Modeling and Formal Verification of Smart Environments

SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2012; **00**:1–14
DOI: 10.1002/sec

Special Issue Paper

Modeling and Formal Verification of Smart Environments

Fulvio Corno, Muhammad Sanaullah*

DAUIN - Dipartimento di Automatica ed Informatica
Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy.

ABSTRACT

Smart Environments (SmE) are a growing combination of various computing frameworks (ubiquitous, pervasive etc), devices, control algorithms and a complex web of interactions. It is at the core of user facilitation in a number of industrial, domestic and public areas. Based on their application areas, SmE may be critical in terms of correctness, reliability, safety, security etc. To achieve error-free and requirement-compliant implementation, these systems are designed resorting to various modeling approaches including Ontology and Statecharts. This paper attempts to consider correctness, reliability, safety and security in the design process of SmE and its related components by proposing a design time modeling and formal verification methodology. The proposed methodology covers various design features related to modeling and formal verification SmE (focusing on users, devices, environment, control algorithms and their interaction) against the set of the requirements through model checking. A realistic case study of a Bank Door Security Booth System (BDSB) is tested. The results show the successful verification of the properties related to the safety, security and desired reliable behavior of BDSB.

KEYWORDS

User behaviour;Smart Environments; Intelligent Domotic Environment; Modeling; Formal Verification; Model Checking; Temporal Logic

*__Correspondence__
email: muhammad.sanaullah@polito.it, Tel.: (+39) 011-090-7191
Politecnico di Torino, Torino, Italy.

Received . . .

1. INTRODUCTION

Smart Environments (SmE) are gradually being introduced and employed in almost every nook and corner of our daily lives, such as homes, hospitals, offices, industries, airports, railways, transportation mediums and many public places. They promise to deliver intelligent services by considering the presence and actions of users [1, 2]. For specific services, users can interact with the system in any manner and at any time. The users can belong to different demographic groups, possess different behaviors and can interact with the system as they please [3]. It is opportune for SmE to react to users' behavior for delivering the services intelligently. In addition to users, behavior of the services is also dependent upon their application domain. For example, energy management is fundamental in industries; easy environment management is important in smart homes; assisted living is needed by the elderly in hospitals. Moreover, the safety and security services are essential requirements for many SmE, and depend upon the context (interchangeably mentioned as 'environment') and domains of the application [4, 5, 6]. For example, the safety service in the case of fire is to switch on the security alarms, unlock and open the emergency exit doors, turn on the emergency and path-pointing lights directing people towards the emergency exit, make recorded calls to nearby fire and rescue offices and other key officials of respective environment; whereas the security requirement for accessing the bank is achieved by crossing two automatically locked doors, in which, one door will not be opened until the other door is closed.

These services can be achieved by controlling the functionalities of the associated devices, which is performed through gateways at which all the devices are attached, by using some wired or wireless medium, and the computational elements (also known as Control Algorithms –which are responsible for the achievement of specific constraints of safety, security, availability and reliable behavior) that are installed on such gateways [7, 8, 9].

Due of the intricate communication between the SmE components (users, devices, computational elements, environment) along with the implementation of various constraints, the behavior of SmE becomes complex [4, 5]. As each of these components are self independent, with their own working capacities and internal behaviors [10], their independent and interactive nature introduce complex details in the system, and as a result, the likelihood of design errors (or failures) may increase [11]. For sensitive systems like fire control units, theft or traffic control systems, nursing care houses and others, where the failure can break security measures and may cause criticality, design time verification is recommended by various authors [11, 12, 13].

A number of approaches can be adopted for the design time verification with their own strengths and weaknesses [14, 15]. Ideally, a technique is required which may ensure the exhaustive verification of various requirements. Since complexity and ambiguity are usually the common features of such systems, formal verification processes help to root them out, and, in result, a reliable secure system can be designed, which has all the desired features and consistency among its integrated components with the environment [14]. As, formal methods promise holistic design time verification based on the following strengths: 1) they are strongly based on mathematical evidence and increase the understandability of the modeled system; 2) they are used for reliably modeling a system at design time; 3) they can model the concerning requirements in the form of properties by using logic based on mathematics; 4) they can formally verify the modeled system against the requirements (reliable behaviour, along with other requirements of the system); 5) they can trace back the errors and can help in fixing them at early design stages.

In this paper, an extended formal methodology from our previous work [16], for the modeling and verification of SmE, is proposed. The current proposal encapsulates the existing technique with a detailed and structured methodology, which is this paper's novelty. The proposal incorporates the users' and environment modeling in the current extension, with a detailed and multi-dimensional verification, which are added features to previous technique (capable of performing device and their control verification). Built upon the core features, the current methodology constructs the overall SmE where the users and environment are also considered, modeled and verified. The modeling of each component of SmE is performed from requirement gathering to the entire system verification, along with the security and safety constraints at design time.

The methodology uses DogOnt (a semantic web solution for the formal modeling of SmE through Ontologies) [17], SCXML (a textual Statechart [18] formalism for the behavioral modeling of SmE related components) [19], UCTL (Temporal Logic formalism for specifying the requirements in the form of properties) [20] and UMC (a model checker designed for the verification of interacting State-machines) [21] as tools. The Bank Door Security Booth System (BDSB) case study [16] is accordingly extended with users and environment configuration and has been successfully verified. The results demonstrate the feasibility of the methodology by which the security-critical SmE systems can be verified at design time.

In the remaining paper, the related work is presented in Section 2; the state of the art and the tools used for implementing the methodology are summarized in Section 3.1; an architectural overview by considering the operational flow in SmE is presented in section 3.2; the proposed methodology is presented in Section 4; the technique designed for implementing the methodology is described in Section 5; the description of the Bank Door Security Booth System (BDSB) is given in Section 6; the requirements formalism in the form of properties with their verification results are given in Section 7; discussion about the results is given in section 8 and finally, the concluding remarks and future works are given in Section 9.

2. RELATED WORK

The extended literature review did not find considerable amount of work in the area of formal modeling and verification of the users, devices, context (which is related to the environment configurations) and control algorithms of SmE altogether. Different researchers are concentrating on different areas for the formal modeling and verification of SmE and its related components. These dimensions can be classified as: device modeling, devices interaction and their control modeling, users' behavior modeling, users and their interaction modeling with the devices, and context modeling, jointly known as 'SmE Modeling Dimensions'. A precise description of these dimensions is presented below. Table I presents the literature and the dimensions followed within.

Table I. Modeling Dimensions covered by the Techniques

Papers	SmE Modeling Dimensions
[5]	Context, Devices Interaction and Control
[10]	Device
[11]	User, Context, User and Devices Interaction
[13]	Devices Interaction and Control
[16]	Devices Interaction and Control
[21]	User, User and Devices Interaction
[22]	Context, Devices Interaction and Control
[23]	User, Context, User and Devices Interaction
[24]	User, User and Devices Interaction
[25]	Devices Interaction and Control

In the device modeling category, the modeling and verification of the individual devices is performed according to their specified (reliable, safe, secure) behavior [10]. Whereas, in devices interaction and their control modeling category, the modeling and the availability of interaction services and functionalities are verified

Security Comm. Networks 2012; **00**:1–14 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

along with ensuring the specified constraints of the systems, which are controlled through some sophisticated algorithms. The device interaction and their control modeling are carried out in the following literature [5, 13, 16, 22, 25].

