International Journal of Smart Home
Vol.3, No.2, April, 2009

Improving Reusability of Hazard Analysis Model with Hazard Template for Deriving Safety Properties of Home Network System

Ben Yan[1], Masahide Nakamura[2], Lydie du Bousquet[3], and Ken-ichi Matsumoto[1]
[1]*Graduate School of Information Science, Nara Institute of Science and Technology*
8916-5 Takayama, Ikoma, Nara 630-0101, Japan
[2]*Graduate School of Engineering, Kobe University*
1-1 Rokkodai-cho, Nada-ku, Kobe, Hyogo 657-8501, Japan
[3]*Joseph Fourier University (Grenoble I)*
BP72, F-38402, Saint-Martin d'H`eres Cedex, France
[1]*yanbenjp@gmail.com, matumoto@is.naist.jp*
[2]*masa-n@cs.kobe-u.ac.jp*
[3]*Lydie.du-Bousquet@imag.fr*

Abstract

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. 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. 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. As for the HNS integrated services, however, we have to consider the safety much more carefully. This paper presents a systematic method that can derive the verifiable safety properties from a given HNS model and hazard contexts. Specifically, we propose the HNS Hazard Analysis Model (HNS-HAM) for deriving safety properties with conducting a goal-oriented analysis. 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). Then the safety properties and their responsible operations are derived from the complete model, which give the strong rationale of the safety of the HNS. On the other hand, to make the proposed method more practical, it is necessary to save the analysis cost while appliances and services in the HNS are added or changed. For this, we have also proposed the notion of the hazard template, which characterizes the generic portion of the HNS-HAM. For every hazard context, the hazard template is supposed to be constructed once by the safety experts. The template can be reused for various kinds of the HNS objects for the common hazard context. The reusable templates make it possible to save the analysis cost and improve the quality of the HNS-HAM.

Keywords: *home network system, safety properties, Hazard Analysis Model, cause-and-effect chains, hazard template*

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. 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 each integrated services of the HNS, it is usually implemented as a software application (i.e., program), which automatically controls multiple appliances through the network.

For instance, let us consider a service, say CookingPreparationService. 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, and the kettle is turned on in a boiling mode 0. 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* 0 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 are usually considered at *online stage* or *offline stage*. Online stage 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*. Offline stage always proposes a method for *supporting how to design a safe specification for a system*. For our proposal, we consider the safety issues on the offline stage and want to propose a systemic method for helping the designers design a safe HNS specification.

In our previous work 0 0, we propose a novel framework for the safety, consisting of the following two contributions. The first contribution is to propose a method for validating the safety for HNS 0. First, we proposed Object-Oriented Model and specification language for describing HNS. Based on this model, we also defined three kinds of safety properties for HNS integrated services with considering the nature of the HNS and integrated services. They are (1) local safety properties which is the safety to be ensured by the safety instructions of individual appliances, (2) global safety properties which is specified over multiple appliances as required properties of an integrated service, and (3) environment safety properties which is prescribed as residential rules in the home and surrounding environment. Finally, we proposed a method for validating the three kinds of safety property with using the technique of DbC (Design by Contract) and JML (Java Modeling Language). The key idea is that we consider three kinds of safety property as contract and define them in the specification of the HNS between a provider and a consumer of an HNS object. Following this specification, the safety contracts are embedded in Java source code of the appliance (service, or home, respectively) objects with JML, then the safety properties can be validated through testing using related testing tools.

The second contribution is to propose a requirement-engineering approach 0 that can systematically 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: (1) hazard context level, (2) hazardous state level, (3) object attribute level and (4) object method level. 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. 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. Table 1 shows examples of the derived safety properties related to

HNS.

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 proposed methods in our previous work can derive and validate derived safety properties for a given HNS specification, systematically. The drawback of the requirement-engineering approach is that the construction of *HNS-HAM* poses not a little cost even for a single instance of the HNS. If appliances and services in the HNS are added or changed, the expensive analysis has to be performed again for the new configuration, basically. To make the proposed method more practical, it is necessary to save the analysis cost. To cope with this problem, this paper presents a hazard template to reuse the existing hazard analysis model for various instances of the hazard context.

2. Preliminaries

2.1. Object-Oriented model for 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).

To understand the HNS well, we proposed an object-oriented model for the HNS 0 0, 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. Every object consists of attributes and methods. These models 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 switchOn() updates the value of the attribute isWorking to true. Similarly openLid() updates the value of lid to OPEN.

