Design-time formal verification for smart environments: an exploratory perspective

Journal of Ambient Intelligence and Humanized Computing manuscript No.
(will be inserted by the editor)

Design-Time Formal Verification for Smart Environments: An Exploratory Perspective

Fulvio Corno · Muhammad Sanaullah

Received: date / Accepted: date

Abstract Smart Environments (SmE) are richly integrated with multiple heterogeneous devices; they perform the operations in intelligent manner by considering the context and actions/behaviors of the users. Their major objective is to enable the environment to provide ease and comfort to the users. The reliance on these systems demands consistent behavior. The versatility of devices, user behavior and intricacy of communication complicate the modeling and verification of SmE's reliable behavior. Of the many available modeling and verification techniques, formal methods appear to be the most promising.

Due to a large variety of implementation scenarios and support for conditional behavior/processing, the concept of SmE is applicable to diverse areas which calls for focused research. As a result, a number of modeling and verification techniques have been made available for designers. This paper explores and puts into perspective the modeling and verification techniques based on an extended literature survey. These techniques mainly focus on some specific aspects, with a few overlapping scenarios (such as user interaction, devices interaction and control, context awareness, etc.), which were of the interest to the researchers based on their specialized competencies. The techniques are categorized on the basis of various factors and formalisms considered for the modeling and verification and later analyzed. The results show that no surveyed technique maintains a holistic perspective; each technique is used for the modeling and verification of specific SmE aspects. The results further help the designers select appropriate modeling and verification techniques under given requirements and stress for more R&D effort into SmE modeling and verification research.

Keywords Smart Environments · Formal Modeling · Formal Verification · Design-Time Verification

Fulvio Corno
Politecnico di Torino, Dipartimento di Automatica ed Informatica, Corso Duca degli Abruzzi 24, 10129 - Torino, Italy
E-mail: fulvio.corno@polito.it

Muhammad Sanaullah ✉
Politecnico di Torino, Dipartimento di Automatica ed Informatica, Corso Duca degli Abruzzi 24, 10129 - Torino, Italy
E-mail: muhammad.sanaullah@polito.it

1 Introduction

Over the last few years, the concept of enriching the physical world with sensation, making decisions and taking actions corresponding to the user desire has become an open and challenging research area, particularly in computer science. A number of researchers are working for designing such environments which are "richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network" (Weiser, 1991). The availability of low cost devices, advancement in artificial intelligence, ubiquitous and pervasive computing, wired and wireless networking, databases and other relevant technologies have enabled to achieve such environments.

The environments, based on adopted technologies and their application scenarios, are mostly referred in literature as Smart Environments (SmE), Smart Spaces, Smart Homes, Ambient Intelligence (AmI), Intelligent Environments (IEs) and Intelligent Domotic Environments (IDE). The goal of SmE is to facilitate the environment by interweaving these technologies for benefiting the user with ease and comfort along with the safety and security (Acampora and Loia, 2005; Chen

and Helal, 2012; Cook, 2009; Diane and Sajal, 2004). Due to these benefits, SmE have penetrated into homes, hospitals, offices, industries, airports, railways, transportation mediums and many other important places (Sadri, 2011).

In SmE, the computation is added in the environment in such a manner that the user remains unaware of its existence, but the system considers his presence and actions, and performs the activities accordingly through electrical devices in controlled and intelligent manner. The computational elements can be referred as control algorithms. The communication among control algorithms and devices can be performed with the exchange of messages. The intricate communication between several heterogeneous devices, computational elements and the user actions sensation, along with implementing a number of constraints regarding safety, security and reliable behavior of the system in a sophisticated manner, make SmE complex. Moreover, the working capabilities and internal behavior of each component (devices, control algorithms and user's considerable actions) are independent from those of other components of the system. The independent and interactive nature of these components, along with satisfying system constraints, adds into the complexity of SmE system. As a result, errors may occur frequently, and lead towards critical/unwanted situations. To cope with such situations, the reliable behavior of the system can be ensured by exploiting verification approaches, which help in identifying and correcting the errors in early design stages of the system (Clarke and Wing, 1996).

The verification process can be performed at design or implementation time; based on complexities and application scenarios (like theft or traffic control systems, nursing care houses and others, where the errors can be the cause of criticality), it is advisable to verify SmE models at design time for reducing criticality, time, cost and energy, and achieving the reliability (Bernardeschi et al, 1998; Clarke and Wing, 1996; Coronato and Pietro, 2010b; Wang, 2004). For design time verification, simulation or formal (mathematical) methods (strategies and structured approaches) are commonly used with strengths and limitations of the adopted methods. It may be very difficult to verify the accuracy of SmE on all possibly reachable paths through simulations, because of their complex behavior. Thus, formal methods are preferred due to their implicit coverage of all possible paths (Woodcock et al, 2009). Moreover, formal methods are mathematically oriented, which specify and verify a model of the system in accordance with the desired behavior using several techniques and tools. The adoption of formal methods resolves ambiguities, inconsistencies and incompleteness in the designed model (Clarke and Wing, 1996).

The formal verification of all possible aspects of SmE is arduous and laborious undertaking owing to the complex nature of these systems. Therefore, research seems to have emphasized the specific aspects based on requirements and interests. As a result, this focused approach has deprived the academicians and new researchers/designers from a generic and one-size-fits-all kind of modeling and verification technique. In an attempt to collect the existing state-of-the-art, this paper brings together the techniques/approaches that are exploited in formal verification of SmE with respect to different aspects. Moreover, the paper proposes a parameter-based empirical methodology of several approaches adopted in verification of SmE at design-time. The proposed empirical methodology helps to understand the verity of adopted modeling and verification techniques in different applications and scenarios. The study expands upon the uncovered modeling and verification areas of SmE. The findings of the research may help other researchers and designers to deeply understand the existing techniques, and their adoption in different scenarios.

The rest of the paper is organized as following: Section 2 defines a generic framework of SmE; the existing formal modeling and verification process adopted for SmE are reported in Section 3; the surveyed literature is presented in section 4; the proposed parameter-based empirical methodology is described in section 5 with the overview of existing state-of-the art; and finally the analysis and concluding remarks, on the surveyed literature against the proposed methodology, are presented in section 6 and 7, respectively.

2 A Framework of Smart Environments

The basic components of SmE are users, devices, control algorithms and context/environment (Weiser, 1991; Sadri, 2011; Augusto and Hornos, 2013). The execution flow of the instructions starts from the user, as shown in Figure 1, who wishes to achieve certain goals/desires for which it is important to perform some specific actions. These actions are sensed by sensors, directly performed on the devices, or instructed through hand-held computing devices (by programming and using APIs). These actions are sent to the control algorithms in the form of messages. The control algorithms are responsible for controlling and satisfying the system level specific constraints (e.g. safety, security and reliable desired behavior), and may reside at the gateway level where all the devices are connected through some wired or wireless medium. On receiving a message, the control

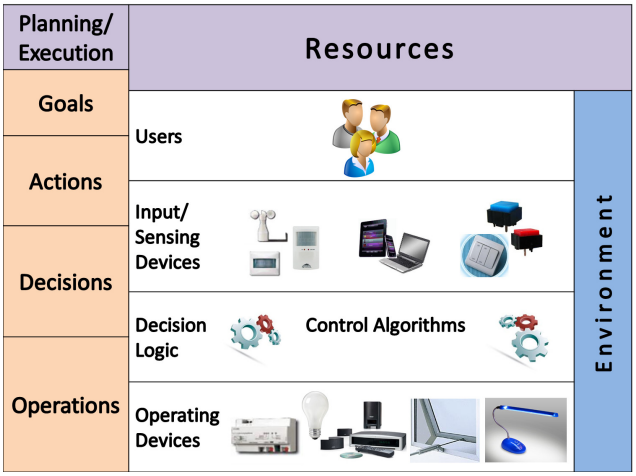

Fig. 1 A Generic Framework of Smart Environments

algorithms decide for the preferable operations by using artificial intelligence, rule based or some other relevant technologies and accordingly send commands to the relevant devices for performing the specific functionalities.

For example, consider a smart home where whenever a user enters in the bedroom, it will be illuminated depending upon the outside light intensity (user goal). The smart room senses through sensors the presence/entrance of the user (action). The sensor will send the notification message to the control algorithm. The control algorithm sends a request to the illumination sensor (placed outside the room) that replies with the outside light intensity value. According to this value, the current configuration of window-shutter and the lamp, a control algorithm decides how to illuminate the room (decision): either by moving the window-shutter up or by switching the lamp on. Based on the optimal decision, the control algorithm sends suitable commands to the corresponding devices, which perform the task (operation).

3 Modeling and verification processes

This section describes commonly adopted modeling and verification processes during formal verification. These processes are classified according to their coverage of SmE aspects and application domain, namely: 1) formal modeling, 2) component modeling, 3) formal verification, 4) Adopted Procedures/Tools. The details of each process are described in the following subsections.

3.1 Formal Modeling

Formal Modeling is the process of describing a system (a set of interconnected components performing desired operations) in a well defined formal syntax and semantics language; the following are its different perspectives adopted in the modeling of SmE.

3.1.1 Black Box Modeling

Black Box or Interface modeling is the representation of the information required to interact with the system. The black box modeling focuses on functionalities of the system without any internal details. For instance in the running example, the control algorithm sends a command to the window shutter to move up; the command is fulfilled by the window shutter-actuator. The details about how the command has been sent by control algorithm and how the operation is performed by the shutter-actuator are not considered in the black box models. Instead, the modeling of which message is sent and which action is performed against it are the main focus of the black box.

3.1.2 White Box Modeling

White Box or Behavioral modeling is a representation of a complete internal behavior of the system. The details of how commands are issued, how operations are carried out, and how the system (or individual component) requirements are fulfilled are taken care of in white box modeling. In the running example, for instance, details about how the control algorithm sends commands, how other devices perform their tasks: in other words, complete flow of actions done by the system (or components) is modeled in this category.

3.1.3 Intelligence Modeling

One of the basic objectives of SmE is to provide services in an intelligent way according to the system-level specifications and constraints. To enrich SmE with intelligence, different artificial intelligence and database techniques can be adopted, and their representation is called intelligent modeling. For example, the decision logic of control algorithms either to move window shutter up or to switch lamp off can be modeled by adopting different techniques, such as fuzzy logic, decision trees, rules based, event-condition-action, etc.

