DOI 10.1007/s00165-013-0277-4
BCS © 2013
Formal Aspects of Computing

Formal Aspects of Computing

Formal verification of a pervasive messaging system

Savas Konur[1], Michael Fisher[1], Simon Dobson[2] and Stephen Knox[3]

[1] Department of Computer Science, University of Liverpool, Liverpool, UK
[2] School of Computer Science, University of St Andrews, St Andrews, UK
[3] School of Computer Science and Informatics, UCD Dublin, Dublin, Ireland

Abstract. As ubiquitous computing becomes a reality, its applications are increasingly being used in business-critical, mission-critical and even in safety-critical, areas. Such systems must demonstrate an assured level of correctness. One approach to the exhaustive analysis of the behaviour of systems is *formal verification*, whereby each important requirement is logically assessed against all possible system behaviours. While formal verification is often used in safety analysis, it has rarely been used in the analysis of deployed pervasive applications. Without such formality it is difficult to establish that the system will exhibit the correct behaviours in response to its inputs and environment. In this paper, we show how model-checking techniques can be applied to analyse the probabilistic behaviour of pervasive systems. As a case study we apply this technique to an existing pervasive message-forwarding system, *Scatterbox*. Scatterbox incorporates many typical characteristics of pervasive systems, such as dependence on sensor reliability and dependence on context. We assess the dynamic temporal behaviour of the system, including the analysis of probabilistic elements, allowing us to verify formal requirements even in the presence of uncertainty in sensors. We also draw some tentative conclusions concerning the use of formal verification for pervasive computing in general.

Keywords: Formal specification, Pervasive systems, Probabilistic verification

1. Introduction

Pervasive or *ubiquitous* computing covers a very broad class of systems that are mobile, dynamic, and can sense their physical and digital environments and adapt their behaviour accordingly. Such systems are often autonomous, distributed and concurrent, and involve humans, intelligent software agents and computational artifacts in the system together [DSNH10]. Their requirements are becoming steadily more complex, involving a great diversity in types of services [WPBF02], such as multimedia, display, communication and automation services, as well as multiple co-operating (or not) users.

Developing software for pervasive scenarios is difficult; developing software whose behaviour can be *guaranteed* is even more challenging. Even failures in systems that are ephemeral to users' experiences erode trust and acceptance; however, as pervasive systems become more mission- and even safety-critical, the need for hard guarantees on adaptive behaviours increases dramatically. It is therefore timely to study the application of formal verification to pervasive systems development, in particular to guarantee that a system demonstrates the correct behavioural adaptations in all circumstances.

Correspondence and offprint requests to: S. Konur, E-mail: konur@liverpool.ac.uk

Published online: 28 March 2013

While formal verification is increasingly used in the analysis of safety-critical systems, it has rarely been used in the analysis of pervasive applications (see Sect. 8). Such verification presents challenges not regularly encountered in more traditional systems. Critically, pervasive systems are presented with large degrees of *uncertainty* in terms of their sensor inputs, their assumptions about user behaviour, and their use of context. This means that, while there may be a notionally "correct" behaviour in each circumstance—whose selection we can verify—there remains a *probabilistic* element to the system's overall behaviour deriving from the unreliability of the system's inputs and inferences. Since it is well-known that the uncertainties of these aspects cannot be engineered away completely, we must therefore quantify the reliability we can expect from the system given these uncertainties: in given circumstances, what is the likelihood that the system will exhibit a different behaviour to that expected?

In this paper, we study the formal verification of the probabilistic behaviour of pervasive systems. We show how probabilistic model-checking can be applied to analyse the probabilistic behaviour of a pervasive system: specifically, a pervasive message-forwarding system called *Scatterbox* [KSC+08], which incorporates many typical characteristics of sophisticated pervasive systems, including sensors, uncertainty, context-handling, user interaction and communication [DN05] (see Sects. 3 and 4). By verifying not only the dynamic and temporal aspects of the system (formally modelled in Sect. 5), but also its probabilistic behaviours (see Sect. 6), we are able to analyse formal requirements even in the presence of uncertainty in sensor accuracy, user behaviour, and context identification. This allows us not only to say whether the system is "correct" or not, but to also quantify how reliable we expect it to be given its use of unreliable components (Sect. 7).

The formal characterisation of pervasive systems serves two distinct purposes. Classical notions of exhaustive analysis of correctness are appropriate for the commissioning (and even certification where appropriate) of pervasive and other sensor-driven systems. Perhaps more importantly, however, probabilistic model-checking approaches can help designers uncover undesirable properties of a system that, due to the uncertain nature of the world, are *impossible to avoid* by the system as designed. This can then be used as a tool for re-design, to reduce the potential for such occurrences to an acceptable level.

The contributions of this work are threefold. Firstly, we present what we believe to be the first analysis of the probabilistic behaviour of a viable pervasive system, including a quantitative understanding of the effects of uncertain sensor input and inference. This analysis is realistic, both in the sense of being applied to an implemented and deployed system, and in the sense of dealing quantitatively with the engineering limitations on sensor precision and accuracy that all such systems will continue to face. Secondly (and conversely), we demonstrate that pervasive systems lie within the remit of existing formal methods by applying a known method to a previously under-addressed problem domain. In doing so, we demonstrate the broad utility of probabilistic formal methods. Thirdly, we highlight areas in which formal approaches to pervasive systems analysis and design may be improved with reference to a realistic target system analysed with a realistic method. These observations have applications to sensor networks and other embedded sensor-driven systems.

2. Formal verification

Formal verification describes a family of techniques for exhaustively analysing the logical correctness of a system. The essence of formal verification is to analyze a logical requirement (typically stated in temporal logic) against all possible behaviours of the system in question.

In the case of *deductive* verification, this involves *proving* that a formula capturing all the possible system behaviours always implies a formula describing the logical requirement. Unfortunately, this deductive process is often difficult and might well require human intervention. The more popular approach at present is that of *algorithmic* verification. Here, a structure (such as a finite-state automaton) describing all possible system executions is exhaustively checked against the required logical properties. Specifically, the core algorithmic verification technique of *model-checking* [CGP99] is widely used. Model-checking has come to prominence in recent years as it provides fast, automated, and relatively easy to use verification techniques. These, in turn, are embodied within tools that are widely available, such as NuSMV [CCGR99], SPIN [Hol03], UPPAAL [BLL+95], and Kronos [BDM+98].

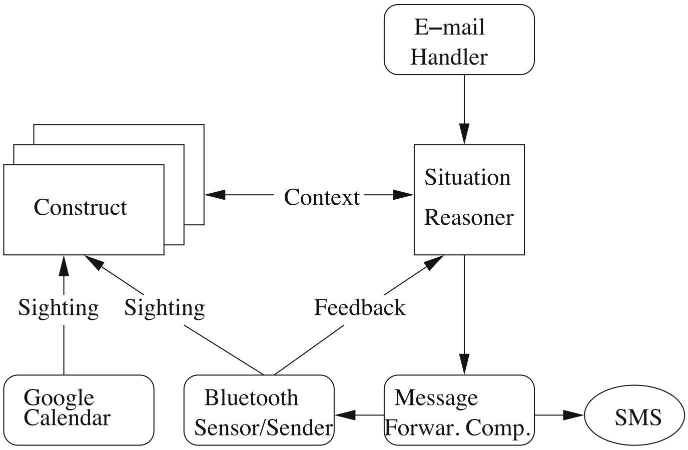

Fig. 1. Scatterbox system architecture

Such tools are based on the analysis of essentially *temporal* behaviours, for example properties such as

$\mathsf{AF}\,\textit{fulfilled}$

meaning that within *all* executions (A), then the condition *fulfilled* will eventually (F) be true. It is important to note that such formal verification goes well beyond standard testing techniques, providing a comprehensive mathematical analysis of *all* the possible system behaviours [CGP99, BK08]. However, although standard model-checking is a strong way to assess temporal/dynamic properties, within pervasive and ubiquitous systems we ought to do more. Specifically, we need to take into account the inherent *uncertainty* within such systems. There is uncertainty about sensor accuracy, uncertainty about the veracity of schedule information, uncertainty about the speed of execution of components, uncertainty about the reliability of communication, etc. Crucially, there is uncertainty about exactly which context any particular component is in at a specific moment in time. Thus, any formal verification of the logical properties of such systems should take at least some of these into account during the verification process.

For this reason, we will here use a *probabilistic* model checking tool. The most widely used tool currently is PRISM [HKNP06]. This is a probabilistic model checker, which provides support for analysis of various different probabilistic structures such as Discrete-Time Markov Chains (DTMC) and Markov Decision Processes (MDP). Probabilistic models are described in the PRISM modelling language, a state-based language, and properties are specified in the logic PCTL[1] [HJ94]. This language then allows us to verify properties such as *"what is the probability that 'fulfilled' will occur on an execution path?"*

As we will see, this gives us additional possibilities to specify relevant properties.