In users' behavior modeling category, the user model is confirmed to fulfill all the desired activities and the associated actions in the specified manner by following the environment imposed constraints. As each user has a different behavior (demographic age groups), therefore, in this category, it is also ascertained that the users model can incorporate all these behaviors and can achieve the desired activities by performing the required actions in a secure way. The formal modeling and verification of users is performed in the following literature [11, 21, 23, 24].

In users and their interaction modeling with the devices category, the interaction of the users with the devices is confirmed by verifying that the system is able to consider the users actions and whether or not the system can perform all the desired operations in a secure manner as a result. The work presented in the following literature [11, 21, 23, 24] cover this category.

In context modeling category, the concerning element is environment which changes its configurations when any change occurs in the system. Here the availability of the associated services is verified, along with the satisfaction of security and safety constrains, when the system changes its configuration at some particular state. The works of [5, 11, 22, 23] cover this category.

The proposed methodology tries to consider and incorporate these dimensions for the modeling and verification of SmE.

3. BACKGROUND

In the following sub-section 3.1, the tools required for the implementation of the technique are concisely explained, and in sub-section 3.2, the architecture of SmE is explained by focusing on the operation flow between each component.

3.1. State of the art and Adopted Tools

The interface (black-box) modeling can be performed with the use of Object Oriented paradigm, Semantic Web technologies or by other means. Ontology is a semantic web artifact for providing a mechanism to store the concepts and their corresponding relationships with the related characteristics to others concepts. Ontological solutions are recommended by various authors [26, 27, 28, 29, 30] for the modeling of SmE. Similarly, behavioral (white-box) modeling can be performed with the use of Statecharts [18], Petri-nets [31], Communicating Sequential Processes (CSP) [32] or other ways.

DogOnt [17] is an Ontology that provides formal modeling and suitable reasoning facilities to smart environments through semantic web technologies. The main focus is on the interface (also referred as black-box) modeling of the devices, where the states (at which a device can be at any time), functionalities (the capabilities of the device), commands (triggering the functionalities) and notifications (acknowledgment after the completion of the task) of the connected devices are modeled.

The behavior (also referred as white-box) modeling of systems, in which the activities are performed with the exchange of messages from one state to another, and in the systems where there are more than one destination states from the source-state depending upon the conditions, can be represented through Statecharts [18, 21, 33, 34]. The proposed methodology adopts Statecharts for the modeling of complex behavior of SmE and its related components. The modeling of devices behavior, in the form of Statecharts, is performed with the help of DogOnt. Every device modeled in DogOnt have a corresponding behavioral model as a Statechart. The input/output (interface) of the Statechart model are consistent with the information available in DogOnt, while the internal states and transitions can be described according to the actual device behavior [10]. This behavior modeling is performed by adopting the W3C standard SCXML language [19]. The other components of SmE, such as users' behavior, environment and control algorithms are also modeled in the Statecharts format and can be used for the verification through simulation, emulation and model checking.

For the verification of the systems modeled in the form of Statecharts, various model checking tools like [35, 36, 37, 38] are designed. UMC [21] is one of these model checking tools. It is an "on-the-fly" model checker exhaustively verifies the requirements, either concerning to *ACTL* (Action Based Branching Time Logic) [39] or *CTL* (State Based Branching time logic) [40], whereas, various other model checkers can support only one type of logic (state or action based). "State Space Explosion" is a major problem for the verification of complex systems, which UMC overcomes by not generating the global model of the system [21] but by exploiting a linear complexity model checking algorithm for the exhaustive verification of the system. An on-line version of UMC model checker is also available*.

Temporal Logics [40, 39, 41, 42] are widely used in formal verification for formalizing and specifying the requirements of complex systems. The truth value of these specified requirements depend upon time: whether the specific requirement will be true at any path (Exists) or on all the paths (All). In addition to Exists and All, there are other temporal quantifiers like Global, Next, Future, Until, Implies, etc, which help in verifying the complex requirements on different branches (of Statecharts) from some specific state at a certain time.

UCTL [20] is a UML-oriented branching-time temporal logic, which has the combined power of *ACTL* and *CTL*. Due to the rich set of state propositions and action

*http://fmt.isti.cnr.it/umc/

expressions, UCTL is best fitted for the verification of communicating state machines. With the help of UCTL, we can verify different properties like liveness (something good eventually happens), security (nothing bad can happen) and properties with or without the fairness restrictions. UCTL uses box *necessary* and *possible* operators from Hennessy-Milner Logic and temporal operators (like Until, Next, Future, Globally, All, Exists) from CTL/ACTL. By combining these logics, it can check the Absence, Existence, Bounded Existence, Universality and others of any *state* and *action* predicate. UMC is capable of accepting the properties specified in UCTL [20] format.

3.2. Architecture of Smart Environments

An architecture of SmE is designed, which is focusing on the operation flow in SmE system, as shown in Figure 1. In this architecture, the operational flow of SmE is classified in four layers: goals, actions, decisions and operations. Goals are the desires of the users which they want to achieve with the help of SmE. For achieving a certain goal, users have to perform some specific actions. The actions can be sensed through sensors, or they can be input by directly performing the action on the devices, or can be commanded by using the designed APIs of the SmE (through various computing devices).

When user perform any action a notification message (or a set of message) is sent to the control algorithms, where the concerning requirements related to the safety, security and reliable behavior of SmE are incorporated. Control algorithms act as sophisticated bridge between the input actions and the output operations. Against each incoming message, the current configuration of the system and devices is considered, and according to the incorporated constraints, a decision for the specific operations (services) is made. Further, on the basis of these decisions, control algorithms send the commands to the devices for performing the decided operation. The devices, according to their current configuration and internal constraints, perform the specific operations and acknowledge back about the status of the operation to the control algorithms (these acknowledgments are also considered as notifications).

As devices are from different manufacturers and follow different communication protocols and naming conventions, it is recommended to filter the unnecessary messages and if needed, convert the concerning messages following a recognized convention before sending them to control algorithms. By following this process, the modeling complexities and ambiguities of control algorithms can be slightly reduced.