3.1.4 Requirements Modeling

Requirements are the starting point of any formal verification process and are specified in the form of properties or axioms by adopting the syntax of some formal language. These properties or axioms are designed to represent behavioral and non-behavioral aspects of

the system. The behavioral aspects are related to reliable functionalities (relation between event and action) and non-behavioral aspects are related to security and safety policies, performance, and other characteristics of the system. For instance, in the running example, the requirements related to the events – when the outside light intensity is high then the smart home has to move the window shutter up as well as switch off the lamp (if it is found on) – are presented in formal way in this modeling approach. Formal representation of the requirements is often expressed in temporal logic.

3.2 Component Modeling

The components of SmE are users, context/environment, devices and control algorithms. Depending upon the application domain, covered features and the interested scenarios, the (black-box or/and) white-box modeling of these components are accordingly performed. Modeling of these components along with their interaction details are considered in this classification.

3.3 Formal Verification

The system correctness with respect to its specifications and constraints can be formally (comprehensively) verified and this process is known as formal verification. During the verification process, different aspects of the system are verified. The description of the noted aspect is presented in the following sub-sections.

3.3.1 Consistency Verification

The consistency verification provides coherency of modeling when both black box and white box processes are applied. It is important to verify that both of the formalisms are consistent with each other; otherwise there is a fair chance that one of the formalisms may have some additional or missing information. Due to inconsistencies, each formalism may behave differently and the access to desired functionality in an independent way may be difficult. For example, if the command to move window shutter up in black box is recognized as "UP", whilst the same command in white box is identified as "RISE", this causes inconsistency between the two modeling processes and will lead towards denial of the desired outcomes (Corno and Sanaullah, 2011b). Similarly, it is important to verify that the specified requirements are incorporated in the designed model and will behave properly in all scenarios.

3.3.2 Entire SmE Verification

SmE are integrated environments and promise to deliver services in an intelligent requirements-accomplished way. As mentioned in Section 1, SmE covers different aspects of given areas, the verification of the behavior of individual components along with their interaction in the entire system can be formally performed by using model checking or theorem proving techniques, for ensuring the specified SmE behavior, reliable interaction along with the safety and security constraints.

3.4 Adopted Procedures/Tools

In this classification, the investigation of the verification processes is performed on the basis of the adopted procedures (through which the comprehensive verification of correctness of system is analyzed with respect to specified requirements) and tools. During the investigation, the maturity of surveyed technique is analyzed in terms of automation, scalability, adopted tool and the examined scenario (case study).

4 Surveyed Literature

Various techniques regarding the modeling and verification of SmE and their related components are analyzed under the empirically derived parameters (explained in section 5). Although various literature on SmE is available, the papers considered for this survey encapsulate the formal modeling and verification techniques; providing the SmE developers and designers with a specific study material aimed at collecting SmE-centric work. Though during survey a number of techniques were found which extend the logic for developing the tool ((Birkedal et al, 2006; Mascolo et al, 2009; Siewe et al, 2011; Roman et al, 2007)), verification of the protocols ((Forejt et al, 2011; Mottola et al, 2010; Li and Regehr, 2010)) and modeling of the cognition-based user behavior ((Bolton et al, 2012; Biswas et al, 2010)), such techniques are out of the scope of this paper and therefore are not included.

In (Ahmed and Tripathi, 2003), the authors present static verification of security requirements for CSCW (Computer Supported Cooperative Work) systems using finite state techniques (model checking). They use a role based collaboration model for specifying coordination and security constraints of CSCW systems. The completeness and consistency of the specification is ensured by verification within the global requirements. They have developed a number of verification models for checking security properties (task-flow constraints,

information flow or confidentiality) and assignment of administrative privileges. Their primary contribution is a methodology for verification of security requirements during designing of collaboration systems. Finally, they have run a rather peculiar case study of collaborative activities of academic nature. It is our understanding that replacing the components of this case study with SmE devices, it can also be applied on a complex system.

In (Augusto and Hornos, 2013), the authors propose MIRIE (Methodology for Improving the Reliability of Intelligent Environment) by focusing and motivating on the use of formal methods for the modeling and verification of the reliable behavior of the systems at early design states. The focused components are Users, Devices (sensors, actuators), Control unit and Environment (context) which are attempted to be modeled. The behavior modeling of the system is performed with the use of Promela (Process Meta Language); a language through which the synchronous and asynchronous communication among the components can be modeled as non-deterministic automata and the resultant model can be verified with the use of SPIN model checker. System requirements are specified with the use of LTL temporal logic. Iteratively extending the system model, they explained/guided different properties/features of SPIN model checker. The feasibility of proposed MIRIE is ensured on the Nocturnal (Night Optimized Care Technology for UseRs Needing Assisted Livestyles) project.

In the first part of (Augusto and Mccullagh, 2007), the authors describe important characteristics, parent technologies and the applications of SmE in various domains. Then they present behavior models of various components of SmE. The modeling of each component is performed by using the semantics of finite state machines (network of automata). The controlling component, known as coordinator system, detects the presence of home occupant with the use of seven motion sensors, placed in kitchen, living room, bedroom and bathroom. By sensing some activity, the system specified constraints are checked and the suitable operations are performed. Also the TV component is modeled: the controller deactivates the TV when it is found unattended for a long time. Similarly, an alarm manager component is modeled which continuously monitors the triggers from smoke alarms, burglar alarm and emergency pull cord, and contacts fire brigade, security or nursing unit based on triggers. Other controlling components, such as door-bell manager, telephone manager, temperature system manager, environment manager and vital signs monitoring are modeled in the form of state machines. After modeling these components, they design different behavioral properties regarding the verification of specifications-accomplished behavior, individual component behavior, safety and security with the use of Timed Computation Tree Logic (TCTL) by considering the timing factor (real-time system). For verifying these properties on the model, UPPAL is suggested as model checking tool. The process of system modeling and properties designing is performed manually.

In (Benghazi et al, 2012), the authors, having worked in the area of verifying AAL (Ambient Assisted Living) systems, present a verification approach for checking the satisfaction of non-functional requirements, such as timeliness and safety based on timed traces semantics and UML-RT models (MEDISTAM-RT). They use a real-time system design and analysis methodology based on the semi-formal UML-RT models (which are generally recognized to be well suited for designing complex time-constrained systems) and the formal CSP+T notation. In their methodology, the system is designed in a stepwise refinement manner, where components are divided hierarchically into sub-components till the final level. The behavior of these basic components are separately designed by Timed State Diagrams (TSD) and the behavior of the whole is derived from the behavior of its constituent parts by following a compositional specification process based on CSP+T. Their methodology is aimed at ensuring safe deadlock-free communication between components. The authors verify an Emergency Assistance System using this verification approach.

In (Bernardeschi et al, 1998), the authors present a formal verification environment for ensuring the desired behavior along with the 'safety' and 'liveness' properties. A case study of Computer Based Railway Interlocking system is reported in which all the communication is controlled through a sophisticated control unit. The system behavior is modeled by using the formalism of Calculus of Communicating Systems (CCS). Just Another Concurrency Kit (JACK) is used as a model checking tool and the properties are specified by using ACTL logic. The model is abstracted by using the "Zooming" technique. In case of need for more reduction, the "Testing signal values" and "Static configuration parameters" techniques can be applied on the model. The system modeling and properties designing process is manually performed.

In (Bonhomme et al, 2008), the authors present their work for the modeling and verification of SmE. In their methodology, the design process is based on the Systems Engineering standards, especially on EIA-632. In the design process, UML2 and SYSML standard diagrams are used. They take the example of energy manager system (known as ERGDOM) for home comfort. The ERGDOM is a self-configuring system, which identifies the users' comfort patterns, habits and the cur-

rent temperature of the home, and accordingly makes the environment comfortable by controlling the functionalities of HVAC systems, shutters, air-conditioning and convectors. The specification related to main functionalities (e.g. Measurement of temperature, moisture, luminosity and air quality in each part of the home for home comfort), the roles of each component (users or devices) and their interaction with the system, is formalized by the use of context diagrams. Then with the adoption of use-case diagrams, the use cases of the system for the desired services (goals) by the users and devices are designed. Further, the behavior of each use case with their interaction is formulated by using sequence diagrams. These sequence diagrams are usually detailed due to the controlling aspects and are then summarized into the concise activity diagrams. These activity diagrams, based on the automatic translation of the ERGDOM model and their relation, are converted into Petri-net by adopting HiLes functional formalism. Various temporal properties, related to the structure and dynamic behavior verification of the control model of ERGDOM, are verified by using TINA (a model checking tool for Petri-net formalism).

In (Boytsov and Zaslavsky, 2013), the authors propose a method for the verification of the context and situation in pervasive computing environments. As devices are the basic elements of SmE and each device has some capabilities (features), which can be controlled by changing its value. These values can be non-numeric (a lamp can be *on* or off) or numeric (the light intensity of dimmer lamp can be controlled from 0% to 100%). During the verification of the context/situation of SmE, it is essential to confirm the particular feature values of concerning devices. A context/situation (defined by some experts) may be associated, through the relation of generalization, composition, dependence or contradiction, with other contexts (Ye et al, 2012). The modeling of the context is performed with the use of Context Space Theory (CST) which is further formalized in property format by using Situation Algebra Expression. On the basis of the rules defined in (Padovitz et al, 2008; Zadeh, 1965; Ye et al, 2012), they designed 3 algorithms by which the context modeling can be converted into the Orthotope-based situation space and situation algebra expressions (Boytsov and Zaslavsky, 2013). These expressions are further checked on the Orthotope-based situation space for identifying the emptiness or counterexample in case of validation. The feasibility of the proposed methodology is confirmed with the example of Smart Office Environment. It is opportune to note that this paper does not refer the context as the user location.

In (Corno and Sanaullah, 2011a,b, 2013), the authors present generic methodologies for the formal modeling and verification of SmE and its related components. The suite of methodologies is holistic in its nature: not only the rules and important points to be considered for the modeling and verification of entire SmE, but also the consistency amongst the various modeling formalisms is checked and corrected wherever required. The interface modeling is performed with the use of an ontology, whereas the behavioral modeling is performed with the use of statecharts. They perform consistency checking among both interface and behavioral formalisms and then the detailed internal behavioral verification of the individual devices (Corno and Sanaullah, 2011b), for this, they use the example of dimmer lamp. Whereas in their other papers (Corno and Sanaullah, 2011a, 2013), they present a generic methodology for the formal modeling and verification of SmE. For the verification purposes, they use UCTL temporal logic for describing the requirements in the form of properties and UMC as a model checking tool. The methodology is tested by using a case study of a Bank Door Security Booth (BDSB) system.

