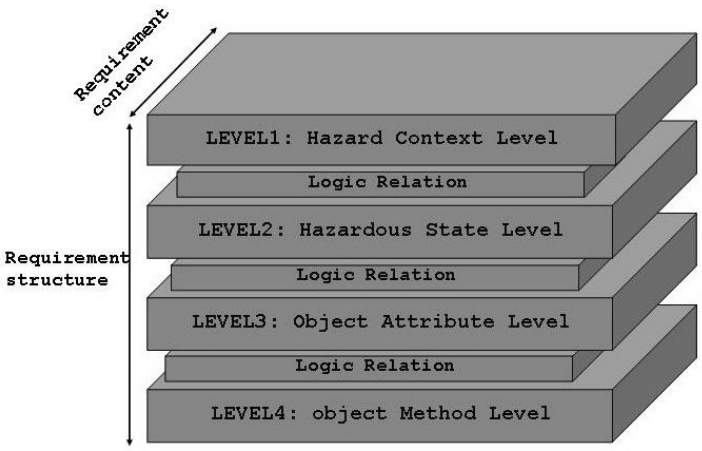

Figure 4. Structure of HNS-HAM

Figure 5 shows an example of HNS-HAM investigating the hazard context "scald" within ElectricKettle (let it be HC1). In the figure, a rectangle node represents a hazard context, an oval node represents a hazardous states (other nodes will be explained later). Each arrow from node *A* to *B* denotes a causal relationship that "*B* is caused by *A*". In this example, the possible cause of the scald is characterized by the AND composition of two states, "(HS1): the water temperature is too high", AND "(HS2): the lid is opened". In this example, HS1 is further decomposed into two sub-states: "(HS11): the kettle is in the boiling mode", OR "(HS12): the temperature setting is above 60". HS11, HS12 and HS2 are atomic hazardous states. The purpose of Level 2 is to map the general hazard context into concrete causes in the HNS currently focused on. In this level, we can see a condition under which the hazard context is realized (in the natural language).

(C) ***Level3: Object Attribute Level.*** This level encodes every atomic hazardous state defined in Level 2, in a formal condition over attributes of a HNS object. Since each hazardous state is somehow conceptual representation, this level transforms the state into rigorous expression in the HNS specification. In Figure 5, a round-box node represents a condition over attributes of `ElectricKettle`. For instance, the atomic hazardous state HS11 is encoded by an expression `mode==BOILING`. In this level, the hazard context can be captured in terms of as concrete attribute values of the HNS objects.

(D) ***Level4: Object Method Level.*** Finally, this level identifies object methods that can trigger the hazard context. More specifically, for each attribute condition *cond* in Level 3, we find methods m_1, m_2, ..., m_r that can make *cond* true. These methods can be easily identified by investigating post-condition of the methods defined in given HNS specifications. The purpose of this level is to clarify operations that must be anticipated for the safety assurance. In Figure 5, a node with brackets represents a method of `ElectricKettle` that makes a certain attribute condition true. For instance, we can see that executing `setMode(BOILING)` causes a condition `mode==BOILING`, as specified in the specification in Figure 2, and that the execution would be one factor causing scald.

4.2. Constructing HNS-HAM

The HNS-HAM is constructed by the following procedure. Note in the following that a HNS-HAM is constructed for every pair of a hazard context *hc* and a given specification *spec*.