4. PROPOSED METHODOLOGY

A comprehensive methodology is proposed for the design and verification of SmE with specific focus on system security. The methodology entails all the major components of SmE; users, devices, environment and control algorithms. It is advisable that for designing the SmE, the detailed specifications of these components are listed at requirement gathering phase. The organized specifications provide a better understandability of the system (and its related components) through which the ambiguities during modeling can be sufficiently reduced. Further, these organized specifications help in designing the properties related to the verification of reliable behavior (consisting of safety, security and other major aspects) of the system. For the behavioral modeling of each component, the methodology adopts Statecharts. The methodology provides an nine step process, explained below:

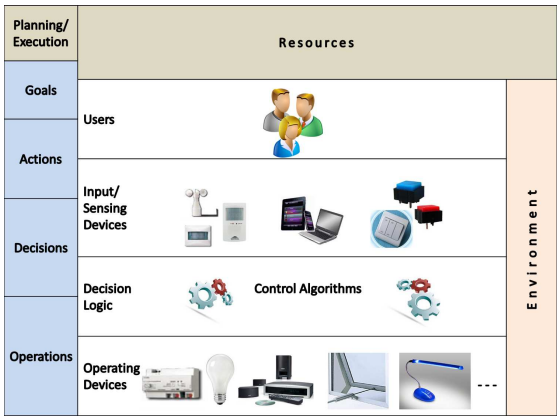

Figure 1. An architecture of Smart Environments

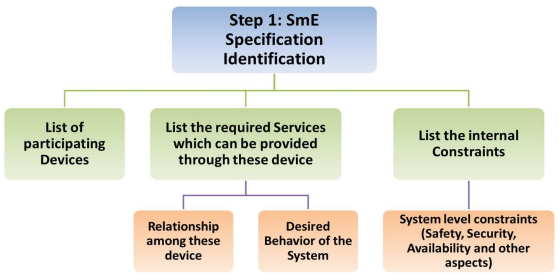

Figure 2. Step 1: SmE Specification Identification

Step 1:*SmE Specification Identification*
Requirements gathering and listing in a suitable way is normally the first step from where any complex project begins [43]. The same process is adopted for the design of SmE where the system level specifications are identified. These are related to the physical components of the system, their functional behavior (along with their interaction details) and the overall constraints (e.g. Security, Safety) for the designed SmE. A graphical view of the activities carried out in this step is explained in Figure 2.

The devices which are used in SmE are of two types: Controllable and Uncontrollable. Controllable devices can

Security Comm. Networks 2012; **00**:1–14 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

be divided into two main categories based on their usage: input and output devices. Input devices are used for taking the input from the environment, by observing the actions of the users (e.g. sensors) or with the direct interaction of the users to the devices (e.g. touch sensors); whereas the output devices (e.g. actuators) are used for performing the required operations, they can be self-operating or they can be attached with some other uncontrollable devices (e.g. doors, windows and gates) for controlling their functionalities. As these uncontrollable devices are used as an interface in the environment but they cannot be directly controlled through messages, for controlling them controllable devises are required to associate with them.

For the design of any SmE, it is required to identify the list of these devices with their positioning details. Also, the list of services, which are to be accomplished by using these devices, is created. Each service is associated to some devices in some relationship and against each service, SmE perform some certain functionality. At this step, it is also required to identify these relationships and the desired functionalities. Then, the overall constraints on the system are required to be identified so that they can be considered while modeling the SmE, such as the security constraints is to close all the entry points (e.g. main door, windows, rear door) when a smart home goes in "sleep mode" and the safety constraint is to open all the entry and exit points in case of fire.

After this step, a clear picture of the SmE will be obtained. Caution is advised at this stage because reliable, secure and safe implementation of the system will closely follow these specifications.

Step 2: *Users Modeling*

Users play a key role in the operations of SmE. According to their presence (observed from different sensing devices) and actions (performed on devices), SmE perform specific operations. For the identification and modeling of such requirements, a two steps process is adopted: goal modeling and behavior modeling. In goal modeling, the Goals, Actions and Roles of the users which they can achieve from SmE are described. Goals are the set of objectives which can be performed/demanded by the users. For achieving these goals, users have to perform specific actions. Roles establish a relationship between the user actions and the environment, which allows the users for performing specific task according to the environment configurations. The flow of the task carried out in this step is shown in Figure 3.

The users have complex web of behaviors which they can adopt during the interaction with the system. In behavior modeling, the analysis of their all possible moves are identified and modeled by incorporating their organized goal information. Among different perspectives, some of the behavioral aspects, which we considered for users modeling in this paper, are following:

1. How the users can interact with the system?
2. Which user actions are considerable for the system?

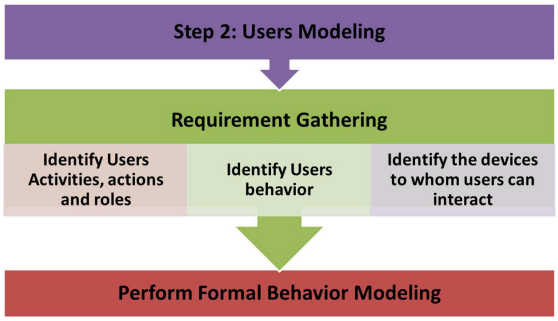

Figure 3. Step 2: Users Modeling

3. Where a user can be positioned after performing an action?
4. What are the set of possible user behaviors which they can adopt?

Some other aspects of the users, though not considered for this paper, are the following: 1) Users identification 2) Actions history of the users 3) Division of users on the basis of their roles

Stage 3: *Devices Modeling*

Controlling and commanding the functionalities of electrical (low cost or smart) devices are main goals of SmE. These devices are of heterogeneous nature with some common and distinguish features (such as functionalities, commands, notification, states and others). The desired functionality from the relevant devices is accessed by inputting some specific commands or by interacting with them depending upon the type of the devices (as mentioned in Section 3.2). For the sensor, the input is received by sensing the environment and its output is usually a notification message; whereas for other devices, the input can be a command and the output can be a physical operation. The input and output depend on the category of the devices; further the devices can be smart by having some internal constraints. These elements (input, constraints, output) are required to be gathered, organized and described at requirement gathering phase.

For the design and verification of complete interaction among SmE components, it is also required to model the attached devices at design time. The modeling of these devices can be performed by adopting interface (black box) and behavioral (white box) modeling schemes. Before modeling a device it is first required to collect their detailed relevant information, which includes the interface information – the commands (triggers) it may receive, the associated functionality (operation) it may perform, the constraints (rules) it has to follow, the states at which it will be at any time, the notifications which it sends after the completion of task – and behavioral information – the acceptance of specific commands on a particular state, the implementation of constraints, the operations which may be performed on that state after the satisfaction of constraints – of the particular devices. A graphical flow of

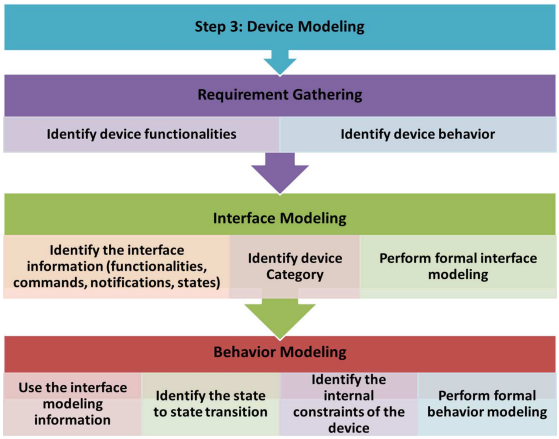

Figure 4. Step 3: Devices Modeling

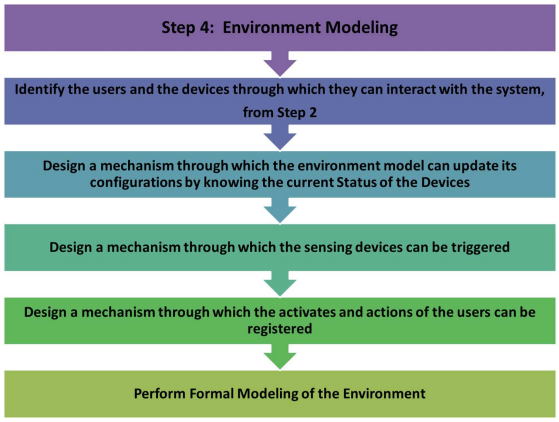

Figure 5. Step 4: Environment Modeling

the task carried out at this step is presented in Figure 4.

