Formal Modelling and Verification of Pervasive Computing Systems

Yan Liu

National University of Singapore
yanliu@comp.nus.edu.sg

Abstract

Pervasive computing (PvC) systems are emerging as promising solutions to many practical problems, e.g., elderly care in home. However, such systems have long been developed without sufficient verification. Formal methods, eps. model checking are sound techniques for complex system analysis regarding correctness and reliability requirements. In this work, a formal modelling framework is proposed to model the general the system design (e.g., concurrent communications) and the critical environment inputs (e.g., human behaviours). Correctness requirements are specified in formal logics which are automatically verifiable against a system model. Furthermore, Markov Decision Processes (MDPs) are adopted for modelling probabilistic behaviours of PvC systems. Three problems are analysed which are overall reliability prediction based on component reliabilities, reliability distribution w.r.t., how reliable should the component be to reach an overall reliability requirement and sensitivity analysis w.r.t., how does a component reliability affect the overall reliability. Finally, the usefulness of our approaches are demonstrated on a smart healthcare system with unexpected bugs and system flaws exposed.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases System Analysis, Formal Modelling and Verification, Reliability Analysis

Digital Object Identifier 10.4230/OASIcs.FSFMA.2013.61

1 Introduction

Many problems arise with the proliferation of ageing population in all industrialised societies, e.g., creating enormous costs for elders' intensive care. The context-aware and self-adaptive PvC [11] system enables their independent living with little supervision [7]. Therefore PvC systems are safety critical and should be verified before deployment. However, traditional techniques such as simulation and testing are expensive and not complete. Formal methods instead, especially model checking [4] techniques are promising solutions for their expressive system modelling and exhaustive verification. Thus, we propose to apply these techniques to formally analyse PvC systems.

Motivation. PvC systems are inherently complex making it a challenging task to perform system analysis. They are usually composed of a physical layer with sensors to monitor the environment changes; a middleware layer to manage and reason about the sensed contexts with predefined rules; an application layer to make adaptations, as shown in Fig. 1a. Failures happen with various reasons like a wrong reminder could be caused by a sensor failure or an incorrect rule [5]. In practice, such faults could only be exposed during deployment as it is impossible to capture all scenarios at development phase. Thus, there is a need of a systematic and complete analysis approach. Furthermore, a PvC system is probabilistic due to limited reliability of its components [8]. It is essential to manage the system reliability at

1st French Singaporean Workshop on Formal Methods and Applications 2013 (FSFMA'13).
Editors: Christine Choppy and Jun Sun; pp. 61–67

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

an acceptable rate. Besides, the system behaviours are nondeterministic for unpredictable user activities. Thus, we choose MDPs-based verification technique for reliability analysis.

Our **contributions** are two-fold. Firstly, a formal modelling framework is proposed with modelling patterns for common features of PvC systems, e.g., compositional architecture and concurrent interactions. Critical requirements of stakeholders are specified as desirable properties (safety and liveness) and testing purposes (conflict cases). Case study on a smart healthcare system revealed multiple bugs such as conflict reminders. Secondly, reliability models of PvC systems are constructed using MDPs upon which three general problems are investigated: 1) "What is the overall system reliability given the component reliabilities?"' refers to *reliability prediction*. 2) "What is the reliability required on components for an expected reliability on overall system?" refers to *reliability distribution*. 3) "What is the most critical part contributes to the system reliability?" refers to *sensitivity analysis*. Experiments on AMUPADH system shows that the overall system reliability is below 50%.