In (Coronato and Pietro, 2010a,b, 2011), the authors present their work for the modeling and verification of ambient intelligence applications. The authors' key focus is on the location dependent movement of users (also referred as ambient) by incorporating the concepts of Pervasive and Ubiquitous Computing. They propose a seven step process for the interface, behavioral and constraints modeling of SmE. The Ontology is used for interface modeling where ambient calculus (AC) (a process calculus formalism derived from pi-calculus) is used for behavioral modeling. They theoretically extend AC by borrowing the concepts from different formal modeling techniques for incorporating the real-time constraints and conditional movements which are among the limitations of AC. The properties related to the pervasive and ubiquitous concepts are specified in terms of Ambient Logic (AL) – having a combined power of propositional logic, first-order-logic, temporal logic, somewhere and everywhere operators– the properties related to explicit real-time constraints are specified in terms of Real-Time Temporal Logic (RTTL) and the properties related to the pre-and-post conditions are specified by using Design-By-Contract (DBC). A case study of a patient monitoring system is modeled according to the above mentioned formalisms. The patient's movements among different rooms, their activities and operations are identified with the use of RFID Tag. For the modeling and verification, they develop a tool, known as Ambient Designer which can visually model the system in the form of AC and AL. It has

an additional functionality of translating the model in the acceptable language of NuSVM model checker. The designed model can be verified by implementing the model checking algorithm for AC in the designed tool or by using NuSVM model checker tool. By this, the properties related to the functional correctness, reliability, availability, safety and security of the system can be verified.

In (Gnesi et al, 1999), the authors present a branching time model-checking approach for the formal verification of dynamic aspects of complex systems. Authors defined some formal semantics, based on the work of (Latella et al, 1999b) and JACK (model checker), for considering the dynamic aspects of the system (described in the form of Hierarchical Automata). For the verification, authors consider the Statechart modeling of user interaction with TV system. The dynamic behavioral properties are specified in ACTL logic and verified on the model with the help of JACK model checking tool. The syntax and static semantics of Statecharts are formally defined; however their dynamic aspects are informally defined.

In (Gnesi and Mazzanti, 2004), the authors present the UMC model checker tool for the formal verification of the dynamic behavior of complex systems. The systems which can be verified through UMC are required to be specified in the form of UML communicating Statecharts which can interact with others. The system requirements are formalized by using the syntax and semantics of mu-ACTL logic (ACTL logic with the complete power of mu-calculus as well) and verified on the model with the use of UMC. A case study of the system, consisting of two airports, two passengers and an airplane, is considered for showing the satisfactory outcomes of the model checker.

In (Hoogendoorn et al, 2009, 2013), the authors present an agent based ambient system for the formal modeling and verification of the interaction among multi-agents. For the generic and domain specific behavior modeling of the interaction, they used predicate logic. And for the verification of the specification, they used rule based Temporal Trace Language (TTL) (Bosse et al, 2009), which is specially designed for the formal specification and analysis of dynamic properties, regarding the qualitative and quantitative (in-term of time) interaction aspects of the systems belonging to biological, cognitive and social domains. They have modeled the Medicine Usage Management system, in which patient takes medicines from the intelligent Medicine box (which has the ability of knowing the quantity of the dosage and the time of previously taken medicine). On crossing the threshold values (maximum and minimum quantity of dosage and time), the system notifies with beep and by automatically sending the SMS to the patient. In case of no reply (or response) from the patient, the system sends a history SMS to relevant doctor. For the modeling of each component (agent), input, internal and output states are considered in predicate logic format (referred as Ontology). A stochastic model of the patient is considered, and interaction of the model system is sent to the LEADSTO (Bosse et al, 2007), which executes and simulates the traces of the system. The TTL properties are also analyzed on the modeled system (by using these traces) through TTL checking tool (Bosse et al, 2009) or by using SMV model checker[1].

In (Ishikawa et al, 2009), the authors present a theoretical framework for the formal modeling of SmE by concentrating upon the concepts of Pervasive computing. They perform the formal modeling of requirements, assumptions and behaviors of application software with respect to the user (identification, movements, scopes) and the accessible features of the surrounding devices. According to the requirements and assumptions, the abstract interaction modeling of the accessible features of the devices (by the users at some certain time) is performed, which is further analyzed and formally verified. For the behavioral modeling of such system, Event Calculus is used; a formalism for expressing and reasoning the effects of any action (Shanahan, 1999). According to the scopes (the direct interaction of the users with the accessible devices) and duration, the requirements along with the implementation of assumption are modeled in the form of axioms (rules). A theorem proving inference approach is used by adopting Discrete Event Calculus Reasoner (IBM, 2005) as a tool for formally satisfying the system requirements. Discrete Event Calculus is for representing the requirements in the properties format. Discrete Event Calculus is converted into the (well-known) SAT problem and inference is made on the model. With the example of Meeting Support System, they justify the feasibility of their proposed framework.

In (Leelaprute et al, 2005), the authors present their work related to the modeling and verification of the integrated services in the home network system (HNS). They described and modeled the HNS by using the semantic of Object Oriented modeling in which the en-

[1] Moreover, Jan Treur (one of the authors of (Hoogendoorn et al, 2009, 2013)) extended the work by covering other aspects of the agent based ambient system, with the collaboration of other researchers (Aziz et al, 2010; Sharpanskykh and Treur, 2012). In his work (Sharpanskykh and Treur, 2012), the cognitive analysis is performed through simulation (therefore is not including in our survey). The purpose of mentioning this is that it is the only found work which performed cognitive analysis for the ambient system.

vironment, appliances, properties, methods, states and other relevant information are considered. After that, they presented a descriptive language for the modeling of the HNS. Then the services' reliability of HNS is verified. Authors used a case study of a home system in which air conditioner, inside and outside thermometers, smoke sensor, ventilators and windows are modeled. The appliances and their integrated services are verified with respect to different CTL specified properties by using SMV model checking tool.

In (Liu et al, 2012), the authors propose the use of formal methods to analyze the pervasive computing systems. They start with proposing a formal modeling framework for covering the main characteristics of pervasive computing systems. They adopt CSP# for modeling and verification as it is rich in the syntax for modeling concurrent system with hierarchies. Later, the safety requirements are identified and the specification patterns for safety and liveness properties are provided as they have classified the important requirements into these categories. By doing so, the critical properties against the system model can be verified by using model checking to detect the design flaws at the early design stage. Finally, a case study of a smart health care system for mild dementia patients (AMUPADH) is run to demonstrate the practicality of proposed framework.

In (Masci et al, 2012, 2013a,b), the authors' main goal is to verify the (software of) medical devices by using their UIs. They verify the devices by adopting different strategies. They investigate the user's actual behavior in the field and verify it with the prescribed one as mentioned in the user manual (Masci et al, 2012). Similarly, they provide a solution to the investigation authorities for verifying as to which specified user-interface requirements are satisfactorily incorporated in the medical device after their implementation (Masci et al, 2013a). Moreover, they extend their work and investigate the interaction design issues in the implementation by generating the keying sequences (data entry task) and analyzing them with the user-interaction behavior (Masci et al, 2013b). For the verification purpose, they adopt theorem proving approach. The Prototype Verification System (PVS) is adopted as a theorem proving tool, and the model of the system is designed by using the reverse engineering processes. The designed model is further translated into the acceptable format of PVS, which is based on higher ordered-typed logic and equipped with similar features of various languages (like C++). The requirements which are required to verify are formalized into axioms (according to the template for properties) and then verified on the model. The verification process is performed in (Masci et al, 2012, 2013a) by using proof obligations component of PVS and in (Masci et al, 2013b) by using configuration diagrams (a labeled graph of the modeled system/device in which nodes represent configurations and edges represents transition with guard conditions). These configuration diagrams help in generating the test cases for exposing the interaction issues in the model. With the case study of glucose monitoring procedure in oncology ward (Masci et al, 2012), infusion pump (Masci et al, 2013a) and a layout of medical device (Masci et al, 2013b), they proved the authenticity of their work.

In (Ranganathan and Campbell, 2008), the authors present their theoretical contribution for the modeling and verification of pervasive computing environment. They consider the software controlling components, devices, users, environment and other physical objects in the environment as ambient which are spatially interrelated with other objects. Along with the movement, an ambient can enter or leave the environment and can be part of other environments. The modeling of these ambient along with their operations and activities are performed with the use of Ambient Calculus. The properties related to verifying the availability of the services at anytime and anywhere, and devices mobility in case of changing their context (entering and existing of ambient in other environment) are performed with the use of ambient logic. A case study (named as Gaia) of university is considered which is equipped with multiple sensors, computers and actuators. Students can enter with their digital devices (mobiles, PDAs, laptop) and can perform various pervasive activities. Different model checking algorithms/tools, such as specified in (Charatonik and Talbot, 2001), can be used for the verification of the pervasive properties.

5 Empirically-derived Parameter-based Methodology

For the development of SmE, it is evident from the literature to firstly design and verify the system (motivation is given in Section 1). Practically, project manager (along with the team) may have many questions regarding the modeling and verification of the system. As SmE has the capacity to cover various domains with different perspective, different techniques and tools are used – according to their application areas and covered aspects – for the modeling and verification. On the basis of our experiences and surveyed literature, we try to identify and classify the emerging concerns (listed below) into four groups.

- Among the basic components of SmE (mentioned in Section 2), which components are required to be modeled for this specific application area;

- For the modeling of the selected components, which aspects are required to be covered;
- How the modeling of each selected component is performed by considering the level of details necessary to be achieved?
- How the intelligence/computation is modeled by considering the system constraints?
- How the requirements of different perspectives are modeled, for confirming the correct incorporation in the system model?
- How the verification of the different aspects/perspectives of the components or system is performed?
- Which techniques and tools are used for the modeling and verification of the system?
- Which application area is selected for proving the reliability of the proposed approach?
- Which abstraction technique is employed/adopted for reducing the size of the model so that the verification can be easily performed by focusing on the interested perspective?