3. Scatterbox: a message delivery system

Scatterbox [KSC+08] is a message forwarding system, which has been designed to serve as a test bed for context-aware computing in a pervasive environment. It provides a content-filtering service to its users by forwarding relevant messages to their mobile phones. The user's context is derived both by tracking his/her location and by monitoring his/her daily schedule. This context data is analysed, and situations are identified that indicate the user's level of interruptability. As messages arrive, Scatterbox forwards them to the subscribed users provided each user's available context suggests they are in a situation where they may be interrupted. Scatterbox consists of the following components, as shown in Fig. 1:

- *Construct*, a distributed, fully decentralised platform supporting the construction of context-aware, adaptive, pervasive and autonomous systems;
- *Bluetooth* and *calendar sensors* which provide Scatterbox with contextual data;
- an *e-mail handler* that interacts with an e-mail server to access a user's e-mail and determine an e-mail's importance and relevance to the user context;

[1] PRISM also supports Continuous-Time Markov Chains, but then uses a different logic for verification.

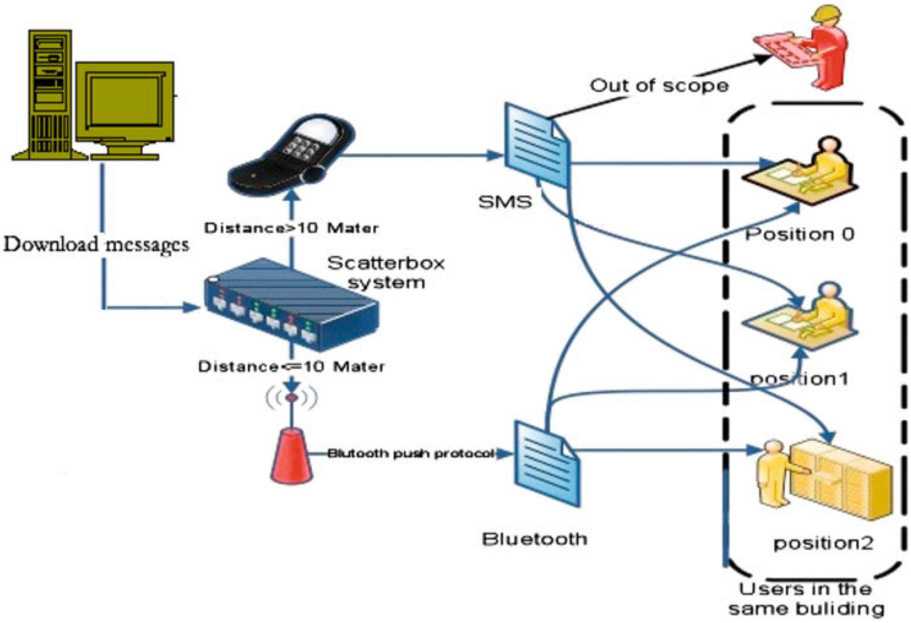

Fig. 2. Scatterbox message forwarding component

- a *situation reasoner* that takes these context data and determines whether or not the user is interruptible, and whether a particular message is relevant enough to be forwarded to the user; and
- the *message delivery component*, which sends relevant messages to the user via either Bluetooth or SMS depending on the distance.

Scatterbox's e-mail handler connects to the user's e-mail server as an IMAP client. Throughout the day, it downloads all unread messages and extracts information from them that will be used by Scatterbox's reasoning component to determine their importance. If Scatterbox decides to forward a message to the user's mobile device, the transmission is done through SMS or Bluetooth's Push protocol, which allows a file to be transferred between devices. If a user's Bluetooth device is in range of a Bluetooth-enabled node, a message can be routed to that node and pushed to the mobile device (see Fig. 2). When it arrives on the user's handset, the user has the opportunity to accept or reject the message.

4. Scatterbox message processing

Scatterbox combines the perceived level of interruptability of a user with the importance of the incoming e-mail to decide whether an incoming message should be forwarded or not.

Consider the following scenario. Bob is a postgraduate student. He is currently in a scheduled meeting with his supervisor. This meeting is recorded in Bob's calendar, and his phone is detected via Bluetooth in his supervisor's office. During the meeting, Bob receives three e-mails, two from colleagues, and one from Alice, his wife.

As this is an important meeting, Bob should only be notified of extremely important messages, while others can wait. The importance of a message is categorised as a range between 0 and 1, and is determined by 4 factors. The *correspondence*, the conversation *status* (FWD, RE: etc), the *subject*, and the *message* itself. As each of the e-mails described in Table 1 arrives, their importance is determined. The first e-mail is given a low importance, suggesting it is unlikely to ever be sent to Bob. The second mail is immediately given higher priority, not only as it is from Alice, Bob's wife, but because the subject contains keywords such as "urgent". The last e-mail comes from a colleague, but is a direct reply to an e-mail regarding a paper deadline, thus giving it more importance.

Table 1. Importance of an incoming e-mail is calculated from sender and subject

From	Subject	Importance
Colleague A	FWD: Funny joke	0.1
Alice	Urgent! Call me!	0.9
Colleague B	Re: Paper deadline	0.6

The interruptability of a user is discretised in a similar way to incoming e-mails, using a range from 0 to 1. The determination of interruptability is made in Scatterbox by analysing the user's location, situation participants, and calendar keywords (meeting, lunch etc...). The weights of each criterion can be externally defined, allowing different weights to be applied to different aspects. In this example scenario, Bob's location is away from his desk, in the company of his supervisor and has a calendar event called "meeting" scheduled. Bob is therefore given a low interruptability threshold, in this case "0.3". The decision to forward a message to a user is then based on two factors. The interruptability of a user and the importance of a message. These two numbers are simply multiplied together. A threshold value, defined in an external properties file, dictates how sensitive Scatterbox is to alerting a user. For example, in the Bob's scenario, the threshold value is 0.2.

Using this threshold value, and multiplying the interruptability by the current message importance, we get 0.03, 0.27 and 0.18 respectively for each of the e-mails in Table 1. Bob is alerted of the second e-mail, while Scatterbox stores the unsent messages until Bob's situation changes, and the calculation can be redone.

5. Formally modelling scatterbox

To evaluate Scatterbox's effectiveness in using context to determine which messages to forward and which messages not to forward, and the accuracy of these messages, a number of scenarios have been assessed. All these verifications have been carried out in a university domain, and the typical users were research students and staff. We have defined the following contexts which are typical and common for most of the users:

- HOME, denoting that the user is at home;
- OFFICE, denoting that the user is in his/her office;
- OTHER_OFFICE, denoting that the user in his/her colleague's office;
- MEETING, denoting that the user is in a meeting;
- LECTURE, denoting that the user is in a lecture;
- SEMINAR, denoting that the user is in a seminar;
- LUNCH, denoting that the user is having lunch;
- TEA, denoting that the user is having a tea break;
- TOILET, denoting that the user at the toilet;
- SPORT, denoting that the user in the sports centre;
- UNKNOWN, denoting that the user's context cannot be determined with the available sensor information.

We begin, in Fig. 3, by modelling a simple user behaviour. Since user movement might be different for each user, the generic model has non-deterministic transitions between all contexts. For example, the user might move from HOME to *any* other context. In Fig. 3, we only show transitions from HOME to other contexts. Actually, any transition is allowed between any pair of contexts. In Fig. 3 we also show how long a user might stay in a context. For example, if the user is in either MEETING, LECTURE, or SEMINAR, then it stays in that context for an hour; if the user is in LUNCH, then it will stay at least half an hour; if the user is in TEA, then it will leave that context in 10 min; etc. All the timing information was extracted from the user data obtained from statistics generated by the real system usage. As seen in Fig. 3, a user's *exact* interruptability, denoted by 'intr', is assigned to a value in each transition.[2] The interruptability values are user-specific, and are defined by users. Based on the statistics derived from real Scatterbox users, we assume that

- $\mathsf{intr} = 0.2$, if the user is in either LECTURE;
- $\mathsf{intr} = 0.3$, if the user is in either SEMINAR or MEETING;
- $\mathsf{intr} = 0.4$, if the user is in SPORT or TOILET;

[2] The assignments to intr are normally done at the nodes. But in Fig. 3 we show them at the transitions for illustration purposes.