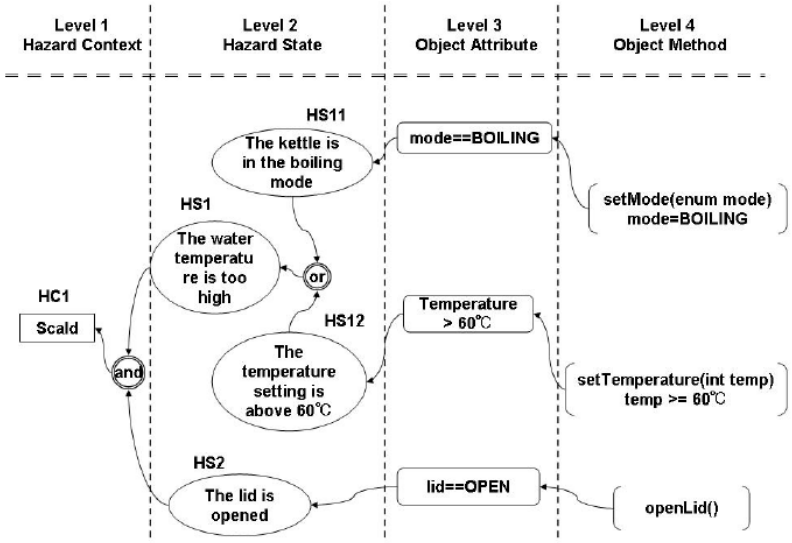

Figure 5. HNS-HAM for ElectricKettle

Step 1 (Definition of Hazard Contexts): Enumerate any hazard contexts that might occur in the given HNS. Since this step requires no technical aspect of the HNS specifications, any stakeholders can join the analysis.

Step 2 (Elaboration of Hazard States): Pick up a specification *spec* from *ASpec* (or *SSpec*). For each hazard context *hc*, characterize *hc* by some hazardous states within the object of *spec*. Then, decompose each hazardous state into sub-states in a goal-oriented fashion until all atomic states are obtained, which completes Level 2. Step 2 is the most important step that determines the quality of the HNS-HAM. Participation of experts in safety engineering would be encouraged to improve the completeness.

Step 3 (Mapping into Attributes Conditions): Encode each atomic hazardous state *hs* by a condition over object attributes based on *spec*. This constructs Level 3. If there is no attribute corresponding to *hs*, check if *hs* can be further decomposed. If *hs* is really atomic, then revise *spec*.

Step 4 (Obtaining Methods): For each attribute condition *cond*, find methods in *spec* that make *cond* true, by consulting the post-conditions of the methods. If the given specification is consistent, this step is not difficult.

4.3. Deriving Safety Properties with HNS-HAM

Suppose that a HNS-HAM *ham*(*o*, *hc*) is constructed with respect to a HNS object *o* (defined by *spec*) and a hazard context *hc*. Now we derive the safety properties that must be conformed by *o* to prevent *hc* from occurring. For this, we use Levels 1 and 2 of *ham*(*o*, *hc*), extensively. According to Levels 1 and 2 of *ham*(*o*, *hc*), *hc* is characterized by a logical formula *fhc* consisting of atomic hazardous states. If *fhc* holds, then the hazard *hc* is realized. Conversely, to prevent *hc* from occurring, we have to assure $\neg fhc$ for all the time. Thus, we want to derive the safety properties as a set of rules $R = \{r1, r2, ..., rn\}$, interpreted as a conjunction $\neg fhc = r1 \wedge r2 \wedge ... \wedge rn$. Using the *clausal normal form* [14] in the classical logic programming, we can obtain such a set $R = \{r1, ..., rn\}$ that $ri = (P1 \wedge ... \wedge Pm) \rightarrow (Q1 \vee ... \vee$

Qn), where *Px* and *Qy* are literals. Based on the idea, we derive the safety properties from a given HNS-HAM as follows.
Input: a HNS-HAM *ham*(*o, hc*) constructed for a HNS object *o* and a hazard context *hc*.
Step 1: From Levels 1 and 2 of *ham*(*o, hc*), derive a logical formula *fhc* = *f*(*hs*1, ..., *hsl*) characterizing *hc* by atomic
hazardous states *hsi* ($1 \le i \le l$).
Step 2: Calculate $\neg fhc$.
Step 3: Convert $\neg fhc$ into the clausal normal form $R = \{r1, ..., rn\}$.
Step 4: Define each *ri* as a safety property.

Let us derive safety properties for ElectricKettle using the HNS-HAM in Figure 5. According to Level 1 and 2, we get

$$fHC1 = (HS11 \vee HS12) \wedge HS2$$

Making a negation, and applying the De Morgan's and distribution lows, we obtain

$$\begin{aligned}\neg fHC1 &= \neg HS11 \wedge \neg HS12 \vee \neg HS2 \\ &= (HS11 \rightarrow \neg HS2) \wedge (HS12 \rightarrow \neg HS2) \\ &\wedge (HS2 \rightarrow \neg HS11) \wedge (HS2 \rightarrow \neg HS12)\end{aligned}$$

Thus, we derived the following four safety properties for ElectricKettle to prevent the scald from occurring.