During the first course of the literature survey, these questions were identified and classified according to criteria mentioned in Section 3. A deep analysis of each classification with the internal categorization is carried out in the second round, and termed as *parameter*. In the third round of survey, the existing state-of-the-art against each parameter is identified and analyzed according to its modeling/verification capacity, termed as *parameter values*.

To the best of our knowledge, the existing state-of-the-art of formal modeling and verification processes (as described in Section 3), with respect to their level of adoption and application scenarios, may be comprehensively represented in a tabular form, in which each formal parameter is represented by the adopted state-of-the-art (parameter values) against the surveyed literature. The complete procedure of designing tabular form (from extracting parameters to their corresponding values against each surveyed literature) is referred as empirically-derived parameter-based methodology.

In order to perform an in-depth analysis of the surveyed literature, an overview of the existing state-of-the-art techniques has been performed. The details of their application domains, level of adoption, and their corresponding scenarios are presented in the subsequent sections. Moreover, uncovered areas by the existing state-of-the-art processes and commonly used ones are also investigated.

The following subsections are the main classification, against each of which, a table is designed that provides the adopted tools/techniques information against each surveyed literature. The inner subsections of this classification work as parameters of these tables. These inner subsections represent different perspectives which may be adopted during the formal modeling and verification, in the surveyed literature. The values of the parameter represent the formalism (existing state-of-the art) or the adoption of perspectives in the surveyed literature.

5.1 Formal Modeling

5.1.1 Black Box Modeling

Different formalisms are used for Black Box modeling such as Structure diagrams (Class diagrams, Object diagrams) (Booch et al, 1998) and Ontologies (Fensel, 2001). Structure diagrams are Unified Modeling Language (UML) artifacts that model the Object Oriented systems, whereas Ontologies are the semantic web solution for describing the data as complete data model, formal semantics, knowledge discovery and sufficient reasoning power; due to these advantages, Ontologies are often preferred for the modeling of SmE.

Black box, as a parameter in our methodology, is used for representing the explicitly adopted formalism of modeling information. In tabular format, this parameter is either represented with the name of the employed formalism or with a cross mark (✘), indicating that it is not adopted (as represented in Table 1).

5.1.2 White Box Modeling

The behavioral modeling of SmE can be performed through UML behavioral diagrams (Booch et al, 1998), process calculus (Baeten, 2005; Bergstra and Klop, 1984) and petri-nets (Nielsen et al, 1981; Bonhomme et al, 2008).

UML behavioral diagrams consist of Use Case, Activity, Sequence, Statecharts and other diagrams. Statecharts (Automata or labeled transition systems) are commonly used artifacts for specifying the system in a formal way. Different variants of state diagrams for modeling different aspects of behaviors, with each variant having its own limitations, are designed. The more famous and exploited variants are Harel Statecharts (Harel, 1987), Communicating Statecharts (Gnesi and Mazzanti, 2004), Automata (Hopcroft et al, 1979; Augusto and Mccullagh, 2007) and Hierarchical Automata (Mikk et al, 1997; Gnesi et al, 1999). The probabilistic and timed behavior of the complex system can be modeled with the use of Probabilistic Statecharts (Jansen et al, 2002) and Timed Automata (Wang, 2004), respectively.

Process algebras can also be represented as labeled transition systems for specifying the behavior of the system. In process calculus, the most commonly used for-

Table 1 Modeling Evaluation

Researchers	Black Box Modeling	White Box Modeling	Intelligence Modeling	Requirements Modeling
Ahmed and Tripathi (Ahmed and Tripathi, 2003)	✘	Role based collaboration model	Role based	LTL
Augusto and Hornos (Augusto and Hornos, 2013)	✘	Activity Modeling Through Promela processes	Event (Activity detection), Condition(location identification), Action (operation graded)	LTL
Augusto and McCullagh (Augusto and Mccullagh, 2007)	✘	Finite State Machine	Event Condition Action	TCTL
Benghazi et al. (Benghazi et al, 2012)	✘	UML-RT (Timed Sequence Diagram, Timed State Diagram), CSP+T	Event Condition (previous history) Action	F_{TT} (Common Formal Semantic Domain)
Bernardeschi et al. (Bernardeschi et al, 1998)	✘	CCS/MEIJE Process Algebra	Event Condition Action	mu-ACTL
Bonhomme et al. (Bonhomme et al, 2008)	System Engineering Standards, EIA-632, Use Case, Sequence, Activity and Dynamic Context Diagrams, UML2, SYSML	Petri-Nets, HiLes	Decision Logic	Temporal Properties
Boytsov and Zaslavsky (Boytsov and Zaslavsky, 2013)	Context Space Theory (CST)	Orthotope-based Situation Space	Weighted Rule Based	Situation Algebra Expression
Corno and Sanaullah (Corno and Sanaullah, 2011a,b, 2013)	Ontology	Statecharts	Event Condition Action	UCTL
Coronato and Pietro (Coronato and Pietro, 2010a,b, 2011)	Ontology	Ambient Calculus	Ambient movement, Pre-and-Post conditions	Ambient logic + RTTL
Gnesi et al. (Gnesi et al, 1999)	✘	Hierarchical Statecharts	Event Condition Action	ACTL
Gnesi and Mazzanti (Gnesi and Mazzanti, 2004)	✘	Communicating State Machines	Event Condition Action	mu-ACTL
Hoogendoorn et al. (Hoogendoorn et al, 2009, 2013)	✘	Predicate logic	Rule Based	TTL
Ishikawa et al. (Ishikawa et al, 2009)	✘	Event Calculus	Rule Based	Axioms Based through Discrete Event Calculus
Leelaprute et al. (Leelaprute et al, 2005)	Object Oriented Modeling, System description	Object Oriented Modeling, Service description	Event Condition Action	CTL
Liu et al. (Liu et al, 2012)	✘	CSP#	Rule Based	LTL
(Masci et al, 2012, 2013a,b)	✘	PVS Logic, a Typed higher-ordered Logic	✘	Axioms Based (according to property template)
Ranganathan and Campbell (Ranganathan and Campbell, 2008)	✘	Ambient Calculus	Rule Based, DL-Based, Relational Algebra	Ambient Logic

malism are Calculus of Communicating Systems (CCS) (Bernardeschi et al, 1998; Hennessy and Milner, 1985), Communicating Sequential Processes (CSP) (Hoare, 1978; Brookes, 1983) and Pi-calculus (Milner et al, 1992), whereas their extension with the context-aware (mobility) modeling information is known as Ambient Calculus (AC) (Coronato and Pietro, 2010b; Cardelli and Gordon, 1998). The probabilistic modeling of the system is mostly performed by enhancing the semantics of process calculus formalisms.

Petri nets are used as framework for specifying the concurrent systems with detailed (mathematical and conceptual) basic semantic for their modeling. Timed-petri-nets is an extension of petri-nets, in which the concurrent behavior of the system is formally specified in terms of time.

White box, as a parameter in our methodology, is used for representing the adopted formalism of modeling information. In tabular format, the value of this parameter is represented with the name of the employed formalism.

5.1.3 Intelligence Modeling

For providing services intelligently, different techniques are adopted among which artificial intelligence (e.g. fuzzy logics in (Hagras et al, 2004; Pedrycz, 2010), decision trees in (Stankovski and Trnkoczy, 2006), machine learning in (Cook et al, 2006), case-based reasoning in (Kofod-

Petersen and Aamodt, 2006), rule-based reasoning in (Boytsov and Zaslavsky, 2013), databases (e.g. event-condition-action in (Corno and Sanaullah, 2011a) and SQL-based data management in (Feng et al, 2004)) are some of the mostly adopted approaches. Based on these approaches, control algorithms decide feasible operations and send commands accordingly to corresponding devices.

In empirically-derived parameter-based methodology, intelligence modeling is used as a parameter (see Table 1). The value represents the name of the employed formalism by the surveyed technique and cross mark (✘) indicates that it is not observed in the surveyed technique (as represented in Table 1).

5.1.4 Requirements Modeling

Temporal Logics are widely used in formal verification in order to formalize and specify the requirements of complex systems (Clarke et al, 1986; Nicola, 1995; Nicola and Vaandrager, 1990; Manna and Pnueli, 1992). 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) of the complex systems. In addition to Exists and All, there are other temporal quantifiers like Global, Next, Future, Until, Implies, which help in verifying the complex requirements on different branches from some specific state at a certain time.

Linear-Time Temporal Logic (LTL) is used to represent the requirements for linear time model of the system, whereas Action Based Branching Time Logic (ACTL) (Nicola and Vaandrager, 1990) and State Based Branching Time Logic (CTL) (Clarke et al, 1986) are used for representing the requirements for computational time temporal logic of the system. Several logics are designed for handling different aspects of requirements, many are formulated by integrating the already designed languages addressing a wider range of requirements like UCTL (Beek et al, 2011), SocL (Fantechi et al, 2008). Time based requirements are usually handled by TCTL, RTL, RTTL, TPTL, RTCTL (Alur and Henzinger, 1992) whereas probabilistic requirements are handled by using PLTL and PCTL logics (Reynolds, 2005).

In our methodology, requirements modeling is used as a parameter and the value (in Table 1) reports the adopted logic by the surveyed technique.

5.2 Component Modeling

5.2.1 User Modeling

Users interact with the SmE in their own ways which, in turn, responds according to the specified and modeled behaviors. The level of details and sophistication varies from system to system, context to context and goals to goals. Among different perspectives, some of the behavioral aspects which are considered for user modeling are:

- User identification (UI): the identification of the user through sensing and/or input devices;
- User actions history (UH): the stored history of previous user actions;
- User privileges –on the basis of their roles– (UPr): based on the role categorization, the system functionality provision granted to the user;
- User position –pre- and post-action execution– (UP): the geographical location of the user within the system boundaries with respect to a specific action;
- User's possible actions (UA): the actions of the user which can be contemplated and facilitated by the system;
- User's possible behaviors (UB): the behavior (related to movement and context-approved actions) of the user which can be contemplated and facilitated by the system;

In Table 2, the values at the end of listed items (placed in parenthesis) are used as parameter values for representing the modeling aspects covered by the referring technique.