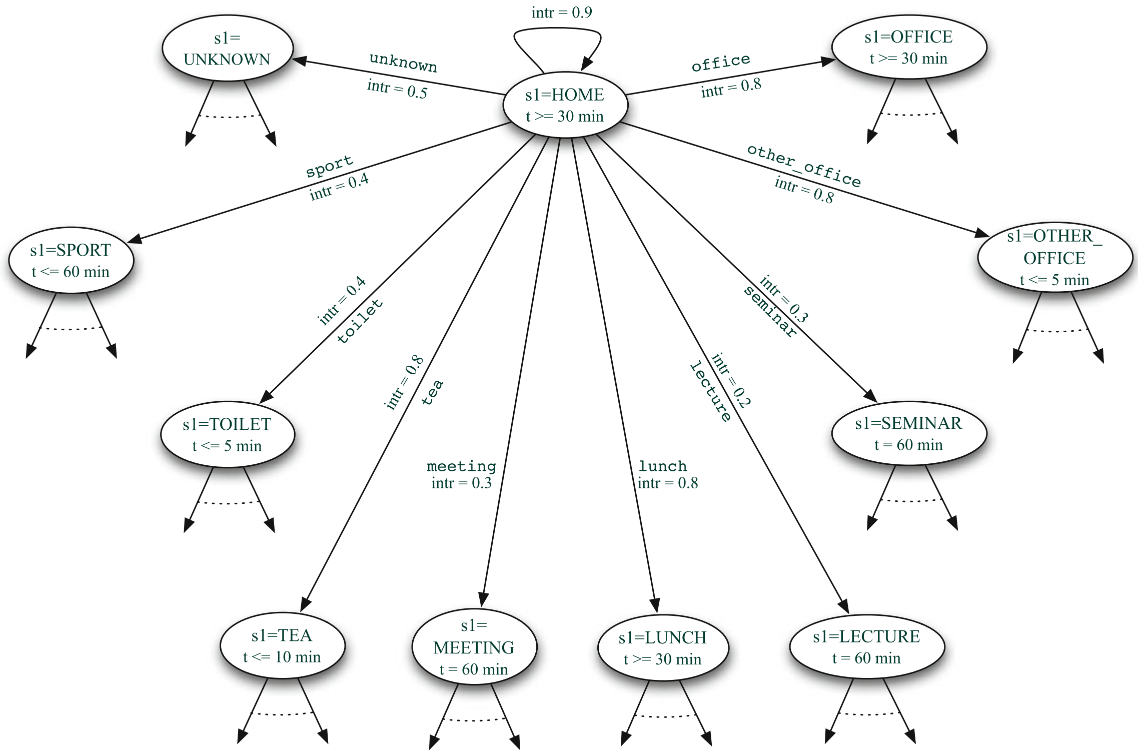

Fig. 3. Simple model of user movement

- intr = 0.5, if the context is UNKNOWN;
- intr = 0.8, if the user is in either TEA, LUNCH, OFFICE or OTHER_OFFICE;
- intr = 0.9, if the context is HOME;

Within Fig. 3, note that each context, such as MEETING, can lead on to every other context. We have indicated this within the figure, but have only expanded all the transitions for the HOME context.

The model of user movement in Fig. 3 denotes the average behaviour of Scatterbox users in a university domain based on statistics. Figure 3 is a general model for the users. It does not follow the exact movement of one particular user, because all the transitions are non-deterministic. The model does not force a particular trace. For example, after the LECTURE, one user might go to LUNCH, and the other user might go to OFFICE, all of which are expressed in the model. Actually, the movement and location of individual users does not affect our analysis, because the correctness of the system does not depend on individual user movements, it rather depends on how the system successfully determines the user interruptability and importance level of messages. Therefore, Fig. 3 could also be changed to reflect a particular user's behaviour, if necessary.

We remark that the exact interruptability level and the level that Scatterbox perceives might be different. If the accuracy of location sensors are high, then the user context can be gathered more accurately, and thus the perceived interruptability levels become closer to the exact levels. In the next section, we will analyse how the sensor accuracy, and therefore the context gathering accuracy, affects the accuracy of the Scatterbox system.

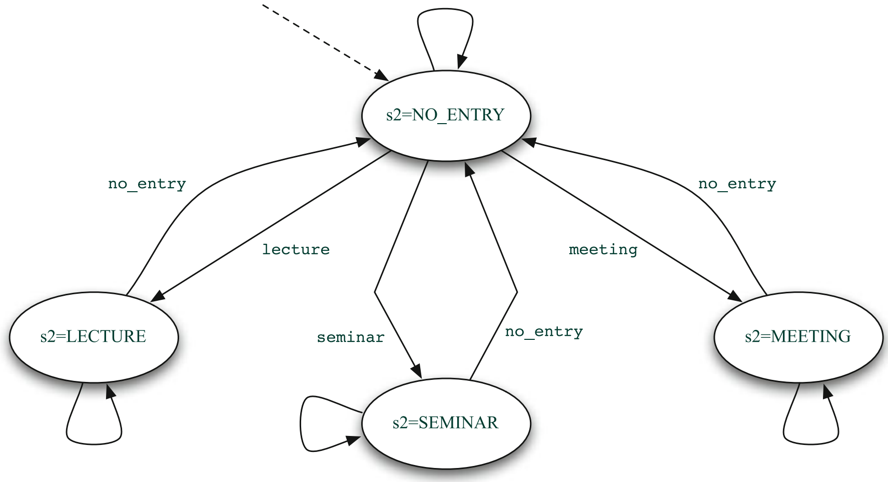

Fig. 4. Simple model of a user calendar

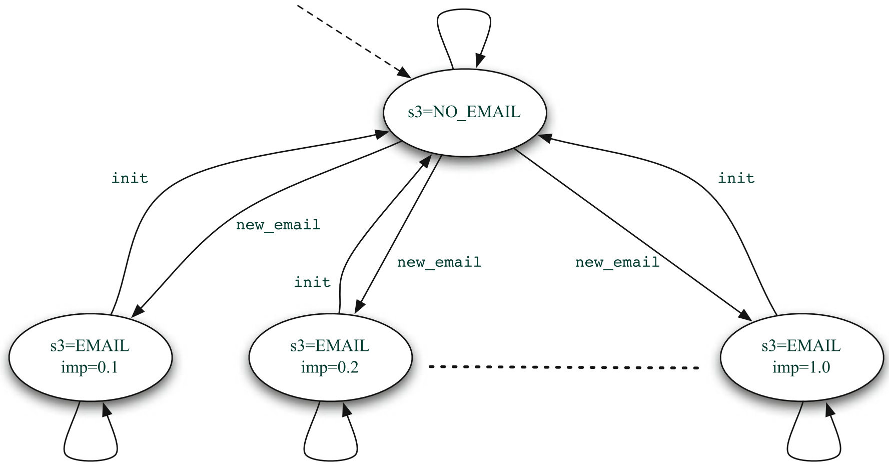

Fig. 5. Model of e-mail retrieval

In Fig. 4 we model the user's calendar. Here, each user might have a different calendar entry because of the non-deterministic choices between entries. It is important to note that the calendar model in Fig. 4 and the user movement in Fig. 3 are *synchronised* by having the same *action* labels on the transitions.[3] For example, if the transition from HOME to LECTURE is triggered in Fig. 3, then through the same label `lecture` the transition from NO ENTRY to LECTURE is activated in Fig. 4. We remark that the exact synchronisation is only possible if the user calendar is 100 % accurate, so it is expected that the user follows his/her scheduled activities. However, in real-life a user's calendar does not always coincide with the user's movement. So, the synchronisation depends on the calendar accuracy.

In Fig. 5 we model the e-mail retrieval behaviour. When a new e-mail arrives, the e-mail handler component determines its importance level, and assigns a value to it, from $\{0, 0.1, 0.2, \ldots, 0.9, 1\}$. In Fig. 5 this is shown by the assignments to '`imp`'. The non-deterministic transitions imply that a newly arrived e-mail can be at any importance level. In this way, we simulate the arrival of an arbitrary e-mail. This allows us to analyse the behaviour of the Scatterbox system for any message type. It is important to mention that the Scatterbox's perception of e-mail importance level depends on the e-mail handler's accuracy in classifying e-mails. That is, the exact importance

[3] In PRISM transitions can be labeled with actions. The actions can be used to to trigger several modules to make transitions synchronously (i.e. simultaneously). When modules are combined (using the standard CSP parallel composition), synchronisation is applied over all common actions.