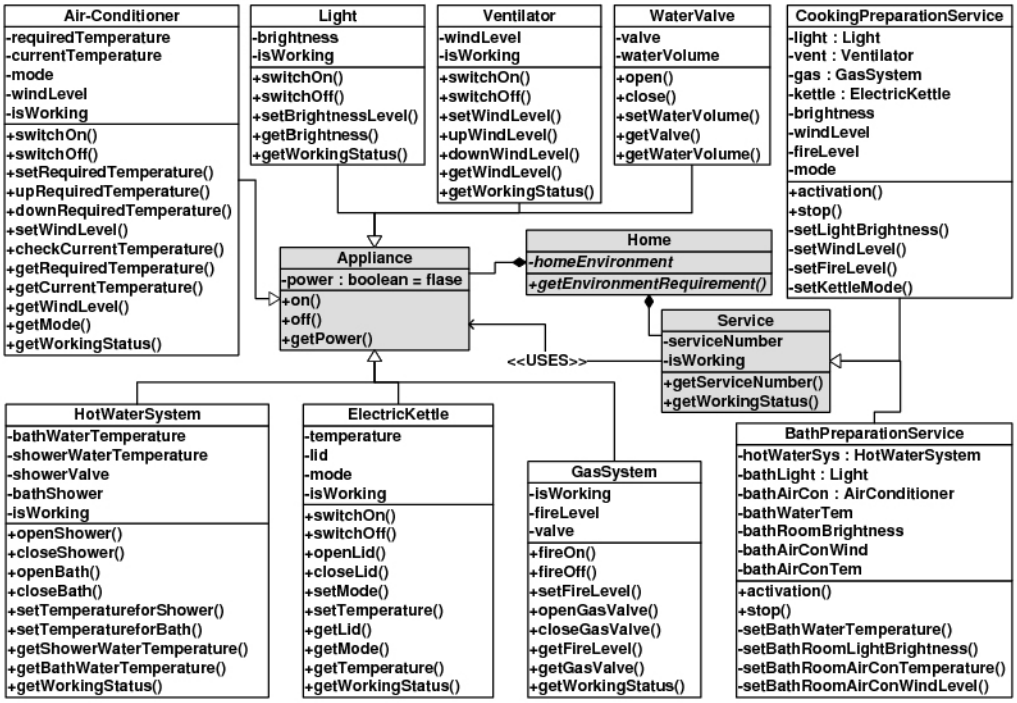

Figure 1. Object-oriented model of HNS

```
APPLIANCE ElectricKettle EXTENDS Appliance
ATTRIBUTES
   isWorking   : {true, false} = false;
   mode        : {IDLE, BOILING, WARMING} = IDLE;
   lid         : {OPEN, CLOSE} = CLOSE;
   temperature : {30,40,50,60,70,80,90,100} = 40;

METHODS
   switchOn()    { PRE: power == ON && isWorking == false; POST: isWorking == true; }
   switchOff()   { PRE: power == ON && isWorking == true;  POST: isWorking == false; }
   openLid()     { PRE: power == ON && lid == CLOSE;       POST: lid == OPEN; }
   closeLid()    { PRE: power == ON && lid == OPEN;        POST: lid == CLOSE; }
   setMode(md) { PRE: power == ON;                         POST: mode == md; }
   setTemperature(tem){ PRE: power == ON;                  POST: temperature == tem; }
   getLid()      { PRE: power == ON;                       POST: ReturnValve == lid; }
   getMode()     { PRE: power == ON;                       POST: ReturnValve == mode; }
   getTemperature()  { PRE: power == ON;                   POST: ReturnValve == temperature; }
   getWorkingStatus(){ PRE: power == ON;                   POST: ReturnValve == isWorking; }
INVARIANTS
   true;
```

Figure 2. Specification of ElectricKettle

2.2 Specification for describing appliance and service

To capture the given HNS model more clearly, we proposed a language for describing the *specification* of the HNS. The language is originally defined in our previous research 00 0, 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.

2.2.1 Appliance: As mentioned in Section 2.1, 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 ten 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, lid can take two values OPEN or CLOSE, which is initialized to CLOSE. 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, pre-condition 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.

2.2.2 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 CookingPreparationService, which was introduced in Section 1. The specification says that this service uses four appliances: a ventilator, a gas system, a light 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, and kettle is turned to the boiling mode.

2.3 The proposal for safety property 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. 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 0. These conditions are so-called *safety properties*.

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

```
SERVICE CookingPreparationService EXTENDS Service
  APPLIANCES
      vent        : Ventilator;
      gas         : GasSystem;
      light       : Light;
      kettle      : ElectricKettle;
  ATTRIBUTES
      Brightness              : {SOFT, MID, STRONG} = SOFT;
      windLevel               : {0, 1, 2, 3} = 1;
      fireLevel               : {ZERO, SOFT, MID, STRONG} = ZERO;
      mode                    : {IDLE, BOILING, WARMING} = IDEL;
  METHODS
      activation() {              //activate the cooking preparation
          PRE:    service.isWorking == false;
          POST:   light.power==ON && light.brightness==brightness && light.isWorking==true &&
                      gas.power==ON && gas.valve==OPEN && gas.fireLevel==fireLevel &&
                      gas.isWorking==true && vent.power==ON && vent.windLevel==windLevel &&
                      vent.isWorking==true && kettle.powervON && kettle.mode==workingMode &&
                      kettle.isWorking==true && service.isWorking==true;
      }
      stop() {                    //shutdown the preparation
          PRE:    service.isWorking==true;
            POST:   light.isWorking==false && light.power==OFF && gas.valve==CLOSE &&
                      gas.isWorking==false && gas.power==OFF && vent.isWorking==false &&
                      vent.power==OFF && kettle.isWorking==false && kettle.power==OFF &&
                      service.isWorking==false;
      }
      setLightBrightness(lb){ PRE: service.isWorking==false; POST: brightness==lb; }
      setWindLevel(wl) { PRE: service.isWorking==fasle; POST: windLevel==wl; }
      setFireLevel(fl) { PRE: service.isWorking==false; POST: fireLevel==fl; }
      setKettleMode(wm) { PRE: service.isWorking==false; POST: mode==wm; }
  INVARIANTS
      true;
```

Figure 3. Specification of Cooking Preparation Service

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 00, we have defined the notion of *local*, *global*, and *environment* safety properties for the HNS.