5.2.2 Devices Modeling

Device modeling can be done by two methodologies: interface and behavior. In interface modeling, we usually consider commands (triggers) a device may receive; associated functionality (operation) it may perform; constraints (rules) it has to follow; states at which it will be at any time; notifications that it sends after the completion of task. Whereas in behavior modeling, acceptance of specific commands on a particular state, implementation of constraints, operations which may be performed on that state after the satisfaction of constraints are considered.

Referring to Table 2, the value "Behavior" under this category show the modeling of internal behavior of the devices in the surveyed technique.

Table 2 Component Modeling

Researchers	Users Modeling	Devices Modeling	Control Modeling	Context Modeling	Interaction Modeling
Ahmed and Tripathi (Ahmed and Tripathi, 2003)	UPr, UA		✔		UI, IC, CO
Augusto and Hornos (Augusto and Hornos, 2013)	UI, UP, UA, UB		✔	✔	US, UC, SC, CO
Augusto and McCullagh (Augusto and Mccullagh, 2007)	UA	Behavior	✔		US, UI, SC, IC, CO
Benghazi et al. (Benghazi et al, 2012)	UH, UA		✔		US, UI, SC, IC, CO
Bernardeschi et al. (Bernardeschi et al, 1998)			✔		IC, CO
Bonhomme et al. (Bonhomme et al, 2008)	UI, UH, UA		✔		US, UI, SC, IC, CO
Boytsov and Zaslavsky (Boytsov and Zaslavsky, 2013)			✔		IC
Corno and Sanaullah (Corno and Sanaullah, 2011a,b, 2013)	UI, UP, UA, UB	Behavior	✔	✔	US, UC, UI, SC, IC, CO
Coronato and Pietro (Coronato and Pietro, 2010a,b, 2011)	UI, UP, UB		✔	✔	US, UC, SC, CO
Gnesi et al. (Gnesi et al, 1999)	UA	Behavior			UI
Gnesi and Mazzanti (Gnesi and Mazzanti, 2004)	UA, UB	Behavior		✔	UC, UI
Hoogendoorn et al. (Hoogendoorn et al, 2009, 2013)	UH, UA		✔		UI, IC, CO
Ishikawa et al. (Ishikawa et al, 2009)	UI, UPr, UP, UA		✔	✔	US, UC, UI, SC, CO
Leelaprute et al. (Leelaprute et al, 2005)		Behavior	✔		IC, CO
Liu et al. (Liu et al, 2012)	UI, UA		✔	✔	US, UC, UI, SC, IC, CO
(Masci et al, 2012, 2013a,b)	UI, UPr, UA	Behavior			UI
Ranganathan and Campbell (Ranganathan and Campbell, 2008)	UI, UA		✔	✔	US, UC, UI, SC, IC, CO

5.2.3 Control Algorithms Modeling

The overall sophisticated control strategy of SmE is implemented through control algorithms. Control algorithms take input from the input/sensing devices and according to the system specifications and imposed constraints, decide for the reliable functionality. For the fulfillment of the desired functionality, they send commands to the relevant operating devices for performing required task/operation (as presented in Figure 1).

In Table 2, a tick mark (✔) under this parameter show that the referred technique takes decision by implementing the mentioned pattern.

5.2.4 Context/Environment Modeling

The identification of the user location is grouped in this category, and termed as Context modeling. Referring to Table 2, a tick mark (✔) under this category shows the referring technique performed this type of modeling.

5.2.5 Interaction Modeling

SmE components can interact with each other for the achievement of desired goals. In the surveyed literature, researchers are found focusing on different interaction levels and accordingly building the system. On the basis of these focuses, we categorized the interaction levels into the following groups:

- User interaction with the environment through sensors (US): the considerable user actions in the environment are monitored or noticed with the use of sensors;
- User interaction according to its context (UC): the user actions are recognized according to user's movements in the environment; although these are usually monitored by sensors, the focus point is that with a change in the position, the system will able to consider the activities;
- User action performance on input devices (UI): user interacts with the system through handheld devices,

or by directly performing action on the real inputting devices;
- Sensor interaction with the control algorithms (SC): the sensed data is sent by the sensors to the control algorithms, on the basis of which control algorithms decide for the preferable action;
- Input device interaction with the control algorithms (IC): the handheld devices or real devices send the commands to the control algorithms for performing the specific task;
- Control algorithms interaction with the operating devices (CO): control algorithms incorporate the intelligence strategies and on the basis of incoming commands, decide for the preferable action and accordingly send messages to the relevant devices.

In Table 2, the values presented at the end of each listing item are used as the parameter values for informing that the referred technique is focusing/performing on which type of interaction modeling.

5.3 Formal Verification

The system correctness with respect to its specifications and constraints can formally (comprehensively) be verified and this process is known as formal verification. Different aspects are verified during the verification process. The description of each aspect is presented in the following subsections.

5.3.1 Consistency Verification

Consistency verification, as a parameter in our methodology, is used for representing whether applied modeling formalisms are consistent with respect to their vocabulary and functionalities, and the specified requirements are properly incorporated in the designed system (as mentioned in Section 3.3.1). In Table 3, a tick mark (✔) shows it is considered and performed in surveyed literature.

5.3.2 Entire SmE Verification

In order to verify entire system, different aspects are covered, which can be classified as the following: Users Behavior Verification, Context Verification, Device Behavior Verification, Devices Interaction and Control Verification, Real Time Verification and Probabilistic Verification.

- *Users Behavior Verification:* The key concern while designing SmE is to facilitate the environment with integrated technologies to benefit users, who have certain goals/desires and a complex web of behaviors which can be adopted during interaction with the system. In this classification, accomplishment of user goals against the specified actions with the input devices (or sensors) and the understanding of the possible behavior (moves) of the users are verified. The tick (✔) sign under this category shows its application in verification of users actions and behavior.
- *Context Verification:* Users interact with the SmE through the environment. According to location (also referred as Context), users can access services (mostly concerned with safety and security) from the environment. The environment models of SmE have extra computational power for determining the current state of the corresponding objects/devices/users and providing specified services accordingly. For instance, room illumination services are only accessible when residents are awake and/or present in room. Table 3 reports whether the surveyed technique performs context verification or not; the tick (✔) sign shows context verification is performed.
- *Device Behavior Verification:* The devices in SmE are of heterogeneous nature with some common and distinct features. They are self-dependent components with their own internal specified behavior that may be complex based on the device features (smart devices). In this classification, the specified internal behavior of devices is explicitly confirmed on their models. The tick (✔) sign in Table 3 shows scenarios in which this verification is performed.
- *Devices Interaction and Control Verification:* The system level requirements are implemented through control algorithms which regulate interaction among devices. In this classification, the system level constraints and the reliable interaction among devices under control algorithms are confirmed. The tick (✔) sign under this category shows it has been applied.
- *Real Time Verification:* The application areas of SmE are almost in every domain. Some applications can be time dependent such as traffic control system, where time factors are also considered in modeling and verification stage. In this classification, real-time verification of the system is ensured. The tick (✔) sign in Table 3 indicates real time verification is performed.
- *Probabilistic Verification:* The system being large along with possibilities of its multi-tasking nature make it more complicated. Probabilistic modeling, in this regard, can be adopted to ensure its smooth behavior with respect to possible actions the system can perform at a given time. SmE may encounter

Table 3 Formal Verification Evaluation

Authors	**Consistency Verification**	Entire System Verification					
		Users Behavior Verification	Context Verification	Device Behavior Verification	Devices Interaction Control Verification	Real Time Verification	Probabilistic Verification
Ahmed and Tripathi (Ahmed and Tripathi, 2003)		✔			✔		
Augusto and Hornos (Augusto and Hornos, 2013)		✔	✔		✔		
Augusto and McCullagh (Augusto and Mccullagh, 2007)				✔	✔	✔	
Benghazi et al. (Benghazi et al, 2012)	✔				✔	✔	
Bernardeschi et al. (Bernardeschi et al, 1998)					✔		
Bonhomme et al. (Bonhomme et al, 2008)	✔(Behavioral Analysis)				✔		
Boytsov and Zaslavsky (Boytsov and Zaslavsky, 2013)	✔						
Corno and Sanaullah (Corno and Sanaullah, 2011a,b, 2013)	✔	✔	✔	✔	✔		
Coronato and Pietro (Coronato and Pietro, 2010a,b, 2011)		✔	✔			✔	
Gnesi et al. (Gnesi et al, 1999)				✔			
Gnesi and Mazzanti (Gnesi and Mazzanti, 2004)		✔	✔				
Hoogendoorn et al. (Hoogendoorn et al, 2009, 2013)					✔	✔	
Ishikawa et al. (Ishikawa et al, 2009)		✔	✔		✔	✔	
Leelaprute et al. (Leelaprute et al, 2005)				✔	✔		
Liu et al. (Liu et al, 2012)			✔		✔		✔
(Masci et al, 2012, 2013a,b)	✔			✔			
Ranganathan and Campbell (Ranganathan and Campbell, 2008)		✔	✔		✔		

challenging conditions such as versatile user behaviors, malfunctioning sensors, broken or out-of-order devices, which may compromise reliable response of the system. To cater to such scenarios, probabilistic modeling and verification is usually performed. In this classification, checking of probabilistic verification in surveyed literature is taken into consideration (see Table 3).

5.4 Adopted Procedures/Tools

In this category, analysis of verification procedures/tools is considered. The following subsections explain in details.