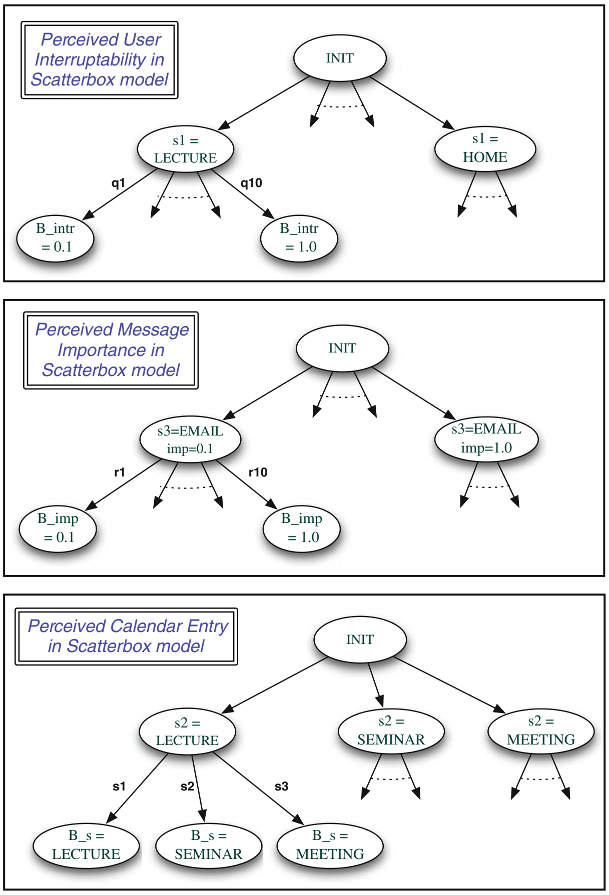

Fig. 6. Model of Scatterbox's beliefs about user interruptability, message importance and calendar information

level and the level that Scatterbox perceives might vary according to the e-mail handler's performance. In the following section we will formally analyse how the accuracy of e-mail classification affects the message forwarding behaviour of Scatterbox.

We now turn our attention to modeling the Scatterbox's perceived user interruptability, the perceived message importance and the perceived calendar information. In Fig. 3 the interruptability levels, denoted by intr, are the exact interruptability levels determined by the exact user movement. But the interruptability that the Scatterbox system perceives depends on the performance of the context acquisition process (i.e. accuracy of the sensors). We therefore distinguish between the *exact* interruptability and the Scatterbox's *belief* about interruptability, as these two might have different values. In the first part of Fig. 6 we show how the system's belief about the interruptability assigned according to the context acquisition accuracy. For example, the figure shows that if the user is in the context LECTURE, where intr $= 0.1$, then the perceived interruptability B_intr is assigned to 0.1 with a probability of $q1$; assigned to 0.2 with a probability of $q2$, and so on; finally, B_intr is assigned to 1.0 with a probability $q10$. Given that A is the accuracy of the context acquisition process (where $0 \leq A \leq 100$), $q1$ is calculated as $A/100$. Also, we have that $q2 + \cdots + q10 = 1 - q1$. If the accuracy of the context acquisition process is 100 %, then $q1$ is calculated as 1, and the perceived interruptability B_intr is assigned to 0.1 with probability 1.

In Fig. 5, imp denotes the exact importance level of the e-mail received. However, Scatterbox's perceived importance level, i.e. *belief* concerning the importance level, might be different because the system's belief depends on the performance of the e-mail handler component. We therefore distinguish between the *exact* importance imp and the *perceived* importance B_imp. In Fig. 6 we illustrate how B_imp is assigned to values. For example, if an e-mail arrives with importance 0.1, then B_imp is assigned to 0.1 with a probability $r1$, assigned to 0.2 with a probability of $r2$, and so on. Given that B is the accuracy of the e-mail handler component (where $0 \leq B \leq 100$), $r1 = B/100$ and $r2 + \cdots + r10 = 1 - r1$.

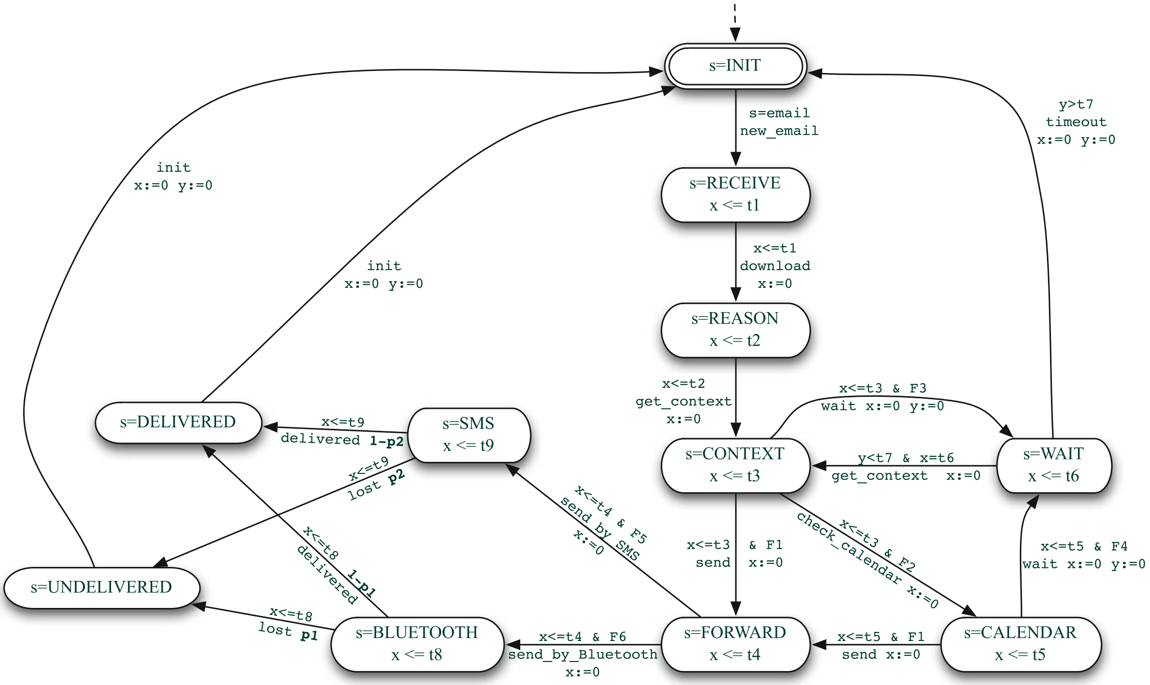

Fig. 7. Scatterbox model

Finally, we plot the *perceived* calendar entry of the user in Fig. 6. Similarly, if the user's calendar entry shows LECTURE, then the perceived calendar information B_s is assigned to LECTURE with a probability of $s1$, assigned to SEMINAR with a probability of $s2$, and assigned to MEETING with a probability of $s3$. Given that the accuracy of the calendar sensors are C (where $0 \leq C \leq 100$), we have that $s1 = C/100$ and $s2 + s3 = 1 - s1$.

We can now talk about the overall operation of the Scatterbox system. We first provide various abbreviations:

F1 $\doteq$ (B_intr × B_imp $\geq$ *Threshold*);
F2 $\doteq$ (B_intr × B_imp $<$ *Threshold* & s1 = UNKNOWN);
F3 $\doteq$ (B_intr × B_imp $<$ *Threshold* & s1 $\neq$ UNKNOWN);
F4 $\doteq$ (B_intr × B_imp $<$ *Threshold*);
F5 $\doteq$ (s1 = LECTURE | MEETING | TEA | TOILET | SEMINAR | LUNCH | SPORT | UNKNOWN);
F6 $\doteq$ (s1 = OFFICE | OTHER_OFFICE | HOME).

We use *discrete-time probabilistic timed automata* to model the behaviour of the system. Probabilistic timed automata (PTA) are a formalism used for modelling systems whose behaviour exhibits non-deterministic, real-time and probabilistic characteristics [KNSS02].[4] PTA are an extension of timed automata [AD94], one of the most prominent formalisms for the formal verification of real-time behaviour. PTA are in general defined on dense time models because the clocks of PTA are real-valued variables. However, some studies [KNSS02, KNPS06], show that for a large class of real-time verification problems, correctness can be established using bounded integer-time models instead of dense-time models. In this paper, we follow the same approach and consider a discrete model of time.

The Scatterbox system operates as follows (see Fig. 7): When an e-mail arrives, the e-mail handler component downloads it within t_1 s. The e-mail handler then determines the importance level that Scatterbox perceives, and assigns it to the e-mail. This is done within t_2 s. The situation reasoner queries Construct to get the available context data, which were obtained by Construct through sensors, and from the data it derives the user context. By

[4] While various definitions of PTAs can be found, we here use the definition of [KNSS02].

using this context information, the perceived interruptability of the user is determined. This process is executed within t_3 s. Since Scatterbox has calculated the message importance level and the user interruptability level, it can now decide whether to forward or not to forward the message. If the multiplication of the user interruptability and message importance is greater and equal to the threshold value (expressed as F1), then Scatterbox forwards the message within t_4 s. If the result of the multiplication is less than the threshold value and the user context is UNKNOWN (expressed as F2), then the system consults the user calendar to check whether it includes any information about the user context, and it then updates the user interruptability according to the entry in the calendar information. This process is carried out within t_5 s. If the multiplication of the user interruptability and message importance is less than the threshold and the user context is other than UNKNOWN (expressed as F3), then the system does not forward the message and decides to wait for t_6 s. Unless the total waiting time exceeds t_7 s, it checks the context information every t_6 s. The message forwarding component forwards a message the user's mobile device through either Bluetooth or SMS. If the user is in either OFFICE, OTHER_OFFICE, or HOME (expressed as F6), then the message is sent through Bluetooth within t_8 s. If the user is in either MEETING, LECTURE, SEMINAR, LUNCH, TEA, TOILET, SPORT, or UNKNOWN, then the message is sent through SMS within t_9 s. The probabilities of the communication error for Bluetooth and SMS are $p1$ and $p2$, respectively.