2 A running example – AMUPADH

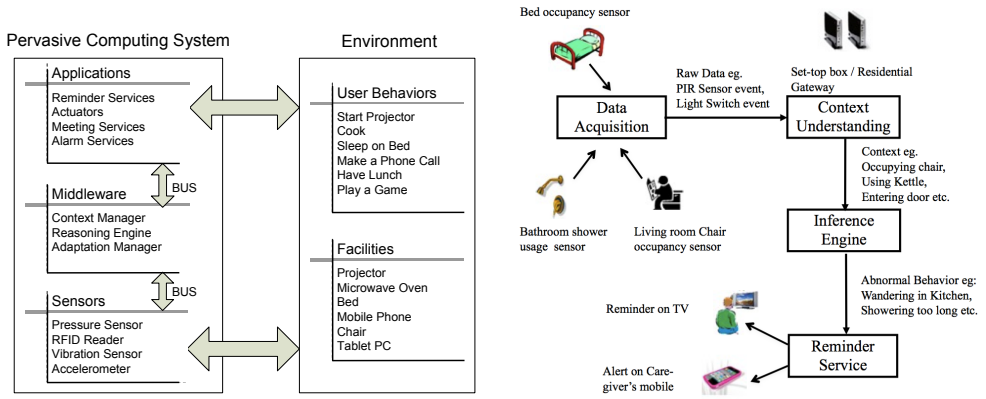
AMUPADH [2] is a smart healthcare system providing assistance to elderly people who have difficulties in remembering activities of daily living (ADLs). It is deployed in a nursing home, PeaceHeaven[1] for a six-month real life trial. The workflow in Fig. 1b consists of:

- **Step 1: Data Acquisition.** Multiple sensors are deployed to monitor the environment such as when someone turns on the shower tap, the shake sensor is triggered and a *Unstationary* signal is sent to the system.
- **Step 2: Context Processing and Reasoning.** Sensor signals are interpreted to low-level contexts like "Tap turned on" in the inference engine, Drools[2]. By evaluating predefined reasoning rules (written in first order logic), high-level contexts such as "Showering too long" are generated.
- **Step 3: Reminder Service Rendering.** If an abnormal activity is recognised, the system will render a reminder service to help the elder. For example, a bluetooth speaker will play a voice reminder upon a error message. A number of devices like a TV or iPad are used to prompt reminders.

3 Correctness Analysis of PvC Systems

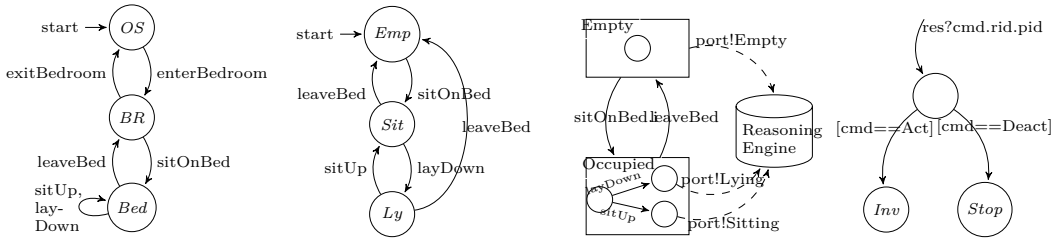
A Formal Modelling Framework. According to the general structure shown in Fig. 1a, we propose to model the system design and environment input separately. *Modeling Environments:* PvC systems are user centric, thus modelling the user behaviours is essential. However it is difficult because user behaviours are unpredictable. As suggested by domain experts, such systems usually target at a determined group of activities whose sequences remain unpredictable. In concurrent modelling languages, nondeterministic choices can be used to enumerate the sequences, an example is illustrated in Fig. 2a. At a location, each possible move of the user is modelled as a choice. An unrealistic scenario may arise that the model allows an activity to be performed repeatedly e.g., the user sits on the bed again and again without standing up. To eliminate such cases, we need a constraint model e.g., a bed model (Fig. 2b, no more sitting is allowed once the bed is occupied). *Modelling System*

[1] Located at 9 Upper Changi Road North, Singapore, 507706. Tel: +65-65465678.
[2] Drools Expert: http://www.jboss.org/drools/drools-expert.html



(a) Common Architectures **(b)** An example PvC System, AMUPADH

Figure 1 Introduction of Pervasive Computing (PvC) Systems.



(a) User Behaviours **(b)** Bed Behavior **(c)** Bed Pressure Sensor **(d)** Service Rendering

Figure 2 Modelling a PvC System.