Step 4: *Environment Modeling*
In reality, users can observe the environment by seeing the current states of the concerning devices and accordingly interact with them for achieving the desired goals. But at design time, these features can be modeled by adding some extra computations through environment models. The environment models can update their configuration when any action or operation is performed by the concerning devices. Similarly, the environment model can be capable of registering the actions, locations and interactions of the users. At requirement gathering phase, the identification and listing of these computations, which are considered to be in the real environment, are required to be described. These descriptions help for the reliable modeling of the environment. The concerning features which are required to be considered for the environment modeling are graphically represented in Figure 5.

Considering the users' modeling at design time, it is suggested that the modeling of the environment component must also be done, as users may observe

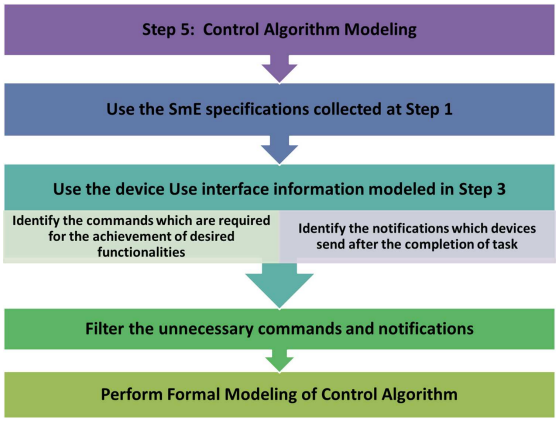

Figure 6. Step 5: Control Algorithms Modeling

the environment configurations and accordingly interact with the system. For this, a mechanism can be designed which stores the state information of interesting devices so that the users' model can observe the environment configurations at design time. As the devices model states change, the environment model updates the current state (of the particular device) with the new values. Similarly, the users' interaction with the sensors can be formalized with the use of environment modeling; the environment model can also register the activities of the user (so that the exact location of the users can be identified).

Step 5: *Control Algorithms Modeling*
Control algorithms aid the computation in the SmE. For achieving a goal, the user performs an action which is forwarded to these controlling algorithms in the form of messages. According to these incoming messages, the current configuration of the whole system and the implemented rules, control algorithms make certain sophisticated decisions and send triggering messages to the associated devices for performing the required operations.

The desired behavior of SmE (listed in Step 1: SmE Specification Identification Stage), related to providing the required services, reliable behavior, security, safety and other constraints, is achieved through control algorithms. The control algorithms accomplish the required behavior by controlling the functionalities of the concerning devices. For an effective communication, control algorithms have to use the devices interface information (which are modeled in step 3: Devices Modeling). A graphical task flow of this step is given in Figure 6.

Stage 6: *Temporal Properties Designing*
It is important for any complex and critical system to ensure the successful modeling of all the desired behavior (related to the safety, security), functionalities and other constraints is performed. For the verification of these features, the modeling of the temporal properties is required so that they can be confirmed at the formal verification step. During the formal verification, some

Security Comm. Networks 2012; **00**:1–14 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

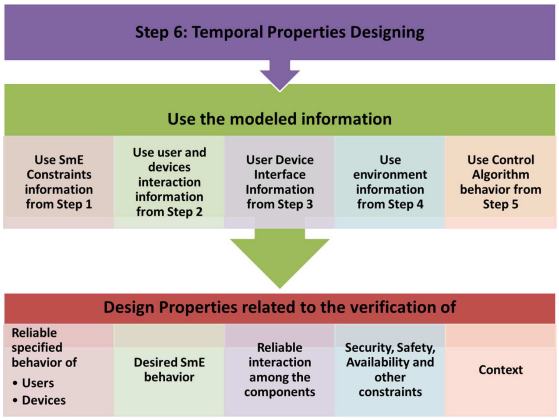

Figure 7. Step 6: Temporal Properties Designing

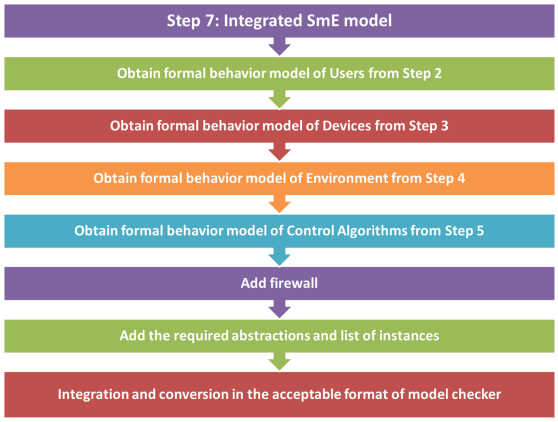

Figure 8. Step 7: Integrated SmE model

properties may likely be ignored due to system complexity. For reducing the chances of ignoring important properties, the requirements described so far are used. These requirements are further formulated by using the syntax and semantics of temporal logics.

The temporal logics are mostly used for the verification of the reachability of certain states, satisfaction of sequence, absence or existence of any predicate (at any state) and the boundary checking or the universality of any state or action. By using these features of temporal logic, the properties can be designed by which the reliable specified behavior, safety, security and other constrains of the SmE can be verified. Tasks carried out in this step are graphically represented in Figure 7.

Stage 7: *Integrated SmE model*

As control algorithms govern all the interaction among devices (and affect the environment), they receive a lot of messages (commands or notifications) from the connected devices. The devices models can send and receive nearly all possible messages related to their functionalities. But among these messages, some messages are of interest for the current system and should be modeled in control algorithms. The rest of the messages are useless for the current system, but it's a good practice that all the incoming messages must be received. If the modeling of all possible messages is performed in the control algorithms, then the size, complexity and ambiguities of control algorithms grow higher.

For curbing these issues, it is suggested to introduce a firewall around control algorithms which, at the initial level, checks the suitability of a received message and sends forward only those messages which concern the current system. Similarly, the received messages can have different parameters; therefore they can also be modified at this stage if required. This helps in optimizing the control algorithms: the processing load is reduced and the "lost-event" errors don't occur (during model checking) due to failure of acceptance at receiver's side.

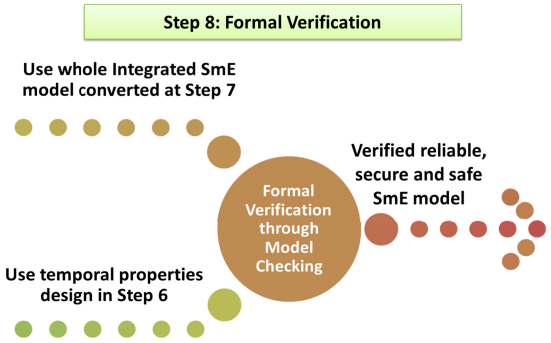

Figure 9. Step 8: Formal verification of SmE Model

Up to this stage, all the prerequisites for the modeling of SmE process are completed. Now it is required to convert them into the acceptable language of the model checker and then combine them so that a complete SmE model can be prepared. For the translation, the behavior models of the users, connected devices, environment, control algorithms, firewall (with messages filter and converter) are required, along with the proper abstraction and list of their instances (connected in the SmE). After converting them the whole integrated SmE model is designed in the acceptable format of model checker. The task carried out in this step are graphically presented in Figure 8.