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, ..., d_n. Typically, *gp* is defined as a functional (or non-functional) requirement of a HNS service that uses d_1, ..., d_n, 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.

We also have proposed a method for validating the safety properties of HNS. The key idea of our method is to cope with the safety validation problem by first describing *LocalProp(s)*, *GlobalProp(s)* and *EnvProp* as the DbC contracts, and then embedding them into the proposed object oriented model. For a given program, the DbC describes properties, conditions and invariants as a set of contracts between calling and callee objects. The contracts are verified during the runtime of the program under testing. During the execution, if a contract is broken, an exception is thrown or an error is reported. Thus, if we could successfully represent the safety properties as DbC contracts among the HNS objects, then the safety validation problem can be reduced to the testing of the HNS implementations. The detail can be referred to the papers 0.

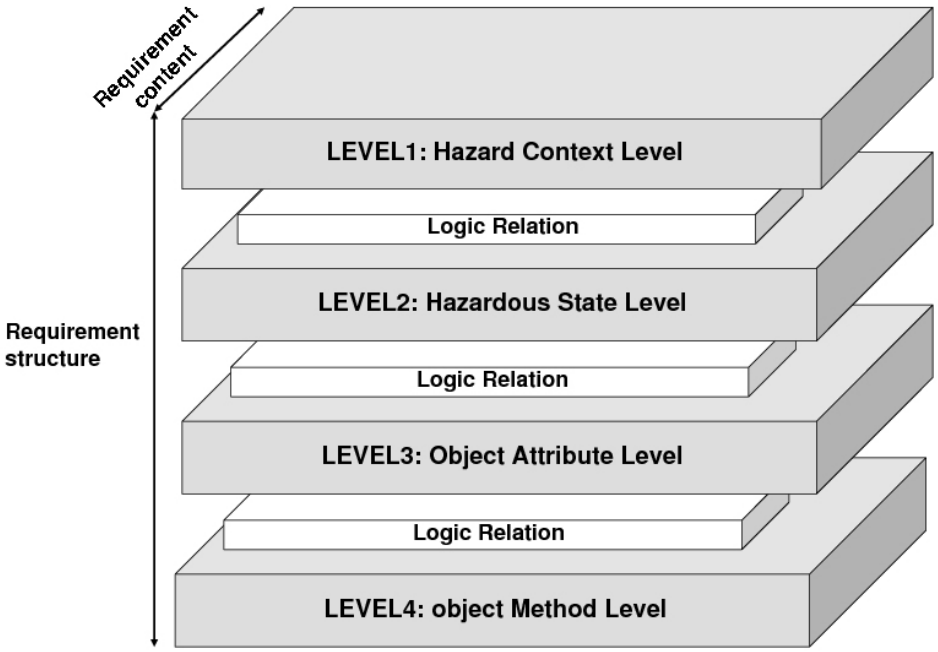

Figure 4. Structure of HNS-HAM

2.4 Requirement engineering approach for deriving safety properties

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 0, model checking 0, 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. If a safety property is not correctly specified, then the validation process yields a wrong verdict. If a critical safety property is overlooked, then the validation process ignores serious accidents in the HNS. If a safety property is quite irrelevant to the given HNS model, we should not put it into the expensive validation process. For this, we proposed a requirement-engineering approach for conducting 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. Moreover, the derived safety properties should be reflected in the (original) specifications, so that the safety properties are explicitly considered

at the design level. [2]The proposal is formulated as follows:

Input:

- (I) *ASpec*: a set of appliance specifications.
- (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*.[2]

2.4.1 HNS hazard analysis model: The hazard analysis model consisting of four levels of abstractions which in order to conduct the hazard analysis of the HNS. Figure 4 depicts the overview of the proposed model.

(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 operators (AND, OR, NOT). Moreover, a hazardous state hs_i can be decomposed into several sub-states hs_{i1}... hs_{ik}. For a hazardous state *hs*, if there is no more sub-state into which *hs* is refined, we call *hs* an atomic hazardous state. 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.

(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

2. 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.

the methods defined in given HNS specifications. The purpose of this level is to clarify operations that must be anticipated for the safety assurance.

Let us to see an example for how to construct *HNS-HAM* for the hazard context "scald by using kettle" within Electric Kettle (let it be HC1). In the Figure 5 , a rectangle node represents a hazard context, an oval node represents a hazardous state (other nodes will be explained later), a round-box node represents a condition over attributes of ElectricKettle, and a node with brackets represents a method of ElectricKettle that makes a certain attribute condition true. Each arrow from node *A* to *B* denotes a causal relationship that "*B* is caused by *A*".