Design: In PvC systems, communication and cooperation of components are most critical. The sensing behaviour of sensors is a concurrent happening with the detectable event in the environment. Thus, sensor models need to be paralleled with environment model such that they are synchronised on common events. As shown in Fig. 2a, 2b and 2c, a sitOnBed event will trigger three models to progress simultaneously. Additionally, the synchronised channel models the message sending from sensors to reasoning engine in negligible time [6]. In the middleware, context managing and reasoning are performed. They are modelled as data operations on global context variables and conditional statements respectively. At the application layer, services are rendered upon decoding of the messages. As shown in Fig. 2d, the reminding system is modelled using channel communications, shared variables and guarded processes. *Compose A Complete Model:* Finally, component models should be composed using sequential, interleave or parallel patterns according to their relations.

Formal Specification of Critical Requirements. Critical requirements from the stakeholders are identified as desirable properties and testing purposes. *Deadlock freeness* (DF) and *Guaranteed reminder services* (GR) properties are desirable which respectively requires the system has no dead state where no more actions can be performed and services should be provided at the right moment for the right user. For instance, a reminder should be sent to a patient whenever he is wandering somewhere is formalised as "$\Box$(Wandering $\rightarrow$ $\Diamond$RemindLeave)". It is also helpful to test the common problems i.e.,

Table 1 Results of Correctness Analysis Experiment.

Property	Result	# States	# Transitions	Time(s)	Bug?
$DF_{Complete}$	–	–	–	OOM	No
$DF_{Bedroom}$	True	1.43M	2.04M	815	No
$DF_{Washroom}$	True	10.8M	15.8M	7045	No
$GR_{UsingWrongBed(UWB)}$	True	1.60M	2.43M	1945	No
$GR_{TapNotOff(TNO)}$	False	0.07M	0.131M	39	Yes
$GR_{WanderInWashroom(WiW)}$	False	2.19M	4.53M	12414	Yes
$GR_{ShowerNoSoap(SNS)}$	False	0.832M	1.66M	729	Yes
$GR_{ShowerTooLong(STL)}$	False	4314	5150	1.6	Yes
$GR_{SitBedTooLong(SBTL)}$	True	1.58M	2.38M	1913	Yes
Inconsistency	True	572	745	0.3	Yes
Conflict Reminder	True	2446	3036	1.11	Yes
False Alarm	True	0.01M	0.02M	6.1	Yes

system inconsistency and *conflicting/ false services*. Both of them can be specified as reachability properties that are verified by checking if there is a state where system knowledge contradicts with actual environment and a state where two conflict services are invoked/ where a service is rendered for a wrong user respectively.

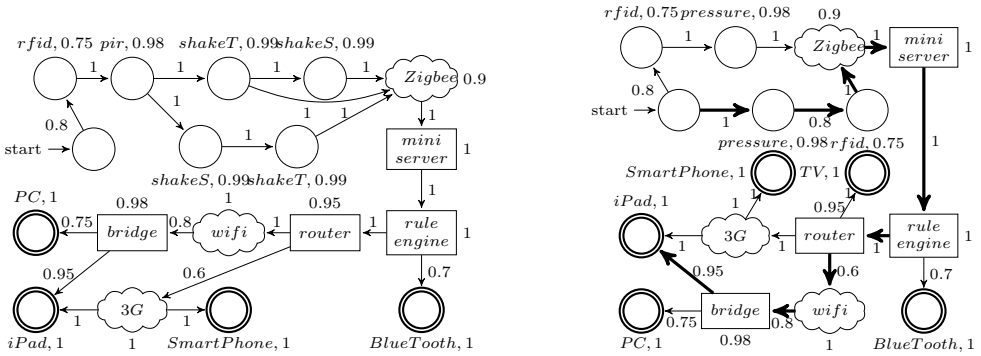
Case Study on AMUPADH System. In the experiment, we implement the modelling framework in CSP# language [9] and run verification by PAT model checker [10]. In Table 1[3], violation of guaranteed reminder(GR) properties reveals a design flaw i.e., inefficient update of the patient's location. An inconsistent state is found i.e., the patient's location context variable remains to be inside washroom even after he has left. The case study shows the usefulness of our approach in analysing PvC system with the counterexamples help in efficient system debugging. It is also observed that the state space reaches the limits of the model checker. Thus, a future direction is to explore state space reduction techniques.