5.4.1 Formal Verification Techniques

Model checking is suitable for the system in which the state space is finite (Clarke et al, 1994) but it can also work for infinite state space models represented as a finite state space by adopting some reduction technique (such as abstraction, inactive variable elimination, internal transition by passing, approximation). Several model checking tools are available for the formal verification of SmE related systems. The verification can be performed using Linear-Temporal Logics or Branching-Time Temporal Logics. Some of the reported model checkers that use Linear-Temporal Logics are in (Latella et al, 1999a; Gallardo et al, 2002; Mikk et al, 1998), HEGO (Schafer et al, 2001), vUML (Lilius and Paltor, 1999) based on SPIN (Holzmann, 1997), whereas JACK, (Gnesi et al, 1999), SMV (McMillan, 1992), CMC (Beek et al, 2009) and UMC (Gnesi and Mazzanti,

Table 4 Adopted Procedures/Tools

Researchers	Verification Technique	Abstraction	Automatic	Scalability	Verification Tool	Case Study
Ahmed and Tripathi (Ahmed and Tripathi, 2003)	Model Checking	incremental modeling with separation of concerns and property specific abstractions	Automatic		SPIN	Computer Supported Cooperative Work (CSCW) system for Monitoring Exam Activities
Augusto and Hornos (Augusto and Hornos, 2013)	Model Checking	✘	Manual	✔	SPIN	Nocturnal (Night Optimized Care Technology for UseRs Needing Assisted Livestyles)
Augusto and McCullagh (Augusto and Mccullagh, 2007)	Model Checking	✘	Manually	✔	UPPAL	Smart Home
Benghazi et al. (Benghazi et al, 2012)	Transformation and Mapping rules	✘	Semi-automatic	✔	✘	Emergency Assistace System for Cardiac patient
Bernardeschi et al. (Bernardeschi et al, 1998)	Model Checking	Testing Signal Values, Static configuration parameters, Zooming	Manually		JACK	Computer Based Railway Interlocking System
Bonhomme et al. (Bonhomme et al, 2008)	Model Checking	✘	Semi-automatic		TINA	Smart Energy Management System for Home Comfort (EDGDOM)
Boytsov and Zaslavsky (Boytsov and Zaslavsky, 2013)	Rule Based	✘	Manual		Self-designed Algorithms for Emptiness Check	Smart Office Environment
Corno and Sanaullah (Corno and Sanaullah, 2011a,b, 2013)	Model Checking	State and Action based	Semi-automatic	✔	UMC	Dimmer Lamp, Bank Door Security Booth System (BDSB)
Coronato and Pietro (Coronato and Pietro, 2010a,b, 2011)	Model Checking	✘	Semi-automatic	✔	Ambient Designer, Nu-SMV	Pervasive Healthcare Application for Monitoring the Patient
Gnesi et al. (Gnesi et al, 1999)	Model Checking	Refinement Function	Manually		JACK	User and TV System
Gnesi and Mazzanti (Gnesi and Mazzanti, 2004)	Model Checking	not generating the global model of the system	Manually	✔	UMC	Plane and Passenger in Airport System
Hoogendoorn et al. (Hoogendoorn et al, 2009, 2013)	Model Checking	✘	Semi-automatic		TTL Checker, SMV	Medicine Usage Management
Ishikawa et al. (Ishikawa et al, 2009)	Theorem Proving	✔	Manual		Discrete Event Calculus Reasoner	Meeting Support System
Leelaprute et al. (Leelaprute et al, 2005)	Model Checking	Symbolic representation of the State space	Semi-automatic		SMV	Air Cleaning Service in Home Network System
Liu et al. (Liu et al, 2012)	Model Checking	✘	Semi-automatic	✔	PAT	Heath care system for Dementia patient
(Masci et al, 2012, 2013a,b)	Theorem Proving	✘	Semi-automatic		PVS	Glucose monitoring procedure in oncology ward, Infusion pump, a real medical device
Ranganathan and Campbell (Ranganathan and Campbell, 2008)	Model Checking	✘	Manually		specified in (Latella et al, 1999a; Gallardo et al, 2002; Mikk et al, 1998)	Gaia (pervasive environments with digital devices)

2004) are used for verifying state and action based branching time temporal behavior. For the verification of real-time systems, UPPAL (Larsen et al, 1997) and nuSVM (Cimatti et al, 2002) model checkers are used, while TINA (Berthomieu and F.Vernadat, 2006), TAPAAL (Byg et al, 2009), ROMEO (Gardey et al, 2005), DREAM (Madl et al, 2006) are exploited when model is specified in terms of Petri-Nets. Time based verification can be performed with the use of UPPAL, TAPAAL, ROMEO, DREAM, CWB (Stevens and Stirling, 1998) and other model checkers, whereas probabilistic model checking can be performed with the use of CADP (Garavel et al, 2001), PAT (Sun et al, 2009), PRISM (Kwiatkowska et al, 2002) and others.

The formal verification on the system can also be performed with the use of theorem proving techniques, in which the system is modeled using invariants and set-theoretical structures. Different logical inference rules, linear and temporal properties can be applied for checking correctness of the system. Inference can be semi-automatic (with user involvement) or fully-automatic (by providing full power of inference to theorem prover). The commonly used semi-automatic theorem prover are HOL (Harrison, 1996), Coq (Barras et al, 1997), ACL2 (Brock et al, 1996) PVS (Owre et al, 1996) and Isabelle (Paulson, 1989), whereas fully-automatic theorem prover are Perfect Developer (Crocker, 2003) and Escher C (Crocker and Carlton, 2007). The possible scenarios, where these modeling techniques are applied, are described in Table 4.

5.4.2 Abstraction

In case of model checkers, abstraction techniques are frequently used for reducing the size of the system model. The abstraction can be applied on states, actions and variables of the model. Under this parameter, either the name of the abstraction technique explicitly adopted by the surveyed literature is mentioned or the cross (✘) sign indicating that it is not performed.

5.4.3 Automated

This parameter is used to represent that the surveyed technique generates the model and the properties "automatically", or it performs some manual instruction and some part is automatically generated ("semi-automatically") or all work is performed "manually".

5.4.4 Scalability

Scalability is among the important factors which are considered for the evaluation of techniques. It is a broader term and can be used in many dimensions. Here the scalability is referred as the ability of the surveyed technique to enhance itself by adding more components of same or different nature in the system. Under this parameter, the tick (✔) sign indicates our observation that the technique can be enhanced by adding other components with their inner aspects and details.

5.4.5 Verification Tool

This parameter indicates that among the several model checking/theorem proving tools (as listed in section 5.4.1), which tools are adopted and in which domains and scenarios. In Table 4, this parameter (verification tool) contains the name of the applied approaches/processes/tools in verification process.

5.4.6 Example/Case Study

This parameter has the name of the application area, which is selected by the surveyed literature as a case study/example, for proving the satisfactory outcomes.

6 Discussion

In this paper, a survey of SmE modeling techniques is empirically conducted. In survey, the modeling techniques which also perform formal verification for confirming their correct behavior are considered.

As evident in the Table 1, the analysis shows that most of the techniques do not perform black box modeling, but white box modeling is globally performed. The reasoning behind this trend can be attributed to the fact that at least behavior modeling is performed in any case due to the minimum formalism requirement. Black box modeling, on the other hand, plays more of a foundational role (in form of common dictionaries and conventions). This role nevertheless has a considerable planning and development cost. Owing to shortage of time and resources, researchers seem to be in a hurry to furnish the obvious functioning aspects of formal verification rather than the foundation. For white box modeling, most of the techniques consistently use Statecharts (or their variants) due to their maturity and ready availability other than mathematical-oriented wide coverage of all possible paths. Intelligence modeling, mostly provided through Event-Condition-Action technique, is almost globally performed by all the techniques. It is imperative to mention that artificial intelligence (fuzzy logic, decision tree) is not diffused in formal verification practice. Finally, using the variants of temporal logic, most of the surveyed techniques perform requirements modeling.

The analysis of Table 2 shows that minimal user modeling is performed by almost all the surveyed techniques. However, it is opportune to mention that most of the techniques acquire the knowledge of user identification and actions to perform user modeling. The real user behavior modeling is performed by a minority of techniques and a majority does not do so due to pertinent complexity of behavior versatility and uncertainty. Further, all surveyed techniques in device modeling are considering the interaction between the devices. But the behavior of individual devices is only modeled by a very few techniques. Further the table shows that almost all the techniques perform control modeling, but their point of reference is different: some involve the user's perspective, some involve device's perspective and some involve the environment's/context's perspective. 7 of the surveyed techniques consider the user movements before taking any decision in context modeling. Further, the interaction modeling seems to be largely covered by the techniques; user identification and action, and based on them the operations which could be performed are major focus of interaction modeling. Leaving aside Liu et al. ((Liu et al, 2012)), no other technique seems holistic and global in its nature. They consider one or the other component of SmE with the control and model it; mostly the user. Four of the techniques are somewhat holistic as they model 3 out of 4 SmE components. There is definitely scarcity of techniques covering all areas of components modeling.

The analysis of Table 3 shows that only 4 techniques perform holistic consistency verification (between black box and white box), whereas Ahmed and Tripathi ((Ahmed and Tripathi, 2003)), though not having performed a black box modeling, still adopt a consistency verification strategy by validating the successive formalisms with the previous ones. Since most of the techniques have not performed black box modeling, therefore it seems appropriate that they (other than Ahmed and Tripathi) do not perform consistency verification. It is also observed that 7 techniques perform user behavior verification. This shows a lack of interactivity and liveliness of SmE modeling and verification practices.

Similarly, the situation is equally alarming in context verification with 7 out of 17 surveyed techniques performing this verification. It can be argued that SmE are context critical systems and demand an understanding of their physical and location-based modalities, therefore such a modeling is highly required and the research impetus is too strong to ignore in future works.

Also, some of the techniques are found to perform device interaction verification. The increasing complexity of devices and intricate nature of their behavior within the system impede this type of verification. But, based on mounting needs, it is imperative to perform this type of modeling. The only surveyed technique which has performed the complex internal device behavior verification is by Corno and Sanaullah ((Corno and Sanaullah, 2011b)).

The analysis further reveals that device interaction and control verification is performed by almost all the techniques. It is grounded on the fact that most of the surveyed techniques use control algorithms for accomplishing the system requirements, therefore it seems natural that all these techniques do perform this kind of verification. Finally, real time verification and probabilistic verification are not found so diffused in the surveyed techniques. These seem to be highly neglected areas of SmE verification and owing to their importance, it is necessary that SmE researchers also divert some effort to these areas.

The analysis of Table 4 shows abstraction is performed by less than half of the surveyed techniques, whereas others do not perform the abstraction. It can be said that those techniques which perform abstraction do so based on their large size and focus. According to the observation, it is found that some techniques are scalable, which can be enhanced by adding the other components and their aspects in more details. Further, it is found that 8 techniques are manual, 8 are semi-automatic and only 1 technique claims to be fully automatic (as it is rule-based). It can be argued there that the complex nature of SmE and correspondingly complex modeling and verification requirements hinder the automation of these techniques, as the only fully automatic techniques is also not truly automatic in its nature and is based on rules. All but two surveyed techniques use verification tools of different nature. Finally, all the surveyed techniques are tested on one or the other case study of varying nature, scope and level of complexity.