Each model in Figs. 3, 4, 5, 6 and 7 is modelled as a PRISM module. The overall probabilistic model is constructed as the *parallel composition* of these modules. Namely, all modules are combined using the standard CSP parallel composition to construct the overall system model. This process is done automatically by PRISM.

6. Formally verifying the scatterbox system

We now turn to the formal verification of properties of the Scatterbox system. We first translated all the transition systems in the previous section into the PRISM input language and then provided the properties to be analysed in the logic PCTL. Through PRISM we are then able to automatically verify that the required properties are true on the system model.

PCTL is an extension of the branching temporal logic CTL. Apart from the usual operators from classical logic such as $\wedge$ (and), $\vee$ (or) and $\Rightarrow$ (implies), PCTL has the *probabilistic operator* $P_{\sim p}[.]$, where $p \in [0, 1]$ is a *probability bound* and $\sim$ is one of $<$, $\leq$, $\geq$, or $>$. Two forms of path formulas α are allowed: $X\phi$ is true in a state of a path, if, and only if, ϕ is satisfied in the next state; and $\phi_1 U^{\leq k} \phi_2$ is true if, and only if, ϕ_2 satisfied within k time-steps and ϕ_1 is true up until that point. Path formulas can occur only within the scope of the probabilistic operator. Intuitively, a state s of a model satisfies $P_{\sim p}[\alpha]$ if, and only if, the probability of taking a path from s satisfies α in the interval specified by '$\sim p$'. As an example, the property that "the probability of F eventually being true is greater than p" can be expressed in PCTL as follows: $P_{>p}[\texttt{true}\ U^{\leq\infty} F]$.

We now present the results from running PRISM on our earlier model with different properties and parameters. The parameters used in these experiments are as follows: $t_1 = t_2 = t_3 = t_4 = t_5 = t_6 = t_8 = t_9 = 30$ and $t_7 = 300$ s. We also take the communication accuracy as being 99 %. i.e. $p_1 = p_2 = 0.99$. Below we present the verification results that we obtained.

We first analyse how the *context acquisition accuracy* affects the correctness of Scatterbox. The context acquisition accuracy mainly depends on the *accuracy of the location sensors*. If location sensors cannot monitor the user location accurately, then the situation reasoner will generate misleading context information. This will result in an incorrect user interruptibility level (see Fig. 6), and the accuracy of message forwarding behaviour will be reduced.

The correctness of action selection (i.e. forward or wait) is informally stated as *"the probability of deciding on an action correctly"*. This property can be written in the PCTL language as the following formula, Φ:

$$P_{=?}\Box((s = \mathit{Forward}) \Rightarrow (\mathsf{intr} \times \mathsf{imp} \geq \mathit{Thr})) \ \wedge \Box((s = \mathit{Wait}) \Rightarrow (\mathsf{intr} \times \mathsf{imp} < \mathit{Thr}))$$

which queries the probability of the following:

> it is always the case that if it is decided to forward a message, then the multiplication of the *exact* user interruptibility level and the *exact* message importance must be greater than or equal to the threshold value, but if it is decided not to forward the message, this multiplication must be less than the threshold value.

Here, ? is regarded as query by PRISM, and it returns the probability calculated. We know that the decision to forward was taken based on the *perceived* user interruptibility B_intr and *perceived* importance level B_imp. Namely, we check whether this is also the case with the *exact* interruptibility intr and importance level imp. The property Φ clearly checks the probability of Scatterbox deciding on an action at the appropriate times. If this probability is high, then we can be more confident about the Scatterbox deciding on an action correctly.

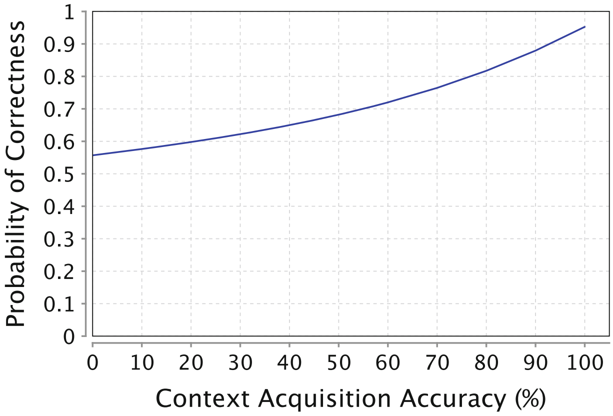

Fig. 8. Probability of correctness (Φ) vs. context acquisition accuracy

We verified the correctness property with respect to different context acquisition accuracies. The result is shown in Fig. 8. PRISM plots the probability values p for the range of 0–100 of the accuracy values, which denote the accuracy level of location sensors. As we can see, if the context acquisition accuracy increases, then the probability of Scatterbox deciding on an action at the appropriate time increases. From these results we can conclude that correctness is very sensitive to the context acquisition accuracy, and thus the sensor accuracy. Note that in Fig. 8 we assumed the message classification accuracy and calendar accuracy are both 100 %.

In Fig. 9 we analyse how the *accuracy of the e-mail classification* of the e-mail handler component affects the correctness. Intuitively, if the e-mail classification cannot be correctly determined, then the importance levels cannot be assigned correctly. This will reduce the correct action selection within the Scatterbox system. This behaviour can be observed in Fig. 9, where we verify the correctness property Φ with respect to e-mail classification performance of the e-mail handler component. As can be seen, this probability is directly affected by the performance of the e-mail handler component. Note that, in Fig. 9, we assumed the context acquisition accuracy and calendar accuracy are 100 %.

In Fig. 10 we analyse how the *calendar accuracy* affects the correctness property Φ. According to this figure, the probability of deciding on a correct action increases if the accuracy of the calendar increases. This increase in probability is not as significant as it was in Figs. 8 and 9, because Scatterbox queries the calendar information *only* if there is no context information available. So we can expect that the Scatterbox system does not frequently query the calendar information, and therefore the calendar accuracy does not affect correctness as much as the context acquisition accuracy and message classification accuracy do. In Fig. 10 the context acquisition accuracy and message classification accuracy are assumed to be 100 %.

It is worth mentioning that in Figs. 8, 9 and 10 the probabilities of correctness reach a maximum of 0.95. This is because, although the context acquisition accuracy is 100 %, after the Scatterbox system decides on an action (forward or wait), the user might change the CONTEXT, reducing the correctness accuracy.

Figures 8, 9 and 10 check correctness according to one variable at a time. We also carried out verification experiments where we checked the correctness according to the context acquisition accuracy and message classification accuracy at the same time. Figure 11 shows the results as a 3D surface map. Using this figure we can easily check the correctness at different combination values.

Above we have shown that we can analyse the probabilistic elements of the system. In PRISM, we can verify many other interesting properties, e.g. *safety, liveness, real-time*, etc. In order to show the usability of probabilistic model checking we verified the properties described in Table 2. Here, we consider multiple combined accuracies, i.e. we assume that the accuracy of context acquisition, message classification and calendar sensor is 75 %. Similarly, we could consider any arbitrary accuracy level.

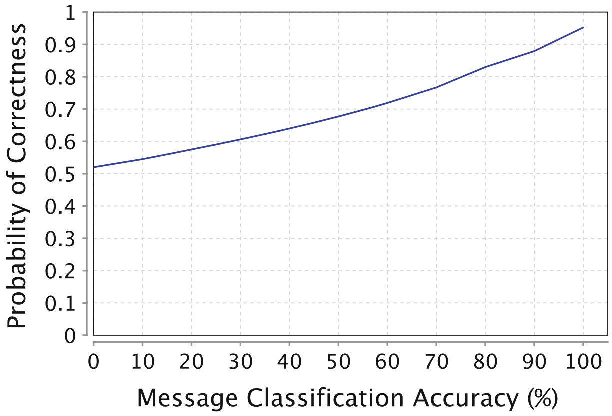

Fig. 9. Probability of correctness (Φ) vs. accuracy of the e-mail handler

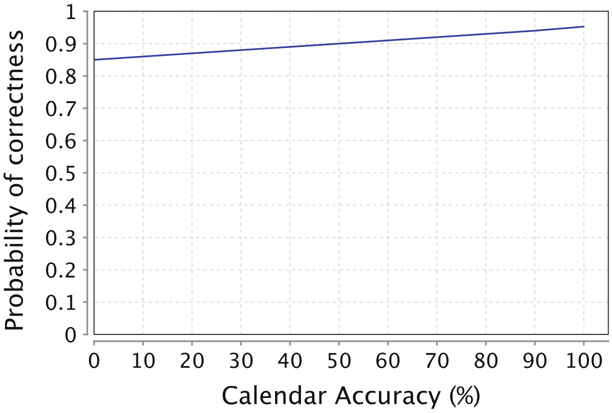

Fig. 10. Probability of correctness (Φ) vs. calendar accuracy

Before explaining the properties in more detail, we remark that the verification result returned by PRISM is calculated at (single) initial state of the model. But, "since PRISM in fact usually has to compute values for *all* states simultaneously, PRISM properties can be customised to obtain different results, which is done using *filters* [Pri11]." The filters are denoted by the following notation:

$$\mathbf{filter}(\texttt{op}, \texttt{prop}, \texttt{states})$$

where `op` denotes a filter operator, e.g. `min`, `max`, `avg`, etc., `prop` denotes a PRISM property, and `states` denotes a set of states over which **filter** is applied. For example, $\mathbf{filter}(\texttt{op}, \mathrm{P} =?[\Diamond \text{“}\texttt{deadlock}\text{”}], s = 1)$ returns the *average* probability value of reaching a "`deadlock`" state starting from any state which satisfies $s = 1$. For illustration purposes, in Table 2, we denote $\mathbf{filter}(\texttt{avg}, \texttt{prop}, \texttt{states})$ as $\texttt{prop}\{\texttt{states}\}$.

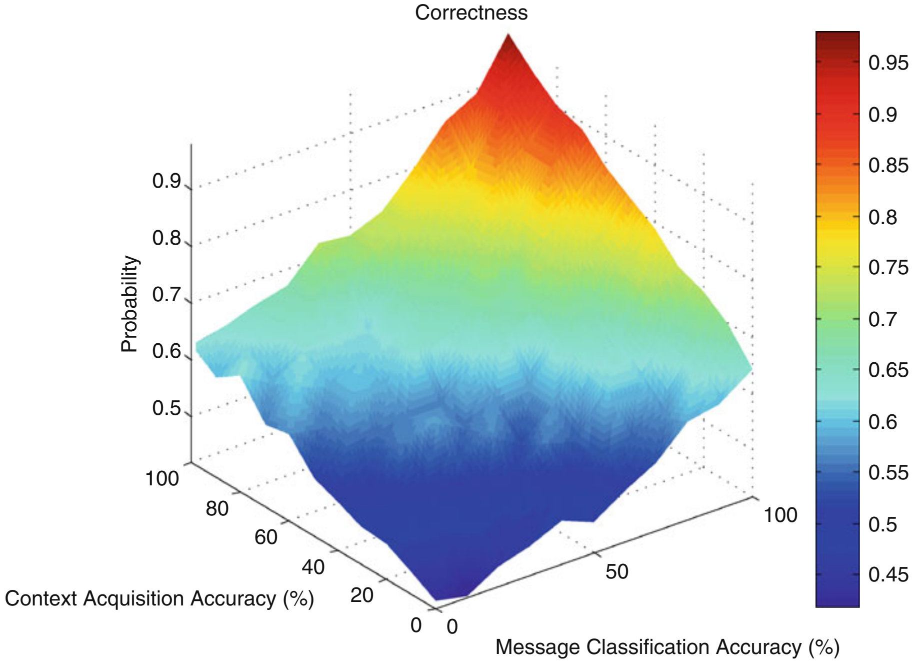

Fig. 11. Probability of correctness (Φ) vs. context acquisition accuracy and message classification accuracy

Table 2. Further sample properties of scatterbox

Prop.	Informal and formal specification	PRISM vrf.
1	Probability of an arbitrary message eventually being delivered $P_{=?}[\Diamond(s = \mathit{Delivered})]\{s = \mathit{Receive}\}$	0.68
2	Probability of an arbitrary message remaining undelivered $P_{=?}[\Diamond(s = \mathit{Undelivered})]\{s = \mathit{Receive}\}$	0.32
3	Probability of a message tagged as 'important' being delivered $P_{=?}[\Diamond(s = \mathit{Delivered})]\{s = \mathit{Receive} \wedge \mathsf{imp} \geq 9\}$	0.88
4	Probability of an arbitrary message being delivered within 200s $P_{=?}[\Diamond^{\leq 200}(s = \mathit{Delivered})]\{s = \mathit{Receive}\}$	0.60
5	Probability of an arbitrary message being delivered in 400s $P_{=?}[\Diamond^{\leq 400}(s = \mathit{Delivered})]\{s = \mathit{Receive}\}$	0.68
6	Probability of an 'important' message being delivered within 200s $P_{=?}[\Diamond^{\leq 200}(s = \mathit{Delivered})]\{s = \mathit{Receive} \wedge \mathsf{imp} \geq 9\}$	0.83
7	Probability of an 'important' message being delivered within 400s $P_{=?}[\Diamond^{\leq 400}(s = \mathit{Delivered})]\{s = \mathit{Receive} \wedge \mathsf{imp} \geq 9\}$	0.88
8	Probability of an 'unimportant' message being delivered when the user is in an uninterruptible context $P_{=?}[\Diamond(s = \mathit{Delivered})]\{s = \mathit{Receive} \wedge \mathsf{imp} \leq 3 \wedge \mathsf{intr} \leq 3\}$	0.11

Table 3. Verification experiments

Property	Verification		Property	Verification	
	Memory (GB)	Time (s)		Memory (GB)	Time (s)
1	2.1	787	5	2.1	671
2	2.1	765	6	2.0	112
3	2.1	785	7	2.1	676
4	2.0	111	8	2.1	766

In Table 2 we gradually refine the queries. Prop. 1 analyzes delivery of an arbitrary message. As the verification result shows the probability of an arbitrary message eventually being delivered is 0.68. This is because this message might be at any importance level (Note that Prop. 2 is the counterpart of Prop. 1). If we consider an 'important message' (denoted as $\mathsf{imp} \geq 9$), the probability of being delivered should increase. Indeed, Prop. 3 verifies that the probability increases from 0.68 to 0.88. Properties 1 and 3 specify 'eventuality' without imposing any time bound. We expect that if we restrict that the message should be delivered within a certain amount of time, the probability should decrease. Prop. 4 shows that if we assume the message should be delivered in 200 s, the probability reduces to 0.60. If we allow more time for the delivery, we expect that the probability should increase. Indeed, Prop. 5 shows that if we increase the time bound to 400 s, the probability increases to 0.68. Prop. 4 and 5 assume arbitrary messages. If we consider 'important' messages, the probability should increase, as Prop. 6 and 7 verify. Prop. 8 shows that the delivery of an 'unimportant' message (denoted as $\mathsf{imp} \leq 3$) is very unlikely when the user is in an uninterruptible context (denoted as $\mathsf{intr} \leq 3$).

Clearly the properties in Table 2 are only indicative of the sort of properties which could be proved, but they still illustrate important temporal effects that are implicit in Scatterbox's rule base, but which are only made explicit by formal verification.

Experiments with PRISM were run on a 2.4 Ghz Core 2 Duo computer with 6 GB RAM running Mac OS 10.5.7. Table 3 illustrates the statistics of the verification experiments in Table 2. As seen in the table, the memory used is just above 2.0GB, and the execution time is in the range of 700s. Here, we also note that the state space of the Scatterbox model is 1.2×10^8 and the number of transitions is 2.6×10^9. The model is constructed in 4 s. In PRISM, we could also extract Boolean answers. This is achieved by replacing the question mark in the query $P_{=?}$ with a bound expression, e.g. $P_{\geq 0.1}$. In this case, we would receive a YES or NO answer instead of receiving an actual probability.

In the verification experiments above, the state space and the number of transition are in the level of 10^9. The number of parallel components and non-deterministic actions, in general, scales up the state space, but in our case there are only a few parallel components with non-deterministic transitions. Therefore, this is not the only reason for the scalability issue. We modeled the user behaviour according to the statistics obtained from real usage of the system. Since user movements are different for each user, the generic model has non-deterministic transitions between all contexts. We also included timing information for each context that users stay in. This level of modelling unavoidably scales up the state space and the transition matrix.

From this we conclude that if a context-aware system requires modelling the exact user behaviour, scalability will be an issue even for small systems. This problem, however, can be partially treated. First of all, we can use different scales for time information to scale down the resulting state space. Second, we can use a more abstract representation of the user behaviour model by combining contexts which have close interruptibility level. In this way, the user movement will be between context groups rather than individual contexts. Applying these two partial treatments to the Scaterbox model reduces the resulting state space approximately 2–3 orders of magnitude.

Another useful approach treating the scalability is to use a *population* approach, which can be applied if individual modelling of each variable in the system is not required or avoided. Namely, if there are many identical, independent processes, the population approach allows us to abstract away from low-level probabilistic details and so just consider global population behaviour. It is simply applied using variables as counters to count the number of agents in the corresponding states. [KNP08, KDF12] shows that the state space of the resulting model reduces dramatically, e.g. in the level of 9–10 orders of magnitude in some cases.