Stage 8: *Formal verification of SmE Model*

The whole integrated SmE model, in the acceptable format of model checker, designed at Step 7 is sent to the model checker, and the temporal properties (designed in Step 6) are verified on the model. On finding any unsatisfactory property, the SmE model is updated with the required modifications, and the verification process is repeated until all the properties are satisfied. The task carried out in this step are graphically presented in Figure 9.

Stage 9: *Development Phase*

When all the properties are verified, it is implied that

the SmE model is according to the specification and will behave reliably, surely and safely under the verified properties in all scenarios. It is time to safely start the development and implementation phases.

5. DESIGNED TECHNIQUE

The proposed methodology is implemented by enhancing our existing technique [16]) by additionally focusing on users and environment (context) modeling. The technique employs DogOnt, SCXML, UCTL and UMC as tools and follows the steps of proposed methodology, as graphically represented in Figure 10. The designed technique works by adopting the following activities:

1. SmE and its related components requirements are organized according to the operational flow (as mentioned in Section 3.2 and various Steps of the proposed methodology);
2. the interface modeling of each connected devices with their corresponding states, functionalities, commands, notifications and other related information is represented in DogOnt Ontology;
3. the behavior Statechart modeling of users, devices, control algorithms and the environment is represented in SCXML semantics;
4. the computation requirements in the form of properties are formalized by adopting the following steps:
   (a) according to the modeled requirements, the possible computational properties are identified;
   (b) for designing the properties, the system configurations (such as the information of all the associated instances of devices with their location, states, functionalities, commands, notifications and others) are queried with the use of DogOnt;
   (c) the Statecharts modeling of the corresponding components are used for querying the sequences of commands, notifications and states[†];
   (d) properties are designed based on above mentioned information, by using the syntax and semantics of temporal logic acceptable by model checker (UCTL in our case);
5. the firewall component (for filtering and converting the messages) is represented in SCXML semantics;

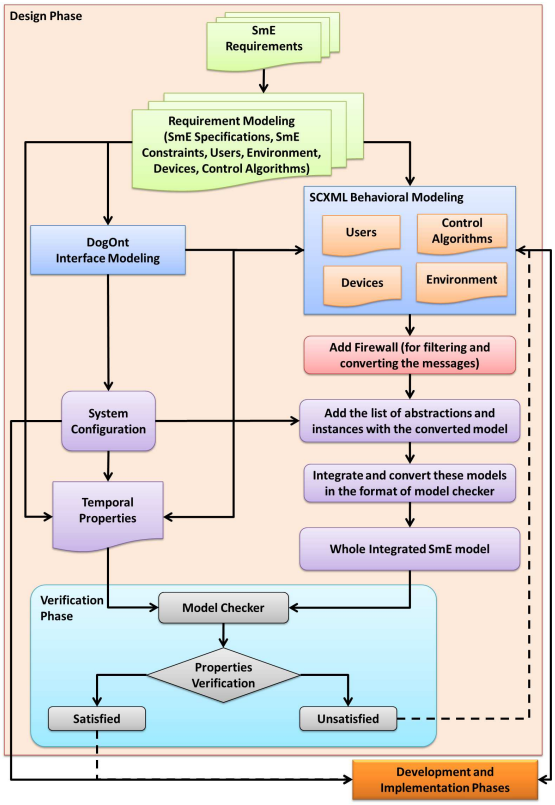

Figure 10. Designed Technique

6. the behavioral models of the SmE components and firewall are converted in the acceptable format of model checker;
7. add required abstractions and the device instances information queried from DogOnt;
8. the designed properties and the complete SmE model are passed to the model checker (UMC in our case), which verifies these properties on the model and reports about their satisfaction:
   (a) in the case of finding unsatisfactory properties, the corresponding behavioral models are updated with the required modifications;
   (b) the verification process is repeated until all the properties are satisfied

When all the properties are satisfied, then the system can be declared as reliable, safe and secure, and will behave well according to the verified requirements in all scenarios. As a result, the implementation phase can be started.

[†]Note: The behavioral modeling of the devices, in the form of SCXML, are consistent with the modeling of DogOnt as their reliable design and consistency is already verified in our previous work [10].

Security Comm. Networks 2012; **00**:1–14 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

6. SME IN REAL WORLD- A BDSB SYSTEM

The Bank Door Security Booth System (BDSB) is our real world example of a SmE system [16], which is extended with the concept of users' and environment (context) modeling. Although BDSB is an initial level small SmE system, it exhibits a complex behavior due to the interaction of multiple users with the system and performs a complex communication between different hardware (e.g. devices) and software (e.g. control algorithms) components according to user interaction. A graphical layout of the BDSB environment is presented in Figure 11.

The BDSB is designed in such a way that multiple users can interact with the system; ideally, the security and safety measures of the BDSB system should never fail. The system is composed of two electronically controlled doors, located outside (known as external door (DExt)) and inside the bank (known as inner door (DInner)). For electronically controlling a door, actuators are installed. DExt and DInner are controlled by DAExt and DAInner door actuators respectively.

There is an isolated space between both doors, where users have to wait so that the opened door is closed first and then the other door may be opened. The user request for door opening is only possible through touch sensors (TS), which are installed near each side of each door. The Touch sensor attached outside the DExt is called T1, and the one attached within the isolated space is called T2. Similarly, the touch sensor attached to the DInner from within the isolated side is called T3 and the one attached from inside of the bank is called T4.

The Door sensors (DSExt and DSInner) are used for querying the status (whether it is open, close or in moving states) of the door; DSExt is attached with DExt and DSInner is attached with DInner. Similarly, two obstacle detection sensors are used for reopening the door when it is in closing state and any object (e.g. person) is held in between the closing path of the door, ODSExt is attached with DExt and OSDInner is attached with DInner. A control algorithm, known as Door Lock Control (DLC), manages all the communication and functionalities of these devices in a safe and secure way.

The design details, by following the proposed methodology and the imposed constraints on BDSB, is given below:

Step 1:*SmE Specification Identification*
The design specifications, the internal constraints and desired behavior of the BDSB systems are given bellow:

Design Specification

1. two doors (external and internal) are used for ensuring the security measures from the harmful access (direct access should not be possible) to the bank;
2. there is an isolated space between external and internal doors;

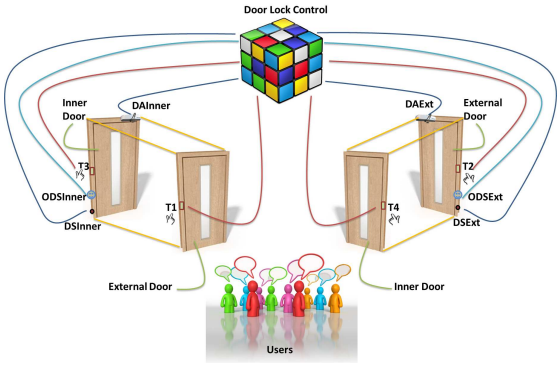

Figure 11. Bank Door Security Booth System

3. doors can be controlled from the outside and inside of the isolated space through the associated touch sensors installed at each side of the door (by sending the door-open request), so that the people can cross the door without being stuck;

Internal constraints and desired behavior

1. doors will remain open for a fixed time after opening and before closing so that the users can cross;
2. when one door is in the process of opening-and-closing and the same door-open request from the associated TS arrives, BDSB checks the state at which the request is received and accordingly performs the following action:
   (a) if the same door-open request arrives when the door is in the opening process, BDSB just holds this request and will not open the door again;
   (b) if the same door-open request arrives when the door is in the closing process, BDSB will re-open the door;
3. if one door is in the opening-and-closing process and the door-open request from the other door arrives, the BDSB will hold the request and wait for the closing of other door. As soon the other door will be closed, BDSB will open the requested door.