(P1) $HS11 \rightarrow \neg HS2$**:** When the kettle is in the boiling mode, the lid must not be opened.
(P2) $HS12 \rightarrow \neg HS2$**:** When the setting temperature is higher than 60, the lid must not be opened.
(P3) $HS2 \rightarrow \neg HS11$**:** When the lid is opened, the kettle must not be in the boiling mode.
(P4) $HS2 \rightarrow \neg HS12$**:** When the lid is opened, the temperature setting must be below 60.

All of the above properties are quite reasonable as the safety instructions of an electric kettle. Note that the properties P1-P4 are all *local safety properties*, since they are closed within a single HNS appliance (i.e., Electric Kettle).

4.4. Updating HNS Specifications with Derived Safety Properties

Based on the safety properties derived, we update the original specification so that the safety properties are reflected. To achieve this, we use Levels 3 and 4 of HNS-HAM extensively. Each safety property is a condition over atomic hazardous states, and Level 3 specifies the correspondence between the atomic states and the object attributes of the model. So each safety property can be encoded by a condition using attributes. An encoded safety property can be specified as an *invariant* in the specification, intended that the property must hold all the time for the safety. Or, if the safety property is encoded by attributes in the same appliance, we can specify the property as pre/post-conditions of methods designated by Level 4. Let us update the specification of ElectricKettle in Figure 2, based on the HNS-HAM in Figure 5 and the safety properties P1 to P4 derived in Section 4.3. First we take (P1): $HS11 \rightarrow \neg HS2$. According to Level 3 of the HNS-HAM, (P1) is encoded to the following invariant over object attributes.

INVARIANT:

mode==BOILING -> !(lid==OPEN)

To satisfy the above invariant, we can refine the specification of method setMode() so as to check the lid status.

```
setMode(md) {
PRE: power==ON && lid!=OPEN;
POST: mode==md && lid!=OPEN;
}
```

The updated pre-condition says that setMode() can be executed only when the lid is closed. The post-condition means that setMode() never opens the lid by its execution. Thus, the unexpected boiling operation when the lid is opened can be avoided. Similarly, we can update the specifications for P2, P3 and P4.

4.5. Procedure of Proposed Method

Finally, we sum up the proposed method against the problem formulated in Section 3. If the proposed method is applied to a HNS appliance, the local safety properties are derived (as seen in the kettle example). If applied to a HNS service, the global safety property can be obtained.

(A) Safety Analysis for HNS Appliance: For each appliance *app* specified in *spec* $\in$ *ASpec*,

1) Define hazard contexts *hc*1, *hc*2 ... *hcx*.

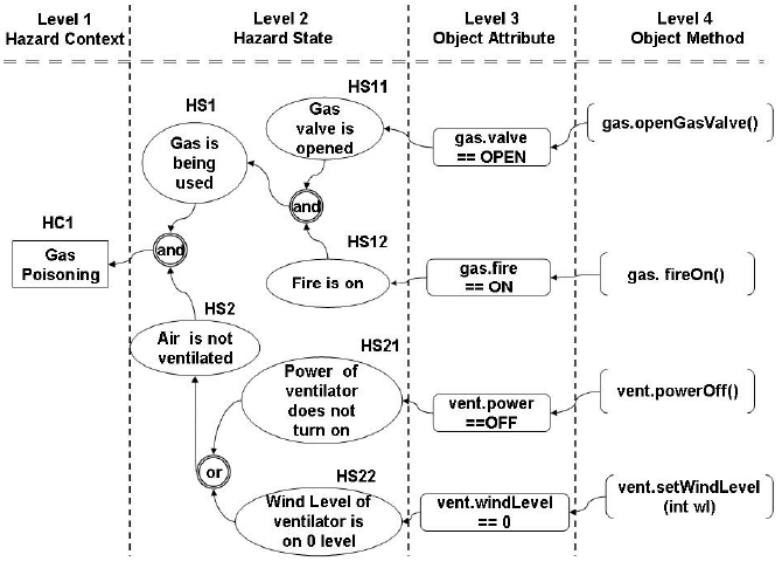

Figure 6. HNS-HAM for CookingRreparationService (Gas Poisoning)

2) For each hazard context *hci*, construct a HNS-HAM *ham*(*app*, *hci*).