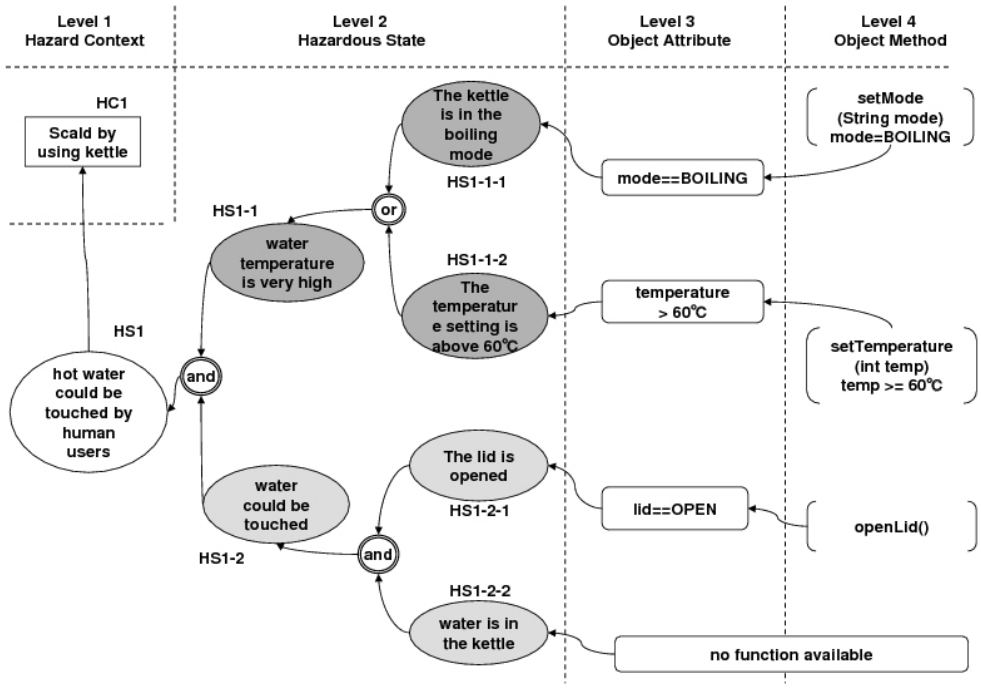

Figure 5. HNS-HAM for ElectricKettle

In this example, the possible cause of the scald is characterized by the hazard state "(HS1): hot water could be touched by human users". Moreover, the hazard state (HS1) is characterized the AND composition of two states, "(HS1-1): water temperature is very high" AND "(HS1-2): water could be touched". Moreover, "HS1-1" and "HS1-2" can be further decomposed into sub-states.

For "HS1-1", it is characterized the OR composition of "(HS1-1-1): the kettle is in the boiling mode" OR "(HS1-1-2): the temperature setting is above 60". For "HS1-2", it is characterized the AND composition of "(HS1-2-1): the lid is opened" AND "(HS1-2-2): water is in the kettle". HS1-1-1, HS1-1-2, HS1-2-1 and HS1-2-2 are atomic hazardous states. For one of the atomic hazardous state HS1-1-1 is encoded by an expression mode==BOILING, and the method setMode(BOILING) can causes a condition mode==BOILING with executing, as specified in the specification in Figure 2 , and that the execution would be one factor causing scald. Just by this thinking, we constructed the *HNS-HAM* for the "scald by using kettle" within ElectricKettle (see Figure 5).

2.4.2 Deriving safety properties with HNS-HAM: After we constructed the *HNS-HAM*, we should to derive the safety properties with this model. Let us to 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 f_{hc} consisting of atomic hazardous states. If f_{hc} holds, then the hazard ***hc*** is realized. Conversely, to prevent *hc* from occurring, we have to assure $\neg f_{hc}$ for all the time. Thus, we want to derive the safety properties as a set of rules $R = \{r_1, r_2, ..., r_n\}$, interpreted as a conjunction $\neg f_{hc} = r_1 \wedge r_2 \wedge ... \wedge r_n$. Using the *clausal normal form* 0 in the classical logic programming, we can obtain such a set $R = \{r_1, ... , r_n\}$ that $r_i = (P_1 \wedge ... \wedge P_m) \rightarrow (Q_1 \vee ... \vee Q_n)$, where P_x and Q_y 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 $f_{hc} = f(hs_1, ..., hs_l)$ characterizing ***hc*** by atomic hazardous states $hs_i (1 \leq i \leq l)$.
Step 2: Calculate $\neg f_{hc}$.
Step 3: Convert $\neg f_{hc}$ into the clausal normal form $R = \{r_1, ..., r_n\}$.
Step 4: Define each r_i as a safety property.