Step 2: *Users Modeling*
The following is the list of users' activities, behaviors and observations which are considered for the users' modeling in BDSB system:

1. user can access and return from the bank by crossing the doors;
2. users can press the associated touch sensors (at each side of the door) for opening the doors;
3. users can press touch sensors more than one time;

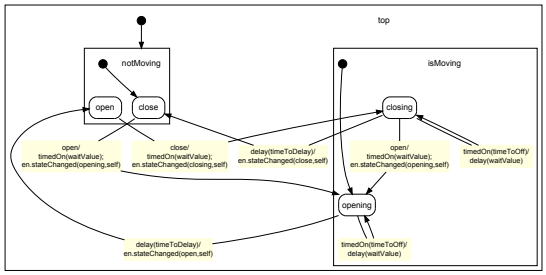

Figure 12. Statechart modeling of Door Actuator (obtained by using UMC model checker)

4. users observe the states of the doors and when a door is found open, they can act in following ways:
   (a) they may cross the door;
   (b) their mind may change and they stay there without crossing the door;
   (c) they cross the door, but sooner their mind may change and they cross-back and come to their previous location.
5. users can change their mind from the isolated space and exit from there without entering into the bank; similarly they can re-enter in the bank without exiting.

Step 3: *Devices Modeling*
Touch sensors, door actuators, door sensors and obstacle detection sensors are used as controllable devices in BDSB. The modeling of each device is performed according to the activities specified in the methodology (from requirement gathering to their behavioral modeling); such as the door actuator component of BDSB system is described to have open-close functionality by which it provides force to open or close the door. The door actuator, at any specific time, can be in moving (opening-closing) or in non-moving (open-close) state. For activating the desired functionality, it accepts open or close command and, accordingly, performs its operation. It can also send the notification back after the state has changed. The behavioral modeling of door actuator, in Statechart format, is represented in Figure 12.

Step 4: *Environment Modeling*
The users can view the states of the door, whether it is in open, close, opening or closing state; and accordingly perform some actions (e.g. cross the door, press the corresponding touch sensor). For designing such a real environment, an environment model is designed by having the ability to update it's configuration as soon as the doors change their states (taking advantage from State-Change-Notification message). Through this the users can have the latest configuration of the environment and can behave accordingly. Similarly, for knowing the proper location of users and accordingly providing access to the relevant devices, environment model registers the actions of the users. Additionally, the interaction with the obstacle detecting sensors can also be made through the environment modeling. All these features are modeled with the help of parallel Statechart formalisms.

Step 5: *Control Algorithms Modeling*
Door Lock Control (DLC) is an intelligent component of a BDSB system. It takes inputs from Touch Sensors (TS), Door Sensors (DS) and obstacle detection sensors (ODS), and according to the designed requirements, instructs the Door Actuators for opening/closing the doors. All the computation requirements (mentioned in SmE Specification Identification Stage) are achieved through DLC. For achieving the desired computational requirements (what to do when the door-open request arrives? when the requested door will be opened? when to send the acknowledgment?), different guards (constraints) are designed with the use of relational and logical operators. These guards work on the basis of incoming messages and the variable values.

Step 6: *Temporal Properties*
As mentioned, the requirements related to the reliable behavior of BDSB along with the safety, security and other constraints are formalized by using the syntax and semantics of UCTL temporal logic. The detailed description of these requirements with designed properties are given in Section 7: one of the requirements of BDSB system (in UCTL format) is that the external door will be opened when the user releases any touch sensor associated at each side of the external door. The touch sensor associated with the external door from the isolated space can only be accessed when user crosses the external door; therefore the first part of the property ensures that one user has crossed the door, now the door-open request of both sensor can arrive. The next path of the property is related to that scenario that the extDoorOpened request will not arrive until the associated touch sensors are pressed.

$$EF_{\{extDoorCrossed\}} A\left[\top\ {}_{\{\neg extDoorOpened\}}U_{\{T1ReleaseorT2Release\}}\ \top\right]$$

Stage 7: *Integrated SmE model*
The firewall is added so that the all the messages can be received and only the useful messages and notifications can be passed. Then, along with the firewall component, the individual behavior model of users, devices, environment and control algorithm are converted into the acceptable format of UMC. Further, the abstractions and instances information is added for completing the holistic integrated BDSB model.

Step 8: *Formal Verification*
All the temporal properties, included the one mentioned above, are verified on the BDSB model and the satisfactory results are obtained, confirming the successful exhaustive verification of our tested SmE, with the explicit focus on

Security Comm. Networks 2012; **00**:1–14 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

safety and security requirements. The results are discussed in the following section.

7. EXPERIMENT AND RESULTS

In this Section, the requirements related to the safety, security and reliable behavior of BDSB and its related components are formalized according to the categories (users' behavior modeling, users and their interaction modeling with the devices, device modeling, devices interaction and their control modeling and context modeling) defined in Section 2, by considering the message exchange behavior of BDSB and its components. These properties are then specified in UCTL format. All the properties are individually verified on BDSB model. The abstracted evolution graph (generated by UMC) of BDSB model consists of 2,79,119 states with the depth at 30 levels. The time taken for verifying each property was usually less than a minute in the on-line version of UMC. In Table II, the reference of these properties is given with their evaluation time, the number of states and computations fragments generated for evaluating them. During the verification process at first stage, it was found that the designed model did not satisfy all the properties. UMC provides an error trace tree through which the errors have been located and the model was updated by fixing the bugs. The verification process has been repeated until all the properties were proven TRUE against the BDSB model.

Properties related to Users behavior

The user modeling is performed according to the specifications; all the users can enter the bank by crossing the external door, the isolated space and the internal door. It is also possible that users may change their mind and stay out without crossing the external door. Therefore, path 'Existence' quantifier is used in the property instead of 'All' quantifier for the verification. Similarly, users' mind may change and they may go back from the isolated space without crossing the internal door. For verifying that users can access the places, the following set of state properties (by using state abstraction) is formalized.

P1) $EF_{(u1AtOutsideTheBank)}$

P2) $EF_{(u1AtIssolatedSpace)}$

P3) $EF_{(u1AtInnerSideOfTheBank)}$

P4) $EF_{\{extDoorCrossed\}}\top$

P5) $EF_{\{extDoorCrossed\}}\top$

Properties related to actions performed by the Users

For achieving any goal, users have to perform some action. To know that users can press and release the respective touch sensors, the following set of properties is formalized. Although all the users can access the outside touch sensor of external door, the other sensors (T2 and T3) can only be accessed when the user has crossed the external door, whereas T4 can only be accessed when user has also crossed the inner door. Therefore, 'Existence' quantifier is used with the properties of other touch sensors.