3) Derive safety properties *p*1, *p*2, ..., *pn* from *ham*(*app*, *hci*). Put *p*1, ..., *pn* in *LProp*.

4) For all *pj* , update *spec*. Put the resultant *spec0* in *Safe-ASpec*.

(B) Safety Analysis for HNS Service: For each service *ser* specified in *spec* $\in$ *SSpec*,

1) Define hazard contexts *hc*1, *hc*2 ... *hcx*.

2) For each hazard context *hci*, construct a HNS-HAM *ham*(*ser*, *hci*).

3) Derive safety properties *p*1, *p*2, ..., *pn* from *ham*(*ser, hci*). Put *p*1 , ..., *pn* in *GProp*.
4) For all *pj* , update *spec*. Put the resultant *spec0* in *Safe-SSpec*.

5. Case study

As a case study, this section demonstrates analysis of CookingPreparationService, introduced in Section 1 and specified in Figure 3. Since this service uses a gas system, so let us consider a hazard context, "gas poisoning". Figure 6 shows a HNS-HAM for this case study. As seen in the model, the gas poisoning is caused by "(HS1) gas is being used", and "(HS2) air is not ventilated". Each of these states is further decomposed into two sub-states.
From the HNS-HAM, we get

$fHC1 = (HS11 \wedge HS12) \wedge (HS21 \vee HS22)$

Then, compute the negation

$\neg fHC1 = \neg(HS11 \wedge HS12) \vee (\neg HS21 \wedge \neg HS22)$
$= (HS11 \wedge HS12 \rightarrow \neg HS21) \wedge$
$(HS11 \wedge HS12 \rightarrow \neg HS22)$

From this, we obtain the following two safety properties:
(GP1) $HS11 \wedge HS12 \rightarrow \neg HS21$**:** When the gas valve is opened and the fire is on, the ventilator must be turned on.
(GP2) $HS11 \wedge HS12 \rightarrow \neg HS22$**:** When the gas valve is opened and the fire is on, the wind level of the ventilator must not be 0.
(GP3) $HS21 \rightarrow \neg HS11 \vee \neg HS12$**:** When the ventilator power is in off, the gas valve must not be opened or fire must not be on.
(GP4) $HS22 \rightarrow \neg HS11 \vee \neg HS12$**:** When the ventilator wind level is 0, the gas valve must not be opened or fire must not be on.

Note that these properties are global safety properties, since these are specified over different appliances (i.e., the gas valve and the ventilator). Using Levels 3 and 4 of the HNS-HAM, GP1 and GP2 are translated as the invariants in the specification.

```
INVARIANTS
(gas.valve==OPEN && gas.fire==ON
-> vent.power!=OFF) && //for GP1
(gas.valve==OPEN && gas.fire==ON
-> vent.windLevel!=0) //for GP2
```

6. Discussion

6.1. Summary and Future Work

In this paper, we have presented a method of deriving safety properties and specifications in the domain of the home network system. Using the proposed hazard analysis model, HNS-HAM, the potential risks leading to the hazard are analyzed systematically. As a result, safety properties and their responsible HNS objects are identified, which provides the strong rationale of the safety of the HNS.

The derived safety properties and specifications are then taken over to the implementation and testing phases. By concatenating our previous safety validation method [16] [17] after the proposed method, we can obtain a consistent HNS service development process with safety assurance. The current limitation of the proposed method is the *completeness* of the safety properties. Whether or not the derived safety properties are complete heavily depends on the construction process of the HNS-HAM (especially Step 2, see Section 4.2). We are currently investigating techniques in goal-oriented requirements engineering [6] [1] to develop more sophisticated construction method.

To evaluate the effectiveness of the proposed method, we are currently doing more experiments using actual HNS [12]. It is also interesting to apply the proposed method to other services other than the HNS.

6.2. Related Work

Traditionally, the safety issues have been addressed in *safety critical systems* [7] [8] [9] [10], such as aerospace systems and nuclear plants. Despite their importance, there are yet little research work in the ubiquitous computing area, including smart home. Compared to the ubiquitous applications, the safety critical systems are quite monolithic, where requirements and system configurations are not frequently changed. Thus, we needed alternative analysis models suitable for the object-oriented model.