7 Conclusion

Smart Environments (SmE) are a growing field which provides implicit computation facilities in the environment so that they behave in a sophisticated desired manner. This sophistication is achieved with the interaction of users with the sensors, actuators, electrical appliances and hidden computation. The versatile nature of these components and their interaction makes the systems huge, complex and ambiguous, motivating to use the formal verification for validating the desired behavior. In this survey, the techniques which are used for the modeling of SmE and its related components, along with the conformance of reliable behavior through formal verification approaches, are considered.

We derived some parameters related to the focused area, modeling formalism, formal verification and other important factors. We analyzed the techniques on these parameters and conclude that the techniques mostly follow Statecharts for the modeling purpose. It was also observed that the black box modeling, owing to lack of its visibility, is scarcely diffused in the techniques. Nevertheless, Black box modeling assumes a fundamental role by providing generic dictionaries and naming/communication conventions which help broadly at the time of implementation. If the context and the user are also focused, then Ambient Calculus is used.

Very few techniques are observed to model and verify – at a deeper level – all basic components of SmE (user, devices, control algorithm, environment/context). The model checking technique is used for the formal verification. Some techniques also use abstractions for reducing the state-space of the model. Results of the survey show that no technique is fully automatic in true nature and covers all the dimensions (e.g. modeling of context, user, devices) of SmE. Based on this, it can also be concluded that there still remains a sizable research gap in SmE modeling and verification area. Mostly complex modeling and verification scenarios and components are given less or no attention partly due to the inherent complexity and party due to personal inclination of current researchers towards areas of their interest. This hinders in providing holistic solutions leaving behind the industry and users with their specific needs and demands. Therefore, it is deduced that more R&D effort, impartial and objective in its nature, needs to be put into the SmE modeling and verification research.

References

Acampora G, Loia V (2005) Fuzzy Control Interoperability and Scalability for Adaptive Domotic Framework. IEEE Transactions on Industrial Informatics 1(2):97–111

Ahmed T, Tripathi A (2003) Static Verification of Security Requirements in Role Based CSCW Systems. In: Symposium on Access Control Models and Technologies: Proceedings of the eighth ACM symposium on Access control models and technologies, vol 2, pp 196–203

Alur R, Henzinger T (1992) Logics and Models of Real Time: A Survey. In: Real-Time: Theory in Practice, Springer, pp 74–106

Augusto J, Hornos MJ (2013) Software Simulation and Verification to Increase the Reliability of Intelligent Environments. Advances in Engineering Software 58:18–34

Augusto J, Mccullagh P (2007) Ambient Intelligence: Concepts and Applications. Computer Science and Information Systems 4 (1):1–27

Aziz A, Klein M, Treur J (2010) An Integrative Ambient Agent Model for Unipolar Depression Relapse Prevention. Journal of Ambient Intelligence and Smart Environments 2(1):5–20

Baeten J (2005) A Brief History of Process Algebra. Theoretical Computer Science 335(2-3):131–146

Barras B, Boutin S, Cornes C, Courant J, Filliatre J, Gimenez E, Herbelin H, Huet G, Munoz C, Murthy C, et al (1997) The Coq Proof Assistant Reference Manual: Version 6.1. INRIA– Institut National De Recharche En Informatique Et Automatique

Beek M, Mazzanti F, Gnesi S (2009) CMC-UMC: A Framework for the Verification of Abstract Service-Oriented Properties. In: Proceedings of the ACM symposium on Applied Computing, New York, NY, USA, pp 2111–2117

Beek M, Fantechi A, Gnesi S, Mazzanti F (2011) A State/Event-Based Model-Checking Approach for the Analysis of Abstract System Properties. Science of Computer Programming 76:119–135

Benghazi K, Hurtado M, Hornos M, Rodríguez M, Rodríguez-Domínguez C, Pelegrina A, Rodríguez-Fórtiz M (2012) Enabling Correct Design and Formal Analysis of Ambient Assisted Living systems. Journal of Systems and Software 85(3):498–510

Bergstra J, Klop J (1984) Process Algebra for Synchronous Communication. Information and control 60(1-3):109–137

Bernardeschi C, Fantechi A, Gnesi S, Larosa S, Mongardi G, Romano D (1998) A Formal Verification Environment for Railway Signaling System Design. Formal Methods in System Design 12(2):139–161

Berthomieu B, FVernadat (2006) Time Petri Nets Analysis with TINA. In: Third International Conference on Quantitative Evaluation of Systems, IEEE, pp 123–124

Birkedal L, Debois S, Elsborg E, Hildebrandt T, Niss H (2006) Bigraphical models of context-aware systems. In: Foundations of software science and computation structures, Springer, pp 187–201

Biswas J, Mokhtari M, Dong JS, Yap P (2010) Mild Dementia Care at Home–Integrating Activity Monitoring, User Interface Plasticity and Scenario Verification. In: Aging Friendly Technology for Health and Independence, Springer, pp 160–170

Bolton ML, Bass EJ, Siminiceanu RI (2012) Generating Phenotypical Erroneous Human Behavior to Evaluate Human-Automation Interaction using Model Checking. International Journal of Human-Computer Studies 70(11):888 – 906

Bonhomme S, Campo E, Esteve D, Guennec J (2008) Methodology and Tools for the Design and Verification of a Smart Management System for Home Comfort. In: 4th International Conference on Intelligent Systems, IEEE, pp 24–2 –24–7

Booch G, Rumbaugh J, Jacobson I (1998) Unified Modeling Language User Guide, The. Addison Wesley.

Bosse T, Jonker CM, Meij L, Treur J (2007) A Language and Environment for Analysis of Dynamics by Simulation. International Journal on Artificial Intelligence Tools 16(03):435–464

Bosse T, Jonker C, Meij L, Sharpanskykh A, Treur J (2009) Specification and Verification of Dynamics in Agent Models. International Journal of Cooperative Information Systems 18(01):167–193

Boytsov A, Zaslavsky A (2013) Formal Verification of Context and Situation Models in Pervasive Computing. Pervasive and Mobile Computing 9(1):98 – 117

Brock B, Kaufmann M, Moore J (1996) ACL2 Theorems about Commercial Microprocessors. In: Formal Methods in Computer-Aided Design, Springer, pp 275–293

Brookes S (1983) On the Relationship of CCS and CSP. Automata, Languages and Programming pp 83–96

Byg J, Jørgensen K, Srba J (2009) TAPAAL: Editor, Simulator and Verifier of Timed-arc Petri Nets. Automated Technology for Verification and Analysis pp 84–89

Cardelli L, Gordon A (1998) Mobile Ambients. In: Foundations of Software Science and Computation Structures, Springer, pp 140–155

Charatonik W, Talbot JM (2001) The Decidability of Model Checking Mobile Ambients. In: Fribourg L (ed) Computer Science Logic, Lecture Notes in Computer Science, vol 2142, Springer Berlin / Heidelberg, pp 339–354

Chen C, Helal S (2012) System-wide support for safety in pervasive spaces. Journal of Ambient Intelligence and Humanized Computing 3(2):113–123

Cimatti A, Clarke E, Giunchiglia E, Giunchiglia F, Pistore M, Roveri M, Sebastiani R, Tacchella A (2002) Nusmv 2: An Opensource Tool for Symbolic Model Checking. In: Computer Aided Verification, Springer, pp 241–268

Clarke E, Wing J (1996) Formal Methods: State of the Art and Future Directions. ACM Computing Surveys 28(4):626–643

Clarke E, Grumberg O, Long D (1994) Verification Tools for Finite-State Concurrent Systems. A Decade of Concurrency Reflections and Perspectives pp 124–175

Clarke EM, Emerson EA, Sistla AP (1986) Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems 8:2:244–263

Cook D (2009) Multi-Agent Smart Environments. Journal of Ambient Intelligence and Smart Environments 1(1):51–55

Cook D, Youngblood M, Das S (2006) A Multi-Agent Approach to Controlling a Smart Environment. Designing smart homes pp 165–182

Corno F, Sanaullah M (2011a) Design Time Methodology for the Formal Verification of Intelligent Domotic Environments. In: Ambient Intelligence - Software and Applications, Advances in Intelligent and Soft Computing, vol 92, Springer Berlin / Heidelberg, pp 9–16

Corno F, Sanaullah M (2011b) Formal verification of device state chart models. In: 7th International Conference on Intelligent Environments, IEEE, pp 66–73

Corno F, Sanaullah M (2013) Modeling and Formal Verification of Smart Environments. Security and Communication Networks pp n/a–n/a, DOI 10.1002/sec.794, URL `http://dx.doi.org/10.1002/sec.794`

Coronato A, Pietro G (2010a) Formal Specification of Wireless and Pervasive Healthcare Applications. ACM Transactions on Embedded Computing Systems 10(1):12

Coronato A, Pietro GD (2010b) Formal Design of Ambient Intelligence Applications. Computer 43(12):60 –68

Coronato A, Pietro GD (2011) Formal Specification and Verification of Ubiquitous and Pervasive Systems. ACM Transactions on Autonomous and Adaptive Systems 6(1):9:1–9:6

Crocker D (2003) Perfect Developer: A Tool for Object-Oriented Formal Specification and Refinement. Tools exhibition notes at formal methods Europe

Crocker D, Carlton J (2007) Verification of C Programs using Automated Reasoning. In: Fifth International Conference on Software Engineering and Formal Methods, IEEE, pp 7–14

Diane C, Sajal D (2004) Smart Environments: Technology, Protocols and Applications. Wiley-Interscience

Fantechi A, Gnesi S, Lapadula A, Mazzanti F, Pugliese R, Tiezzi F (2008) A model checking Approach for Verifying COWS Specifications. Fundamental Approaches to Software Engineering pp 230–245

Feng L, Apers P, Jonker W (2004) Towards Context-Aware Data Management for Ambient Intelligence. In: Database and Expert Systems Applications, Springer, pp 422–431

Fensel D (2001) Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, New York, NY, USA

Forejt V, Kwiatkowska M, Norman G, Parker D (2011) Automated Verification Techniques for Probabilistic Systems. In: Formal Methods for Eternal Networked Software Systems, Springer, pp 53–113

Gallardo M, Merino P, Pimentel E (2002) Debugging UML Designs with Model Checking. Journal of Object Technology 1(2):101–117