P6) $AF_{\{T1Release\}}\top$

P7) $EF_{\{T2Release\}}\top$

P8) $EF_{\{T3Release\}}\top$

P9) $EF_{\{T4Release\}}\top$

Properties related to Users and Device Interaction

The external door will be opened when the user releases any touch sensor associated at each side of the door. Same will happen with the inner door. The following set of properties is used to verify such type of users' interaction with the devices.

P10) $A\left[\top\ _{\{\neg extDoorOpened\}}U_{\{T1Release\}}\ \top\right]$

P11) $EF_{\{extDoorCrossed\}}$
$E\left[\top\ _{\{\neg extDoorOpened\}}U_{\{T2Release\}}\ \top\right]$

P12) $EF_{\{extDoorCrossed\}}$
$A[\top\ _{\{\neg extDoorOpened\}}$
$U_{\{T1ReleaseorT2Release\}}\ \top]$

P13) $EF_{\{innerDoorCrossed\}}$
$E\left[\top\ _{\{\neg innerDoorOpened\}}U_{\{T3Release\}}\ \top\right]$

P14) $EF_{\{innerDoorCrossed\}}$
$E\left[\top\ _{\{\neg innerDoorOpened\}}U_{\{T4Release\}}\ \top\right]$

P15) $EF_{\{innerDoorCrossed\}}$
$A[\top\ _{\{\neg innerDoorOpened\}}$
$U_{\{T3ReleaseorT4Release\}}\ \top]$

Properties related to Safety Constraints

One of the safety constraints is to ensure no user is stuck inside the isolated space. In any case, the user may exit the space by either entering inside the bank of exiting out. The following set of properties is used to verify this type of safety constraints.

P16) $AF_{\{T1Release\}}AF_{\{DoorResponse(open,DAExt)\}}\top$

P17) $EF_{\{T2Release\}}AF_{\{DoorResponse(open,DAExt)\}}\top$

P18) $EF_{\{T3Release\}}AF_{\{DoorResponse(open,DAInner)\}}\top$

P19) $EF_{\{T4Release\}}AF_{\{DoorResponse(open,DAInner)\}}\top$

Table II. The properties with their evaluation details

Property	Evaluation Time *(in Sec.)*	States Generated	Computations Fragments Generated
P1	$<$ 1 ms	2	2
P2	$<$ 1 ms	63	110
P3	0.33	3778	7461
P4	$<$ 1 ms	62	52
P5	0.48	3791	3826
P6	$<$ 1 ms	2	2
P7	0.03	389	593
P8	0.03	388	384
P9	0.50	3940	4310
P10	$<$ 1 ms	2	2
P11	0.02	286	310
P12	0.51	4310	5511
P13	6.91	77397	81725
P14	0.50	3936	3993
P15	0.74	6252	6782
P16	$<$ 1 ms	37	63
P17	0.26	2770	3039
P18	0.26	2770	2976
P19	6.76	77083	80043
P20	0.08	821	1005
P21	0.83	6935	7966
P22	0.07	819	819
P23	0.14	1388	2169
P24	0.52	4117	4281
P25	0.74	6252	6782

Properties related to individual Devices

When the command for opening the door is passed to any door actuator, it will open the respective door as a result. These properties are used to verify the functionalities of the door actuators that, when they receive the open command, after opening the door, they will also close it.

P20) $AF_{\{OpenExtDoor\}}$
$AF_{\{DoorResponse(close,DAExt)\}}\top$

P21) $EF_{\{OpenInnerDoor\}}$
$AF_{\{DoorResponse(close,DAInner)\}}\top$

Properties related to Security Constraints

Ideally, both of the doors should not be opened at a same time, the open door must be closed first and then the other requested door will be opened.

P22) $A[\top\ _{\{\neg DoorResponse(open,DAInner)\}}$
$U_{\{DoorResponse(close,DAExt)\}}\ \top]$

P23) $EF_{\{extDoorCrossed\}}$
$A[\top\ _{\{\neg DoorResponse(open,DAInner)\}}$
$U_{\{DoorResponse(close,DAExt)\}}\ \top]$

P24) $EF_{\{innerDoorCrossed\}}$
$A[\top\ _{\{\neg DoorResponse(open,DAExt)\}}$
$U_{\{DoorResponse(close,DAInner)\}}\ \top]$

Properties related to Context Awareness

The users can access the touch sensors only when they are at a proper location. When users are inside the bank, they can come out from the bank by pressing the touch sensor attached at the inner side of the bank.

P25) $EF_{\{innerDoorCrossed\}}$
$A[\top\ _{\{\neg innerDoorCrossed\}}U_{\{T4Release\}}\ \top]$

8. DISCUSSION

The Table II shows the temporal values of verification of various tested properties. The average time for verifying all the 25 properties is 0.79 sec., with the standard deviation 1.83. As a general rule, the superficial properties (for which the on-the-fly model checker didn't have to go deeper inside the system for verification and a smaller number of states are generated) are verified in relatively lesser time, such as P1, P2, P4, P6, P10 and P16 (takes less than 1 millisecond ($<$ 1 ms)). Whereas the complex properties (for which the on-the-fly model checker had to go deeper inside the system for verification and a larger number of states are generated) are verified using more time, such as P13 and P19.

Security Comm. Networks 2012; **00**:1–14
DOI: 10.1002/sec
Prepared using secauth.cls

9. CONCLUSION AND FUTURE WORK

The proposed design time verification methodology, aided by user behavior modeling, device modeling, environment/context modeling, control algorithm modeling, and their interaction modeling, has demonstrated successful results for verifying the correctness, reliability, safety, security and desired behavior of SmE systems. The methodology proceeds sequentially from requirement listing to modeling and formal verification. The probability of missing any properties has been efficiently controlled by requirement listing. The methodology is implemented through the designed technique and implemented on a small but not so simple real life SmE system. The first run of verification process did not achieve all the properties as satisfactory against the model. After appropriate modifications to the model, it was then proven to conform to design requirements. This verified model can be used safely at the implementation phase. In our future work, we envisage to achieve the implementation of proposed methodology at a grand scale, exposing it to the challenges of large scale execution. Also, the other aspects of user modeling, not included in this paper, may be incorporated for future works.