Let us derive safety properties for ElectricKettle using the *HNS-HAM* in Figure 5. According to Level 1 and Level2, we get

$$f_{HC1} = \text{HS1-1} \wedge \text{HS1-2} = (\text{HS1-1-1} \vee \text{HS1-1-2}) \wedge (\text{HS1-2-1} \wedge \text{HS1-2-2})$$

Since there is no function for HS1-2-2 in specification, we need not consider HS1-2-2 here, then

$$f_{HC1} = (\text{HS1-1-1} \vee \text{HS1-1-2}) \wedge \text{HS1-2-1}$$

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

$$\begin{aligned}\neg f_{HC1} &= \neg \text{HS1-1-1} \wedge \neg \text{HS1-1-2} \vee \neg \text{HS1-2-1} \\ &= (\text{HS1-1-1} \rightarrow \neg \text{HS1-2-1}) \wedge (\text{HS1-1-2} \rightarrow \neg \text{HS1-2-1}) \\ &\wedge (\text{HS1-2-1} \rightarrow \neg \text{HS1-1-1}) \wedge (\text{HS1-2-1} \rightarrow \neg \text{HS1-1-2})\end{aligned}$$

Thus, we derived the following four safety properties for ElectricKettle to prevent the scald from occurring.
(P1) HS1-1-1 $\rightarrow$ $\neg$HS1-2-1: When the kettle is in the boiling mode, the lid must not be opened.
(P2) HS1-1-2 $\rightarrow$ $\neg$HS1-2-1: When the setting temperature is higher than 60, the lid must not be opened.
(P3) HS1-2-1 $\rightarrow$ $\neg$HS1-1-1: When the lid is opened, the kettle must not be in the boiling mode.
(P4) HS1-2-1 $\rightarrow$ $\neg$HS1-1-2: 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).

2.4.3 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 2.4.2. First we take (P1): HS1-1-1 → ¬HS1-2-1. According to Level 3 of the HNS-HAM, (P1) is encoded to the following invariant over object attributes.

INVARIANT: Mode == BOILING → !(lid == OPEN)

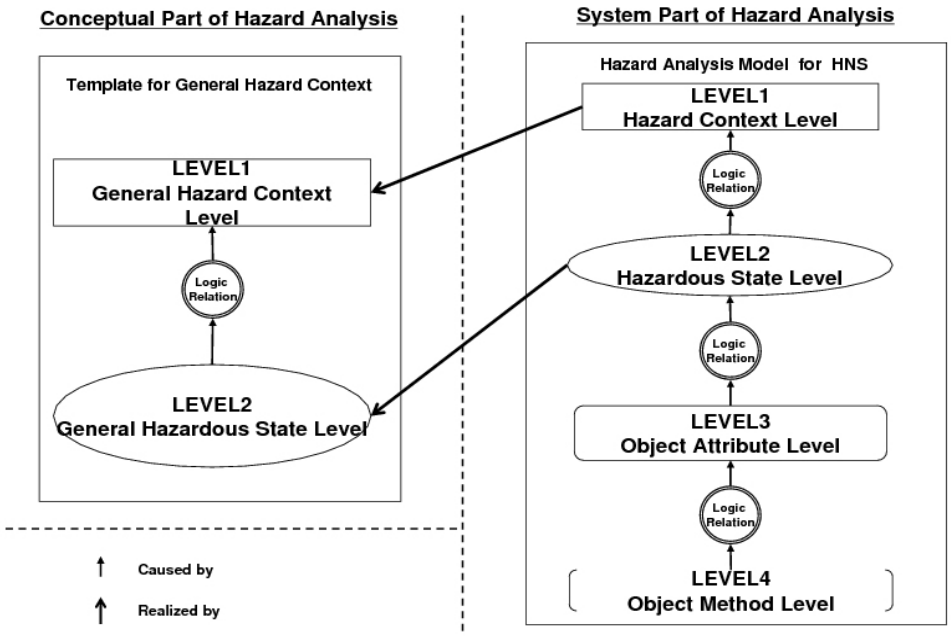

Figure 6. Structure and relation between hazard template and HNS-HAM

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.

3. Improving reusability of HNS-HAM with hazard template

Since here, we can understand that the proposed method with the *HNS-HAM* can derive safety properties for a given HNS specification, systematically. The drawback is that the construction of the *HNS-HAM* poses not a little cost even for a single instance of the HNS. If appliances and services in the HNS are added or changed, the expensive analysis has to be performed again for the new configuration, basically. To make the proposed method more practical, it is necessary to save the analysis cost. To cope with the problem, we try to *reuse* the existing hazard analysis model for various instances of the HNS.