The analysis using the HNS-HAM is similar to the *fault tree analysis (FTA)* [15]. However, compared to the conventional fault tree, the HNS-HAM has the four level of abstractions customized for the HNS model. Also, our approach is not applied to accidents that were already happened, which is different from the general FTA. Our idea of safety analysis in a goal-oriented way was originally motivated by the *goal-oriented requirements engineering*[6], which tries to find system requirements in a goal-oriented way. In this area, there is also a language called GRL [1] for the goal-oriented requirement analysis. Basically they are usually applied in the requirements stage where no design specification is developed yet. On the other hand, our problem setting is to find the non-functional requirements (i.e., safety), in the design and validation phase, assuming that the functional specifications of the HNS are available.

Acknowledgment

This research was partially supported by the Comprehensive Development of e-Society Foundation Software program of the Ministry of Education, Culture, Sports, Science and Technology, the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B)(No.18700062) and Scientific Research (B) (No.17300007), and by JSPS and MAE under the Japan-France Integrated Action Program (SAKURA).

References

[1] Goal-oriented Requirement Language (GRL), available from (www.cs.toronto.edu/km/GRL/).
[2] International Electrotechnical Commission, Household and similar electrical appliances — Safety, IEC 60335-1, Sep.2006.
[3] The Java Modeling Language (JML), available from (www.eecs.ucf.edu/ leavens/JML/).
[4] G. T. Leavens and Y. Cheon.: Design by Contract with JML, available from (www.jmlspecs.org), May.2006.
[5] P. Leelaprute, M. Nakamura, T. Tsuchiya, K. Matsumoto, and T. Kikuno.: Describing and Verifying Integrated Services of Home Network Systems, Proc. of 12th Asia-Pacific Software Engineering Conferences (APSEC 2005), pp.549-558, Dec.2005.
[6] E. Letier and A. van Lamsweerde.: Deriving Operational Software Specifications from System Goals, Proc. of 10th ACM S1GSOFT symposium on the Foundations of Software Engineering (FSE'10), Charleston, Nov.2002.

[7] N. G. Leveson.: Safeware: System Safety and Computers, Addison-Wesley, 1995.
[8] N. G. Leveson, K. A. Weiss.: A New Accident Model for Engineering Safer Systems, Journal of Safety Science, Vol.42, No.4, pp.237-270. Apr.2004,
[9] N. G. Leveson.: Safety in Integrated Systems Health Engineering and Management Proc. of Safety Science, Vol. 42, No. 4, Apr.2004.
[10] N. G. Leveson, K. A. Weiss.: Making Embedded Software Reuse Practical and Safe, Proc. of Foundations of Software Engineering, Nov.2004.
[11] B. Meyer.: Applying Design by Contract, IEEE Computer, vol.25, no.10, pp.40-51, Oct.1992.
[12] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. Matsumoto.: Adapting Legacy Home Appliances to Home Network Systems Using Web Services, Proc. of International Conferences on Web Services (ICWS 2006), pp.849-858, Sep.2006.
[13] M. Nakamura, H. Igaki, and K. Matsumoto.: Feature Interactions in Integrated Services of Networked Home Appliances -An Object- Oriented Approach-, Proc. of International Conferences on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI'05), pp.236-251, Jul.2005.
[14] R. Socher: Optimizing the clausal normal form transformation, Journal of Automated Reasoning, Vol.7, No.3, pp. 325-336, 1991.
[15] J. Xiang, K. Futatsugi, and Y. He.: Formal Fault Tree Construction and System Safety Analysis, Proc. of IASTED International Conferences on Software Engineering, pp.378-384, Feb.2004.
[16] B. Yan, M. Nakamura, L. du Bousquet, and K. Matsumoto.: Validating Safety for Integrated Services of Home Network System Using JML, Journal of Information Processing Society of Japan (IPSJ Journal), Vol.49, No.6, pp.1751-1762, Jun.2008.
[17] B. Yan, M. Nakamura, L. du Bousquet, and K. Matsumoto.: Characterizing Safety of Integrated Services in Home Network System, Proc. of 5th International Conferences on Smart homes and health Telematics (ICOST2007), pp.130-140, Jun.2007.