Garavel H, Lang F, Mateescu R, et al (2001) An Overview of CADP 2001. Research Report RT-0254, INRIA, URL `http://hal.inria.fr/inria-00069920`

Gardey G, Lime D, Magnin M, Roux O (2005) Romeo: A tool for Analyzing Time Petri Nets. In: Computer Aided Verification, Springer, pp 261–272

Gnesi S, Mazzanti F (2004) On the Fly Model Checking of Communicating UML State Machines. In: Second ACIS International Conference on Software Engineering Research, Management and Applications, pp 331–338

Gnesi S, Latella D, Massink M (1999) Model Checking UML Statechart Diagrams Using JACK. In: Proceedings of 4th IEEE International Symposium on High-Assurance Systems Engineering, IEEE, pp 46–55

Hagras H, Callaghan V, Colley M, Clarke G, Pounds-Cornish A, Duman H (2004) Creating an Ambient-Intelligence Environment Using Embedded Agents. Intelligent Systems 19(6):12–20

Harel D (1987) Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming 8(3):231 – 274

Harrison J (1996) HOL Light: A Tutorial Introduction. In: Formal Methods in Computer-Aided Design, Springer, pp 265–269

Hennessy M, Milner R (1985) Algebraic Laws for Nondeterminism and Concurrency. Journal of the ACM 32(1):137–161

Hoare C (1978) Communicating Sequential Processes. Communications of the ACM 21(8):666–677

Holzmann G (1997) The Model Checker SPIN. IEEE Transactions on Software Engineering 23(5):279–295

Hoogendoorn M, Klein M, Memon ZA, Treur J (2009) Formal Verification of an Agent-Based Support System for Medicine Intake. In: Fred A, Filipe J, Gamboa H (eds) Biomedical Engineering Systems and Technologies, Communications in Computer and Information Science, vol 25, Springer Berlin Heidelberg, pp 453–466

Hoogendoorn M, Klein MC, Memon ZA, Treur J (2013) Formal Specification and Analysis of Intelligent Agents for Model-Based Medicine Usage Management. Computers in Biology and Medicine 43(5):444 – 457

Hopcroft J, Motwani R, Ullman J (1979) Introduction to Automata Theory, Languages, and Computation, vol 2. Addison-wesley Reading, MA

IBM (2005) Commonsense Reasoning with the Discrete Event Calculus Reasoner. URL `http://decreasoner.sourceforge.net/`

Ishikawa F, Suleiman B, Yamamoto K, Honiden S (2009) Physical Interaction in Pervasive Computing: Formal Modeling, Analysis and Verification. In: Proceedings of the international conference on Pervasive services, ACM, pp 133–140

Jansen D, Hermanns H, Katoen J (2002) A Probabilistic Extension of UML Statecharts. In: Formal Techniques in Real-Time and Fault-Tolerant Systems, Springer, pp 355–374

Kofod-Petersen A, Aamodt A (2006) Contextualised Ambient Intelligence Through Case-Based Reasoning. Advances in Case-Based Reasoning pp 211–225

Kwiatkowska M, Norman G, Parker D (2002) PRISM: Probabilistic Symbolic Model Checker. Computer Performance Evaluation: Modelling Techniques and Tools pp 113–140

Larsen K, Pettersson P, Yi W (1997) UPPAAL in a Nutshell. International Journal on Software Tools for Technology Transfer 1(1):134–152

Latella D, Majzik I, Massink M (1999a) Automatic Verification of a Behavioural Subset of UML Statechart Diagrams Using the SPIN Model-Checker. Formal Aspects of Computing 11(6):637–664

Latella D, Majzik I, Massink M (1999b) Towards a Formal Operational Semantics of UML Statechart Diagrams. In: Proceedings of the IFIP TC6/WG6, vol 99, pp 15–18

Leelaprute P, Nakamura M, Tsuchiya T, Matsumoto K, Kikuno T (2005) Describing and Verifying Integrated Services of Home Network Systems. In: 12th Asia-Pacific Software Engineering Conference, p 10

Li P, Regehr J (2010) T-check: Bug Finding for Sensor Networks. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, ACM, pp 174–185

Lilius J, Paltor I (1999) vUML: A Tool for Verifying UML Models. In: 14th IEEE International Conference on Automated Software Engineering, IEEE, pp 255–258

Liu Y, Zhang X, Dong J, Liu Y, Sun J, Biswas J, Mokhtari M (2012) Formal Analysis of Pervasive Computing Systems. In: 17th International Conference on Engineering of Complex Computer Systems, IEEE, pp 169–178

Madl G, Abdelwahed S, Schmidt D (2006) Verifying Distributed Real-Time Properties of Embedded Systems via Graph Transformations and Model Check-

ing. Real-Time Systems 33(1):77–100

Manna Z, Pnueli A (1992) The Temporal Logic of Reactive and Concurrent Systems. Springer-Verlag New York, Inc., New York, NY, USA

Masci P, Furniss D, Curzon P, Harrison MD, Blandford A (2012) Supporting field investigators with pvs: a case study in the healthcare domain. In: Software Engineering for Resilient Systems, Springer, pp 150–164

Masci P, Curzon P, Harrison MD, Ayoub A, Lee I, Thimbleby H (2013a) Verification of interactive software for medical devices: Pca infusion pumps and fda regulation as an example. EICS2013 ACM Digital Library

Masci P, Zhang Y, Curzon P, Harrison MD, Jones P, Thimbleby H (2013b) Verification of software for medical device user interfaces in PVS. Submitted paper, URL `http://www.chi-med.ac.uk/researchers/bibdetail.php?docID=656`

Mascolo C, Ghica D, Ryan M, Lupu E (2009) UbiVal: Fundamental Approaches to Validation of Ubiquitous Computing Applications and Infrastructures. Research Proposed EP/D076625/2, EPSRC, URL `https://www.comp.nus.edu.sg/~david/Research/ubival.pdf`

McMillan K (1992) Symbolic Model Checking: An Approach to the State Explosion Problem. Tech. rep., DTIC Document

Mikk E, Lakhnechi Y, Siegel M (1997) Hierarchical Automata as Model for Statecharts. Advances in Computing Science pp 181–196

Mikk E, Lakhnech Y, Siegel M, Holzmann G (1998) Implementing Statecharts in PROMELA/SPIN, booktitle = Proceedings in 2nd Workshop on Industrial Strength Formal Specification Techniques. IEEE, pp 90 –101

Milner R, Parrow J, Walker D (1992) A Calculus of Mobile processes, I. Information and computation 100(1):1–40

Mottola L, Voigt T, Österlind F, Eriksson J, Baresi L, Ghezzi C (2010) Anquiro: Enabling Efficient Static Verification of Sensor Network Software. In: Proceedings of the ICSE Workshop on Software Engineering for Sensor Network Applications, ACM, pp 32–37

Nicola RD (1995) Three Logics for Branching Bisimulation. Journal of the Association for Computing Machinery 42:2:458–487

Nicola RD, Vaandrager F (1990) Action versus state based logics for transition systems. Semantics of Systems of Concurrent Processes, Lecture Notes in Computer Science 469:407–419

Nielsen M, Plotkin G, Winskel G (1981) Petri Nets, Event Structures and Domains, Part I. Theoretical Computer Science 13(1):85–108

Owre S, Rajan S, Rushby JM, Shankar N, Srivas M (1996) Pvs: Combining specification, proof checking, and model checking. In: Computer Aided Verification, Springer, pp 411–414

Padovitz A, Loke SW, Zaslavsky A (2008) Multiple-Agent Perspectives in Reasoning about Situations for Context-Aware Pervasive Computing Systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 38(4):729–742

Paulson LC (1989) The Foundation of a Generic Theorem Prover. Journal of Automated Reasoning 5(3):363–397

Pedrycz W (2010) Human Centricity in Computing with Fuzzy Sets: an Interpretability Quest for Higher Order Granular Constructs. Journal of Ambient Intelligence and Humanized Computing 1(1):65–74

Ranganathan A, Campbell R (2008) Provably Correct Pervasive Computing Environments. In: Sixth Annual International Conference on Pervasive Computing and Communications, IEEE, pp 160–169

Reynolds M (2005) An Axiomatization of PCTL*. Information and Computation 201(1):72–119

Roman GC, Julien C, Payton J (2007) Modeling Adaptive Behaviors in Context UNITY. Theoretical Computer Science 376(3):185–204

Sadri F (2011) Ambient Intelligence: A Survey. ACM Computing Surveys 43(4):36:1–36:66

Schafer T, Knapp A, Merz S (2001) Model Checking UML State Machines and Collaborations. Electronic Notes in Theoretical Computer Science 55(3):357–369

Shanahan M (1999) The Event Calculus Explained. In: Artificial intelligence today, Springer, pp 409–430

Sharpanskykh A, Treur J (2012) An Ambient Agent Architecture Exploiting Automated Cognitive Analysis. Journal of Ambient Intelligence and Humanized Computing 3(3):219–237

Siewe F, Zedan H, Cau A (2011) The Calculus of Context-Aware Ambients. Journal of Computer and System Sciences 77(4):597–620

Stankovski V, Trnkoczy J (2006) Application of Decision Trees to Smart Homes. Designing Smart Homes pp 132–145

Stevens P, Stirling C (1998) Practical Model-Checking Using Games. Tools and Algorithms for the Construction and Analysis of Systems pp 85–101

Sun J, Liu Y, Dong J, Pang J (2009) PAT: Towards Flexible Verification under Fairness. In: Bouajjani A, Maler O (eds) Computer Aided Verification, Lecture Notes in Computer Science, vol 5643, Springer Berlin Heidelberg, pp 709–714

Wang F (2004) Formal Verification of Timed Systems: A Survey and Perspective. Proceedings of the IEEE 92(8):1283–1305

Weiser M (1991) The Computer for the 21st Century. Scientific American 265(3):94–104

Woodcock J, Larsen P, Bicarregui J, Fitzgerald J (2009) Formal Methods: Practice and Experience. ACM Computing Surveys 41(4):19

Ye J, Dobson S, McKeever S (2012) Situation Identification Techniques in Pervasive Computing: A Review. Pervasive and Mobile Computing 8(1):36–66

Zadeh LA (1965) Fuzzy Sets. Information and control 8(3):338–353