In this paper, we did not apply these workarounds, because the model construction times (in the level of seconds) and verification times (in the level of minutes) were still in the reasonable bounds despite the sizes of the state space and transition matrix. We also did not want to deviate from the collected user data.

7. Verifying pervasive systems

Although we have only carried out detailed verification on *one* pervasive application, we suggest that the approach considered here could weel be applicable to a wider range of pervasive systems. Below we provide our reasons for such optimism.

Uncertainty. As stated in many publications on pervasive systems, for example [DN05, DSNH10], the modelling of uncertainty is a vital element in the detailed analysis of pervasive applications. As we have shown within our case study, uncertainty of many aspects can be modelled and verified: sensor accuracy; user behaviour; context categorization. The use of probabilistic formal verification now gives a general and increasingly powerful tool for tackling such uncertainty through a wide range of pervasive systems.

Analysis. It is clear that pervasive systems must be analyzed before being deployed in important, or even critical, areas. Just carrying out basic testing is unlikely to be sufficient to achieve this. Tests only represent individual scenarios and such tests are often chosen to examine the *most likely* causes of error. Formal verification, on the other hand, provides a much more comprehensive analysis technique. It allows us to examine *all* scenarios and so we can not only ask "does this scenario lead to an error?", as is typical in testing, but can also ask "what scenario can lead to an error?". This provides the developer with the tools to find many errors, often in unexpected areas.

Ubiquity. Although we have tackled just one application, the architecture and components we have modelled and verified are very typical of many pervasive applications. Most such systems have networks of sensors, communication mechanisms, context modelling processes, and control involving decisions based on both context and communication. As shown earlier in the paper, these components are all modelled as separate probabilistic entities within the verification approach. This not only leads us to contend that the analogous nature of many pervasive applications will allow us to use similar modelling and verification approaches, but we also believe that building a toolkit of verifiable models for such architectural components is a route towards the wider analysis of pervasive systems.

Modelling. The development of automata representing the key components of the system, such as sensing, communication, and context acquisition, appears relatively straightforward. However, one potentially difficult aspect concerns providing the probabilities used in these automata. Where do these come from? How do we know they are plausible? We are fortunate with Scatterbox in that, since the system is already deployed, substantial data concerning real usage is available. From this data we can extract probabilistic models of a range of "typical" behaviours, which then form the basis for our user models, etc. Similarly, we have significant data on the error margins within sensors, etc. If no data is available on some aspect of the system, then model checking allows us to leave out specific probabilistic values and so effectively consider *all* arbitrary choices. This non-determinism means that *any* combination possible must be verified; this is general, but costly. Thereafter, any refinements available will help restrict the possibilities that need to be considered. Thus, whether we have specific data about the system or not, formal verification can still be applied.

Thus, while we have verified only a *selection* of properties of a *single* pervasive system, we believe that many pervasive systems could be analyzed in a similar way. We must also emphasize that such an analysis allows us to assess not just whether the system is *correct* or *not*, with respect to some formal requirements, but to *quantify* how likely it is to work correctly, given (bounded) uncertainty in the environment. This aspect allows the system designer to tune their pervasive application to have 'acceptable' levels of correctness given the range of error predicted in devices. For example, consider Property 8 from Table 2. Here, there is the undesirable property of an unimportant message being delivered when the user is in an uninterruptible context. According to our verification, this will occur with 11 % probability. The designer might well feel that this is too much and so might change parameters in the system, incorporate better models of the world, or introduce more accurate components (sensors) to reduce this. Subsequent probabilistic model-checking will show that the probability of the undesirable property occurring has been reduced. For example, if we increase the accuracy of context acquisition, message classification and calendar sensors from 75 to 90 % and re-run the verification experiments, the result of Prop. 8 reduces to 0.04. In this case, we are able to decrease the likelihood of undesirable properties.

However, we must also emphasize that efficiency and scalability can sometimes be an issue with (probabilistic) model-checking tools. If this turns out to be the case, then there is much work on abstraction that might well be exploited. For example, if large numbers of, essentially identical, sensors are to be modelled then *counting* abstractions are a well established rounte to simplified verification problems.

8. Related work

Although there have been some attempts at modelling pervasive systems [CFJ03, HI04, SB05, Sim07, WZGP04], there has been relatively little work on formalising and even verifying aspects of pervasive and ubiquitous scenarios. As described earlier, formal verification has rarely been used in the analysis of pervasive systems. Work in [CSRR09] describes a formalism to model context-aware web services, and provides some algorithms to verify context-aware web services against some qualitative properties such as "non-determinism, liveness of states and rules, and absence of blocking states". However, this does not target probabilistic and temporal properties. We believe these aspects to be essential for analyzing pervasive systems that incorporate some (and often many) unreliable hardware components. [ACD$^+$09] analyzes a pervasive home-care application; but that paper only presents the specification of the system properties and actual verification is not performed. Meanwhile [CGU09], studies the modelling and verification of context-aware systems whose models are defined in terms of rule sets. That paper considers a case study of an event driven health-care system; but the properties studied are very limited in the sense that it only considers checking redundancy within the rule set using SAT-solvers. Many important characteristics of pervasive systems such as temporal behaviours, uncertainty, etc. are omitted. In [BBD$^+$06] bigraphical reactive systems are used to model certain aspects of pervasive computing. In [WBB06] a programming paradigm is suggested for pervasive applications based on ambient calculus. And, in [RC08], a formal model is proposed for describing pervasive computing environments based on ambient calculus and an associated ambient logic. Both Coronato and Pietro [CDP11] and Boytsov and Zaslavsky [BZ13], provide frameworks for formal specification and verification different to ours, while a forthcoming journal special issue [BCC$^+$12] promises to provide a snapshot of the state of the art in this area. Very recently [LZD$^+$12], has carried out a very similar work to ours. The authors propose a formal framework which captures main aspects of pervasive systems, such as context-awareness, concurrent communications, layered architecture, etc. They identify and formally specify several critical properties of pervasive systems, such as deadlock-freeness, guaranteed services, security, inconsistency etc., in suitable logics. They also apply their approach to a pervasive nursing home system.

There are other probabilistic model checkers in the literature, such as MRMC [KKZ05], YMER [You05] and VESTA [SVA05]. YMER and VESTA are statistical model checkers, so we do not prefer using them to to be able verify *all* system behaviour, which cannot not be analysed by statistical model checkers. We have not used MRMC, because it does not support discrete-time MDPs. Also [JKO$^+$07], shows that PRISM outperforms MRMC for large models.[5] This is mainly due to the fact that the overhead calculation of PRISM for generating MTBDDs to represent the transition matrix is compensated in large models. MRMC has recently been improved with a better memory management and implementation of the sparse matrices and support for bisimulation minimization [KZH$^+$11]. We therefore expect some improvement in performance on large models. Note that there have been some improvements in PRISM as well, e.g. symmetry reduction. We do not provide an evaluation of the recent versions of these tools, as the authors do not have any empirical data to compare their performances.

Recently, a new model checker has been introduced to analyze hierarchical probabilistic real-time systems [SLS$^+$11]. The modeling language called PRTS is used to specify these systems, which in turn are used to generate finite state MDPs. PRTS has been implemented in the model checking framework PAT which provides a user interface to edit, simulate and verify PRTS models. Through some experiments the authors show that the tool outperforms PRISM in dense-time. In analysing low-level properties, e.g. communication and security protocols, we think PTRS could be more advantageous because of its efficiency in dense-time, and its support for hierarchical systems.

9. Conclusion

In this paper we have studied the formal verification of a deployed pervasive system. We assessed dynamic and temporal behaviour of the system and analysed the impact of *probabilistic* elements within the overall behaviour. We therefore verified formal requirements even in the presence of unavoidable uncertainty in sensors, calendar information, communications, context accuracy, and so on. Our analysis has shown that model-checkers such as PRISM can be useful in verifying these aspects of pervasive applications. Furthermore, we believe that the techniques we have applied here can be used in analysing probabilistic behaviour of other pervasive systems, and potentially more general sensor-driven adaptive systems.

[5] [JKO$^+$07] actually shows that models up to a few million states MRMC mostly performs better than PRISM ; but for larger models PRISM performs better.

Any pervasive system is inherently probabilistic, and as programmers we have a relatively under-developed ability to program with uncertain information. The impact of uncertainty combined with rule-based decision-making on a system's external behaviour remain implicit in the rule and code structure. Without proper analysis it is impossible to be sure that a system will demonstrate the appropriate properties across *all* possible evolutions. Formal verification provides more confidence than is possible through testing or simulation. By using a probabilistic method we are able to quantify the expected reliability of a system even in the face of unreliable components.