REFERENCES

1. Weiser, M.. The computer for the 21st century. *Scientific American* 1991; **265**(3):94–104.
2. Saha, D. and Mukherjee, A.. Pervasive computing: a paradigm for the 21st century. *Computer* 2003; **36**(3):25–31.
3. Olson, G.M. and Olson, J.S.. Human-computer interaction: Psychological aspects of the human use of computing. *Annual Review of Psychology* 2003; **54**(1):491–516.
4. Sadri, F.. Ambient intelligence: A survey. *ACM Comput. Surv.* Oct 2011; **43**(4):36:1–36:66.
5. Augusto, J.C. and McCullagh, P.. Ambient Intelligence: Concepts and Applications. *Computer Science and Information Systems* 2007; **4 (1)**:1–27.
6. Al-Muhtadi, J. and Ranganathan, A. and Campbell, R. and Mickunas, M.D.. Cerberus: a context-aware security scheme for smart spaces. *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*, IEEE, 2003; 489–496.
7. Youngblood, M. and Cook, D.J. and Holder, L.B.. Seamlessly engineering a smart environment. *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, IEEE, 2005; 548–553.
8. Bonino, D. and Castellina, E. and Corno, F.. The DOG gateway: enabling ontology-based intelligent domotic environments. *Consumer Electronics, IEEE Transactions on* november 2008; **54**(4):1656 –1664, doi:10.1109/TCE.2008.4711217.
9. Bourcier, J. and Chazalet, A. and Desertot, M. and Escoffier, C. and Marin, C.. A dynamic-soa home control gateway. *Services Computing, 2006. SCC'06. IEEE International Conference on*, IEEE, 2006; 463–470.
10. Corno, F. and Sanaullah, M.. Formal Verification of Device State Chart Models. *Intelligent Environments (IE), 2011 7th International Conference on*, IEEE, 2011; 66–73.
11. Coronato, A. and De Pietro, G.. Formal design of ambient intelligence applications. *Computer* 2010; **43**(12):60–68.
12. Bolton, M.L. and Bass, E.J.. A method for the formal verification of human-interactive systems. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 2009; **53**(12):764–768.
13. Bernardeschi, C. and Fantechi, A. and Gnesi, S. and Larosa, S. and Mongardi, G. and Romano, D.. A formal verification environment for railway signaling system design. *Formal Methods in System Design* 1998; **12**(2):139–161.
14. Clarke, E.M. and Wing, J.M.. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)* 1996; **28**(4):626–643.
15. Gupta, A.. Formal hardware verification methods: A survey. *Formal Methods in System Design* 1992; **1**(2):151–238.
16. Corno, F. and Sanaullah, M.. Design time Methodology for the Formal Verification of Intelligent Domotic Environments. *Ambient Intelligence-Software and Applications* 2011; :9–16.
17. Bonino, D. and Corno, F.. DogOnt - Ontology Modeling for Intelligent Domotic Environments. *The Semantic Web-ISWC 2008* 2008; :790–803.
18. Harel, D.. Statecharts: a visual formalism for complex systems. *Science of Computer Programming* 1987; **8**(3):231 – 274.
19. Jim Barnett, G. et al.. State Chart XML (SCXML): State Machine Notation for Control Abstraction. *Technical Report*, W3C May 2010. URL `http://www.w3.org/TR/scxml/`.
20. Mazzanti, F.. *UMC 3.3 User Guide, ISTI Technical Report 2006-TR-33*. ISTI-CNR Pisa-Italy September 2006.
21. Gnesi, S. and Mazzanti, F.. On the fly model checking of communicating UML State Machines. *Second ACIS International Conference on Software Engineering Research, Management and Applications*, 2004; 331–338.
22. Bonhomme, S. and Campo, E. and Esteve, D. and Guennec, J.. Methodology and tools for the design and verification of a smart management system for home comfort. *Intelligent Systems, 2008. IS '08. 4th International IEEE Conference*, IEEE, 2008; 24–2 –24–7.
23. Coronato, A. and Pietro, G.D.E.. Formal Specification of Wireless and Pervasive Healthcare Applications.

ACM Transactions on Embedded Computing Systems 2010; **10**(1):12.
24. Gnesi, S. and Latella, D. and Massink, M.. Model checking UML statechart diagrams using JACK. *High-Assurance Systems Engineering, 1999. Proceedings. 4th IEEE International Symposium on*, IEEE, 1999; 46–55.
25. Leelaprute, P. and Nakamura, M. and Tsuchiya, T. and Matsumoto, K. and Kikuno, T.. Describing and verifying integrated services of home network systems. *Software Engineering Conference, 2005. APSEC '05. 12th Asia-Pacific*, 2005; 10 pp.
26. Meshkova, E. and Riihijarvi, J. and Mahonen, P. and Kavadias, C.. Modeling the home environment using ontology with applications in software configuration management. *Telecommunications, 2008. ICT 2008. International Conference on*, IEEE, 2008; 1–6.
27. Gu, T. and Wang, X.H. and Pung, H.K. and Zhang, D.Q.. An ontology-based context model in intelligent environments. *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, Citeseer, 2004; 270–275.
28. Xu, J. and Lee, Y.H. and Tsai, W.T. and Li, W. and Son, Y.S. and Park, J.H. and Moon, K.D.. Ontology-Based Smart Home Solution and Service Composition. *2009 International Conference on Embedded Software and Systems*, IEEE, 2009; 297–304.
29. Fensel, D.. *Ontologies: a silver bullet for knowledge management and electronic commerce*. Springer-Verlag: New York, NY, USA, 2001.
30. Preuveneers, D. and Van den Bergh, J. and Wagelaar, D. and Georges, A. and Rigole, P. and Clerckx, T. and Berbers, Y. and Coninx, K. and Jonckers, V. and De Bosschere, K.. Towards an extensible context ontology for ambient intelligence. *Ambient Intelligence* 2004; :148–159.
31. Nielsen, M. and Plotkin, G. and Winskel, G.. Petri nets, event structures and domains, part i. *Theoretical Computer Science* 1981; **13**(1):85–108.
32. Hoare, C.A.R.. Communicating sequential processes. *Communications of the ACM* 1978; **21**(8):666–677.
33. Bogdanov, K. and Holcombe, M.. Statechart testing method for aircraft control systems. *Software testing, verification and reliability* 2001; **11**(1):39–54.
34. Lee, D. and Yannakakis, M.. Principles and methods of testing finite state machines. *PROCEEDINGS-IEEE* 1996; **84**:1090–1123.
35. Holzmann, G.J.. *The SPIN model checker: Primer and reference manual*. Addison Wesley Publishing Company, 2004.
36. Cimatti, A. and Clarke, E. and Giunchiglia, F. and Roveri, M.. NuSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer (STTT)* 2000; **2**(4):410–425.
37. Kwiatkowska, M.Z. and Norman, G. and Parker, D.. PRISM: Probabilistic symbolic model checker. *Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, Springer-Verlag, 2002; 200–204.
38. Ter Beek, M.H. and Mazzanti, F. and Gnesi, S.. CMC-UMC: A Framework for the Verification of Abstract Service-Oriented Properties. *Proceedings of the 2009 ACM symposium on Applied Computing*, ACM, 2009; 2111–2117.
39. De Nicola, R. and Vaandrager, F.. Action Versus State based Logics for Transition Systems. *Semantics of Systems of Concurrent Processes, Lecture Notes in Computer Science* 1990; **469**:407–419.
40. Clarke, E.M. and Emerson, E.A. and Sistla, A.P.. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems* April 1986; **8:2**:244–263.
41. Hennessy, M. and Milner, R.. On observing nondeterminism and concurrency. *Automata, Languages and Programming* 1980; :299–309.
42. Manna, Z. and Pnueli, A.. *The temporal logic of reactive and concurrent systems: Specification*. Springer-Verlag New York, Inc.: New York, NY, USA, 1992.
43. Booch, G. and Rumbaugh, J. and Jacobson, I.. *Unified Modeling Language User Guide, The*. Addison Wesley. ISBN 0-201-57168-4, 1998.

DOI: 10.1002/sec
Prepared using secauth.cls