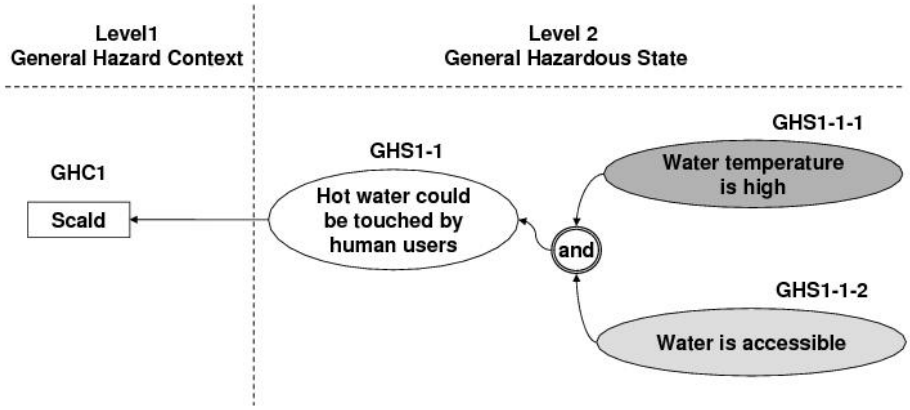

Figure 7. Hazard template for scald

3.1. Hazard template for HNS-HAM

Based on the analysis of *HNS-HAM*, we can understand that the most expensive (but essential) part of the proposed method is in the construction of Level 2. In the *HNS-HAM*, Level 1 deals with the quite general context, where as Level 3 is quite specific to the given HNS instance. Level 2 plays a role of *bridge* between general and specific notions, which complicates the issue.

Our key idea is to find, *within the Level 2, generic hazardous states that are independent of specific HNS configuration, and then to extract the generic portion as hazard template to be reused.*

Figure 6 shows the relationship between the *hazard template* and the *HNS-HAM*. The hazard template has two levels of abstraction, where a general hazard context is explained by general hazardous states. Within the template, these two levels are linked by the "caused-by" relation (denoted by solid arrows), as in the original *HNS-HAM*. Then, the Levels 1 and 2 of the template are respectively linked to those of the *HNS-HAM* by the "realized-by" relation (depicted by dotted arrows).

Let us recall the example of the scald in Figure 5. If we consider the causes of the scald from the most general viewpoint, the following condition appears:

(GHS1-1:) Hot water can be touched by a human user.

GHS1-1 is further decomposed into the following two conditions.

(GHS1-1-1:) Water temperature is high.

(GHS1-1-2:) Water is accessible.

These conditions can be represented as a *HNS-HAM* shown in Figure 7. Here we should note that the model contains *general hazardous states* independent of any specific appliances or services. Therefore, we use the model as a *hazard template* for analyzing the scald.

For a given HNS specification, the hazard template is refined into concrete hazardous states according to the specification. Since the template can be reused for any HNS appliances or services, the cost for constructing Level 2 is significantly reduced.

For instance, the template in Figure 7 can be used for constructing the *HNS-HAM* in Figure 5, whose context is "scald by using ElectricKettle". In the construction, GHS1-1-1 is realized by concrete states HS1-1 connected by HS1-1-1 or HS1-1-2, in accordance with the specific context of ElectricKettle. Similarly, GHS1-1-2 is realized by HS1-2 connected by HS1-2-1 and HS1-2-2. Thus, we can see that the structure of the hazard template is *inherited* in the *HNS-HAM*.

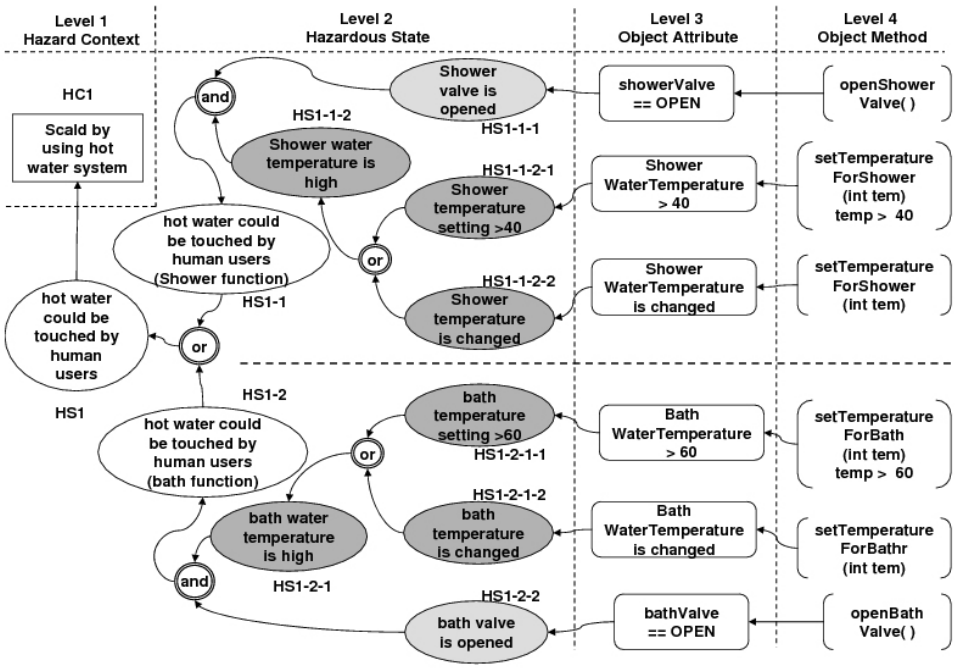

Figure 8. HNS-HAM for HotWaterSystem (scald)

We assume that every hazard template is constructed carefully by experts in the safety engineering. Once good templates are obtained for various hazard contexts, the cost for constructing concrete HNS-HAMs will be significantly reduced. Moreover, the derived HNS-HAMs will be more reliable and consistent.