There are two key problems that occur in all such approaches: (1) what properties do we verify; and (2) where do the probabilities come from. The first of these remains a problem with all formal analysis techniques and, clearly, significant work must be done in capturing the requirements of the system in a formal and logical way. We have been able to avoid the usual problems of (2) by tackling an actually deployed system. Since Scatterbox is in use, data is available through which we are able to give 'justifiable' probability values. Similarly, the potential error ranges of sensors and communications can be extracted through practical use. While all these values can never be perfect, their basis in real activity gives us some confidence that the results of our analysis are relevant.

In this paper we mainly focused on high-level properties, such as action selection accuracy and message delivery. As a future research direction we intend to extend our work with low-level analysis of the system. We want to analyse the communication and security protocols, to be able to check properties such as "a message sent to the user A can never be delivered to user B". More broadly, we are currently exploring the extensions needed to formal methods in order to demonstrate properties such as stability in the face of small perturbations, with a view to allowing the verification of the next generation of sensor-driven adaptive systems.

Acknowledgments

This work is partially funded by EPSRC within the "Verifying Interoperability Requirements in Pervasive Systems" project (EP/F033567), involving the Universities of Birmingham, Glasgow, and Liverpool. Stephen Knox was supported by Enterprise Ireland under the grant "Towards a Semantics of Pervasive Computing".

References

[ACD+09] Arapinis M, Calder M, Denis L, Fisher M, Gray P, Konur S, Miller A, Ritter E, Ryan M, Schewe S, Unsworth C, Yasmin R (2009) Towards the verification of pervasive systems. In: 3rd international workshop on formal methods for interactive systems (FMIS), Vol 22. Electronic Communications of the EASST

[AD94] Alur R, Dill DL (1994) A theory of timed automata. Theor Comput Sci 126(2):183–235

[BBD+06] Birkedal L, Bundgaard M, Damgaard TC, Debois S, Elsborg E, Glenstrup AJ, Hildebr T, Milner R, Niss H (2006) Bigraphical programming languages for pervasive computing. In: International workshop on combining theory and systems building in pervasive computing

[BCC+12] Bakhouya M, Campbell R, Coronato A, de Pietro G, Ranganathan A (2012) Introduction to special section on formal methods in pervasive computing. ACM Trans Auton Adapt Syst 7(1):6:1–6:9

[BDM+98] Bozga M, Daws C, Maler O, Olivero A, Tripakis S, Yovine S (1998) Kronos: a model-checking tool for real-time systems. In: CAV '98 Proceedings of the 10th international conference on computer aided verification. Springer, Berlin, pp 546–550

[BK08] Baier C, Katoen J-P (2008) Principles of model checking. MIT Press, Cambridge

[BLL+95] Bengtsson J, Larsen KG, Larsson F, Pettersson P, Yi W (1995) Uppaal—a tool suite for automatic verification of real–time systems. In: Proceedings of workshop on verification and control of hybrid systems III(1066). Lecture Notes in Computer Science, pp 232–243. Springer, Berlin

[BZ13] Boytsov A, Zaslavsky A (2013) Formal verification of context and situation models in pervasive computing. Pervasive Mob Comput 9(1):98–117

[CCGR99] Cimatti A, Clarke E, Giunchiglia F, Roveri M (1999) Nusmv: a new symbolic model verifier. In: Proceedings of international conference on computer-aided verification (CAV'99), pp 495–499

[CDP11] Coronato A, De Pietro G (2011) Formal specification and verification of ubiquitous and pervasive systems. ACM Trans Autonom Adap Syst 6(1):9

[CFJ03] Chen H, Finin T, Joshi A (2003) An ontology for context-aware pervasive computing environments. Knowl Eng Rev 18(3):197–207

[CGP99] Clarke EM, Grumberg O, Peled D (1999) Model checking. MIT Press, Cambridge

[CGU09] Calder M, Gray P, Unsworth C (2009) Tightly coupled verification of pervasive systems. In: 3rd international workshop on formal methods for interactive systems (FMIS), Vol 22. Electronic Communications of the EASST

[CSRR09] Cubo J, Sama M, Raimondi F, Rosenblum D (2009) A model to design and verify context-aware adaptive service composition. In Proceedings of the IEEE International Conference on Services Computing (SCC). pp 184–191

[DN05] Dobson SA, Nixon P (2005) More principled design of pervasive computing systems. In: Bastide R, Palanque PA, Roth J (eds) Proceedings of the joint working conferences on engineering human computer interaction and interactive systems (EHCI-DSVIS), vol 3425 of LNCS, pp 292–305. Springer, Berlin

[DSNH10] Dobson S, Sterritt R, Nixon P, Hinchey M (2010) Fulfilling the vision of autonomic computing. IEEE Comput 43(01):35–41

[HI04] Henricksen K, Indulska J (2004) A software engineering framework for context-aware pervasive computing. In: Proceedings 2nd IEEE conference on pervasive computing and communications. pp 77–86

[HJ94] Hansson H., Jonsson B (1994) A logic for reasoning about time and reliability. Formal Asp Comput 6:102–111

[HKNP06] Hinton A, Kwiatkowska M, Norman G, Parker D (2006) PRISM: a tool for automatic verification of probabilistic systems. In Proceedings of the TACAS, vol 3920 of LNCS, pp 441–444. Springer, Berlin

[Hol03] Holzmann GJ (2003) The spin model checker. Addison-Wesley, Reading

[JKO+07] Jansen DN, Katoen J-P, Oldenkamp M, Stoelinga M, Zapreev IS (2007) How fast and fat is your probabilistic model checker? An experimental performance comparison. In: Haifa verification conference. pp 69–85

[KDF12] Konur S, Dixon C, Fisher M (2012) Analysing robot swarm behaviour via probabilistic model checking. Robot Auton Syst 60(2):199–213

[KKZ05] Katoen J-P, Khattri M, Zapreev IS (2005) A Markov Reward model checker. In: QEST. pp 243–244

[KNP08] Kwiatkowska MZ, Norman G, Parker D (2008) Using probabilistic model checking in systems biology. SIGMETRICS Perform Eval Rev 35(4):14–21

[KNPS06] Kwiatkowska M, Norman G, Parker D, Sproston J (2006) Performance analysis of probabilistic timed automata using digital clocks. Form Methods Syst Des 29(1):33–78

[KNSS02] Kwiatkowska M, Norman G, Segala R, Sproston J (2002) Automatic verification of real-time systems with discrete probability distributions. Theor Comput Sci 282(1):101–150

[KSC+08] Knox S, Shannon R, Coyle L, Clear A, Dobson S, Quigley A, Nixon P (2008) Scatterbox: context-aware message management. Rev Intell Artif 22(5):549–568

[KZH+11] Katoen J-P, Zapreev IS, Hahn EM, Hermanns H, Jansen DN (2011) The ins and outs of the probabilistic model checker MRMC. Perform Eval 68(2): 90–104

[LZD+12] Liu Y, Zhang X, Dong JS, Liu Y, Sun J, Biswas J, Mokhtari M (2012) Formal analysis of pervasive computing systems. IEEE Int Conf Eng Complex Comput Syst 0:169–178

[Pri11] PRISM Manual (2011) http://www.prismmodelchecker.org/manual

[RC08] Ranganathan A, Campbell RH (2008) Provably correct pervasive computing environments. IEEE Int Conf Pervasive Comput Commun 0:160–169

[SB05] Sheng QZ, Benatallah B (2005) Contextuml: a UML-based modeling language for model-driven development of context-aware web services. In: Proceedings of the International Conference on Mobile Business (ICMB'05) pp 206–212. IEEE Computer Society Press

[Sim07] Simons C (2007) CMP: a UML context modeling profile for mobile distributed systems. In: Proceedings of the 40th Hawaii International Conference on System Sciences, p 289. IEEE Computer Society Press

[SLS+11] Sun J, Liu Y, Song S, Dong JS, Li X (2011) PRTS: an approach for model checking probabilistic real-time hierarchical systems. In ICFEM. pp 147–162

[SVA05] Sen K, Viswanathan M, Agha G (2005) On statistical model checking of stochastic systems. In: CAV. pp 266–280

[WBB06] Weis T, Becker C, Brändle E (2006) Towards a programming paradigm for pervasive applications based on the ambient calculus. In: International workshop on combining theory and systems building in pervasive computing

[WPBF02] Want R, Pering T, Borriello G, Farkas KI (2002) Disappearing hardware. Pervasive Comput, IEEE 1(1):36–47

[WZGP04] Wang XH, Zhang DQ, Gu T, Pung HK (2004) Ontology-based context modeling and reasoning using OWL. In: Context modeling and reasoning workshop at PerCom 04. pp 18–22

[You05] Younes HLS (2005) Ymer: a statistical model checker. In: CAV. pp 429–433

Received 29 June 2012
Revised 8 February 2013
Accepted 11 February 2013 by Dong Jin Song