4 Reliability Analysis using MDPs

System Modelling in MDPs. MDPs allow us to model both probabilistic and nondeterministic behaviours. In general, nondeterministic choice is adopted when no definitive information is given for resolving the choice. In Fig. 3a, it is used to model transitions between sensors because of the randomness of user behaviours. *States* are abstract nodes of sensors, software components and network devices associated with a target scenario while double circled nodes are goals. In PvC systems, there are two types of *transitions* which are the happen-before relation among sensors and message passing directions among the others. *Labels* denote the reliability values of a node or a transition which are usually provided by system engineers estimated from exemplar system runs.

Reliability Analysis Approaches. It consists of three parts, Fig. 4 (a) shows *reliability prediction* which calculates the reachable probability, $Pr(M, s)$ from an initial state to a goal state s on an MDP model M. A reliability range i.e., max. and min. reachability is produced since multiple reachable paths (aka. schedulers) are created due to nondeterminism. *Reliability distribution* calculation (Fig. 4 (b)) takes two inputs: (1) a reliability requirement R on the overall system; (2) a parameterised MDP model M with weights $w_i x$ (denotes the

[3] The test bed is a PC with Intel Xeon CPU at 2.13GHz and 32GB RAM. OOM stands for out of memory.



(a) Tap-Not-Off (TNO) Model **(b)** Using-Wrong-Bed (UWB) Model

Figure 3 MDP models for Scenario TNO and UWB.

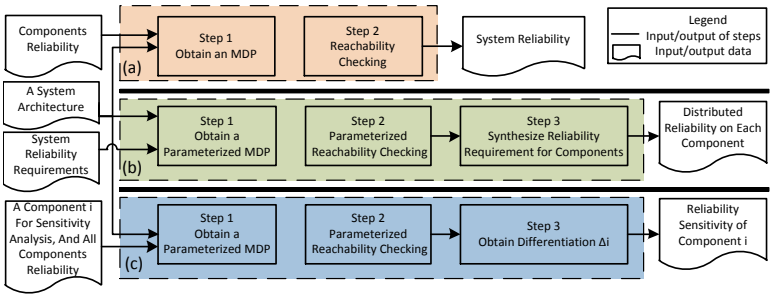


Figure 4 Workflow: (a) reliability prediction; (b) reliability distribution; (c) sensitivity analysis.

reliability of component x has a weight w_i). Given a scheduler δ, we can obtain the system reliability (i.e., $Pr(M, s)$) as a polynomial function of x only. Then the Newton's method is used to calculate the lower bounds on x for finitely many schedulers [1] among which the maximum value gives us the minimum requirement on component reliability. *Sensitivity analysis* is shown in Fig. 4 (c). The sensitivity s_i of the i^{th} component's reliability R_i is defined as a partial derivation (denoted by f w.r.t. R_i) of system reliability R, denoted as $\Delta_i = \frac{\delta f(R_1, R_2, ...R_i, ...R_n)}{\delta R_i}$. In this work, we investigate one component each time that the formula is then reduced to $\Delta_i = \frac{\delta V(init)}{\delta R_i}$ ($V(init)$ is obtained via reliability distribution).

Case Study on AMUPADH system. Six scenarios that need reminders are modelled similarly with Fig. 3. These MDPs models are then analysed using the model checker RaPid [3]. Reliability of these reminders ranges from 0.25 to 0.4 (Table 2a) which is quite low. It is because the RFID readers depend on the wearable tags that the patients throw away from time to time. Furthermore, Table 2b shows the network nodes need a reliability 0.913 for all the scenarios to achieve a system reliability of 0.4. For the requirement of 0.5, it is impossible to distribute. It's because the system cannot differentiate who is sitting on the bed that the SBTL reminder is sent to the wrong person half of the time. As for *sensitivity analysis*, we demonstrate one scheduler in UWB scenario (highlighted path in Fig. 3b). Fig. 5a suggests improvement on RFID and Wi-Fi nodes gains higher system reliability than Zigbee node. Furthermore, when their reliability reaches 0.7, improving Wi-Fi nodes is more efficient than others (Fig. 5b). These experiments give a good estimation and useful guidance on improving the system reliability, especially in budget concerned systems.