4. Case study

As a case study, this section demonstrates analysis of HotWaterSystem with reusing the template shown in Figure 7. Since this appliance is hot water system, so let us consider a hazard context, "scald by using hot water system". Figure 8 shows the *HNS-HAM* for this case study.

As seen in the model, the template in Figure 7 can be used for constructing the *HNS-HAM* in Figure 8, whose context is "(HC1): scald by using hot water system". In the construction,

the hot water system has two functions for shower and bath, so "(GHS1-1-1): Water temperature is high" is realized by concrete states HS1-1-2 or HS1-2-1 and "(GHS1-1-2): Water is accessible" is realized by concrete states HS1-1-1 or HS1-2-2.

It means that the generate states "(GHS1-1-1): Water temperature is high" can be realized by using the hot water system while this appliance is in the "(HS1-1-2): Shower water temperature is high" state or in the "(HS1-2-1): Bath water temperature is high" state. The generate states "(GHS1-1-2): Water is accessible" can be realized by using the hot water system while this appliance is in the "(HS1-1-1): Shower valve is opened" state or in the "(HS1-2-2): Bath valve is opened" state. Based on the specification of this appliance, the possible cause of HS1-1-2 was characterized by two atomic sub-states:

"(HS1-1-2-1): Shower temperature setting > 40" OR
"(HS1-1-2-2): Shower temperature is changed".

The possible cause of HS1-2-1 was characterized by two atomic sub-states:

"(HS1-2-1-1): Bath temperature setting > 60" OR
"(HS1-2-1-2): Bath temperature is changed".

Since here, each of these states can not be further decomposed into other sub-states and the *HNS-HAM* for Figure 8 is completed with above analysis.

From the HNS-HAM of Figure 8, we get

(1) f_{HC1} = HS1-1-1 $\wedge$ (HS1-1-2-1 $\vee$ HS1-1-2-2) or
(2) f_{HC1} = HS1-2-2 $\wedge$ (HS1-2-1-1 $\vee$ HS1-2-1-2)

Then, compute the negation for formula (1)

$$\neg f_{HC1} = \neg \text{HS1-1-1} \vee (\neg \text{HS1-1-2-1} \wedge \neg \text{HS1-1-2-2})$$
$$= (\text{HS1-1-1} \rightarrow \neg \text{HS1-1-2-1}) \wedge (\text{HS1-1-1} \rightarrow \neg \text{HS1-1-2-2}) \wedge$$
$$(\text{HS1-1-2-1} \rightarrow \neg \text{HS1-1-1}) \wedge (\text{HS1-1-2-2} \rightarrow \neg \text{HS1-1-1})$$

Compute the negation for formula (2)

$$\neg f_{HC1} = \neg \text{HS1-2-2} \vee (\neg \text{HS1-2-1-1} \wedge \neg \text{HS1-2-1-2})$$
$$= (\text{HS1-2-2} \rightarrow \neg \text{H1-2-1-1}) \wedge (\text{HS1-2-2} \rightarrow \neg \text{HS1-2-1-2}) \wedge$$
$$(\text{HS1-2-1-1} \rightarrow \neg \text{HS1-2-2}) \wedge (\text{HS1-2-1-2} \rightarrow \neg \text{HS1-2-2})$$

From this, we obtain the following four safety logical formulas:

(P5) HS1-1-1 $\rightarrow$ $\neg$ HS1-1-2-1: While the shower valve is opened, the shower water temperature setting must not be above 40.

(P6) HS1-1-1 $\rightarrow$ $\neg$ HS1-1-2-2: While the shower valve is opened, the shower water temperature must be not changed.

(P7) HS1-1-2-1 $\rightarrow$ $\neg$ HS1-1-1: While the shower water temperature setting is above 40, the shower water valve must not be opened.

(P8) HS1-1-2-2 $\rightarrow$ $\neg$ HS1-1-1: While the shower water temperature is changed, the shower water valve must not be opened.

(P9) HS1-2-2 $\rightarrow$ $\neg$ HS1-2-1-1: While the bath valve is opened, the bath water temperature setting must not be above 60.

(P10) HS1-2-2 $\rightarrow$ $\neg$ HS1-2-1-2: While the bath valve is opened, the bath water temperature must be not changed.

(P11) HS1-2-1-1 $\rightarrow$ $\neg$ HS1-2-2: While the bath water temperature setting is above 60, the

bath water valve must not be opened.
(P12) HS1-2-1-2 $\rightarrow$ $\neg$ HS1-2-2: While the bath water temperature is changed, the bath water valve must not be opened.

Using Levels 3 and 4 of the *HNS-HAM*, we can translate all the derived safety property into the specification. For instance, P5 and P6 can be translated as the POST Conditions in the specification.

```
openShower(){
    PRE:    power    == ON && showerValve == CLOSE && showerIsWorking ==false;
    POST:   showerValve == OPEN && showerIsWorking == true &&
            showerValve == OPEN → showerWaterTemperature <= 40 &&
            showerWaterTemperature == old(showerWaterTemperature);
}
```

Due to the space of this paper, we only show an example for derived local safety properties here. For the detail about how to derive the global safety properties, you can be referred to the papers 0.

5. Summary

In order to derive reasonable safety properties, this paper presented a requirement-engineering approach that can systematically derive the verifiable safety properties for the HNS. Specifically, we proposed a hazard analysis model, called HNS Hazard Analysis Model (HNS-HAM), which consists of four levels of abstractions. By constructing the HNS-HAM and investigating potential hazards within the given HNS model. The safety properties can be derived systematically.

To improve the reusability of HNS-HAM, we have also proposed the notion of the hazard template, which characterizes the generic portion of the HNS-HAM. For every hazard context, the hazard template is supposed to be constructed once by the safety experts. The template can be reused for various kinds of the HNS objects for the common hazard context. The reusable templates make it possible to save the analysis cost and improve the quality of the HNS-HAM.

By using the proposed requirement-engineering approach, the potential risks leading to the hazard are analyzed systematically. As a result, the safety properties and their responsible HNS objects are identified. This provides the strong rationale of the safety for the HNS.

To evaluate the effectiveness of the proposed method, we have shown the case study with some practical appliances. We have derived 12 local safety properties for ElecricKettle and HotWaterSystem. We are currently doing more experiments using actual HNS 0. It is also interesting to apply the proposed method to other services other than the HNS.

Acknowledgment

This research was partially supported by: the Japan Ministry of Education, Science, Sports, and Culture, Grant-in-Aid for Young Scientists (B) (No.200803767)] and [JSPS and MAE under the Japan-France Integrated Action Program (SAKURA)].

References

[1] 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.
[2] International Electrotechnical Commission, Household and similar electrical appliances --- Safety, IEC 60335-1, Sep.2006.
[3] 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.
[4] B. Yan, M. Nakamura and K. Matsumoto.: Deriving Safety Properties for Home Network System Based on Goal-Oriented Hazard Analysis Model, International Journal of Smart Home (IJSH), Vol.3, No.1, pp.67-79, Jan.2009.
[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] 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.
[7] 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.
[8] Goal-oriented Requirement Language (GRL), available from (www.cs.toronto.edu/km/GRL/).
[9] The Java Modeling Language (JML), available from (www.eecs.ucf.edu/ leavens/JML/).
[10] G. T. Leavens and Y. Cheon.: Design by Contract with JML, available from (www.jmlspecs.org), May.2006.
[11] 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.
[12] N. G. Leveson.: Safeware: System Safety and Computers, Addison-Wesley, 1995.
[13] 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,
[14] N. G. Leveson.: Safety in Integrated Systems Health Engineering and Management Proc. of Safety Science, Vol. 42, No. 4, Apr.2004.
[15] N. G. Leveson, K. A. Weiss.: Making Embedded Software Reuse Practical and Safe, Proc. of Foundations of Software Engineering, Nov.2004.
[16] B. Meyer.: Applying Design by Contract, IEEE Computer, vol.25, no.10, pp.40-51, Oct.1992.
[17] R. Socher: Optimizing the clausal normal form transformation, Journal of Automated Reasoning, Vol.7, No.3, pp. 325-336, 1991.
[18] 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.

Authors

Ben Yan received the B.E. degree in Henan University of Science and Technology, China, in 1999, M.E. degree in Department of information Science Okayama University of Science, Japan, in 2006, and Ph.D. degree in the Graduate School of Information Science at Nara Institute of Science and Technology, Japan, in 2008. He is currently working for NttData Sanyo System Corporation. His research interests include the service-oriented architecture, the V&V of home network systems, and requirements engineering for safety critical systems.

Masahide Nakamura received the B.E., M.E., and Ph.D. degrees in Information and Computer Sciences from Osaka University, Japan, in 1994, 1996, 1999, respectively. From 1999 to 2000, he has been a post-doctoral fellow in SITE at University of Ottawa, Canada. He joined Cyber media Center at Osaka University from 2000 to 2002. From 2002 to 2007, he worked for the Graduate School of Information Science at Nara Institute of Science and Technology, Japan. He is currently an associate professor in the Graduate School of Engineering at Kobe University. His research interests include the service oriented architecture, Web services, the feature interaction problem, V&V techniques and software security. He is a member of IEEE, the ACM and IEICE.

Lydie du Bousquet received a ”Magistere d’Informatique” and a ”Diplome d’Etudes approfondies” from Claude Bernard University, Lyon and ”Ecole Normale Superieure de Lyon” in 1996. She was awarded a Ph.D. in Computer Science from Joseph Fourier University, Grenoble, France, in 1999. After one year as postdoctoral position at IRISA, Rennes, France, she became an associated professor at Joseph Fourier University, in 2000. Her main search interests focus on validation of safety critical systems (among with home networks systems) with testing approaches.

Ken-ichi Matsumoto received the B.E., M.E., and Ph.D. degrees in Information and Computer sciences from Osaka University, Japan, in 1985, 1987, 1990, respectively. Dr. Matsumoto is currently a professor in the Graduate School of Information Science at Nara Institute of Science and Technology, Japan. His research interests include software metrics and measurement framework. He is a senior member of the IEEE, and a member of the ACM, IPSJ and JSSST.