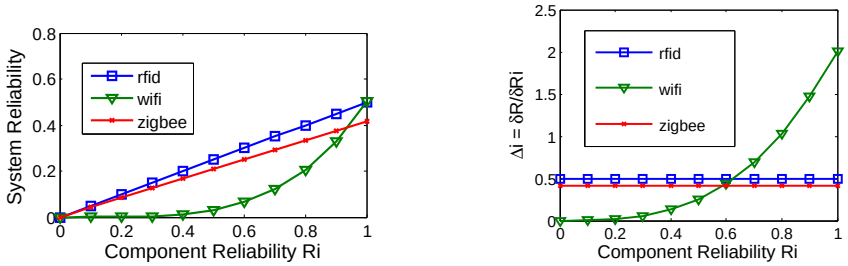
Table 2 Experiments of Reliability Prediction and Distribution Analysis.

(a) Reliability Prediction

Rel.	UWB	SBTL	SNS	STL	TNO	WiW
Scedulers	32	24	32	16	64	16
Max	0.374	0.419	0.367	0.371	0.371	0.371
Min	0.296	0.246	0.290	0.292	0.290	0.292
Time	<1 ms					

(b) Reliability Distribution

Req.	Nodes	UWB	SBTL	SNS	STL	TNO	WiW
0.4	Network	0.854	0.904	0.913	0.911	0.911	0.911
	Sensor	0.886	0.938	0.941	0.923	0.923	0.923
0.5	Network	0.914	-	0.965	0.963	0.963	0.963
	Sensor	0.996	-	0.995	0.994	0.994	0.994
Time(s)		3.45	2.68	3.86	1.87	11.00	2.35



(a) Distribution **(b)** Sensitivity

Figure 5 Using Wrong Bed (UWB)- Sensitivity Analysis on Nodes.

5 Conclusion

In this paper, we demonstrate the approaches of formally analysing PvC systems using model checking techniques, i.e., a formal modelling framework is proposed for correctness analysis; an MDPs-based approach for reliability analysis w.r.t., reliability prediction, distribution and sensitivity analysis. In future, we intend to develop algorithms to alleviate the state space explosion problem.

Acknowledgements. Supervisor: Dr. Jin Song Dong.

References

1 C. Baier and J. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
2 J. Biswas, M. Mokhtari, J. S. Dong, and P. Yap. Mild dementia care at home - integrating activity monitoring, user interface plasticity and scenario verification. In *ICOST*, pages 160–170, 2010.
3 L. Gui, J. Sun, Y. Liu, Y. Si, J. S. Dong, and X. Wang. Combining model checking and testing with an application to reliability prediction and distribution. In *ISSTA*, Accepted 2013.
4 E. M. Clarke Jr., O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.
5 V. Lee, Y. Liu, X. Zhang, C. Phua, K. Sim, J. Zhu, J. Biswas, J. S. Dong, and M. Mokhtari. Acarp: Auto correct activity recognition rules using process analysis toolkit (pat). In *ICOST*, pages 182–189. 2012.

6 Y. Liu, X. Zhang, J. S. Dong, Y. Liu, J. Sun, J. Biswas, and M. Mokhtari. Formal analysis of pervasive computing systems. In *ICECCS*, pages 169 –178, 2012.

7 J. Nehmer, M. Becker, A. Karshmer, and R. Lamm. Living assistance systems: an ambient intelligence approach. In *ICSE*, pages 43–50, 2006.

8 M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Pers. Commun.*, 8:10–17, 2001.

9 J. Sun, Y. Liu, J. S. Dong, and C. Q. Chen. Integrating specification and programs for system modeling and verification. *TASE*, pages 127–135, 2009.

10 J. Sun, Y. Liu, J. S. Dong, and J. Pang. Pat: Towards flexible verification under fairness. In *CAV*, pages 709–714, 2009.

11 M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, 1991.