Human Factors in Design, Engineering, and Computing, Vol. 238, 2026, 1798–1807
https://doi.org/10.54941/ahfe1008408

Applying Formal Methods to an Augmented Reality Interface

Jeremy Gottlieb[1], Dan Bryce[1], Mieszko Muskala[2], Stephen Gilbert[2], Michael C. Dorneich[2], Shayama Khan[2], Maral Jafari Ranjbar[2], Eliot H. Winer[2], John Skovbekk[1], Robert P. Goldman[1], and David Musliner[1]

[1]Smart Information Flow Technologies (SIFT), Minneapolis, MN 55401, USA
[2]Iowa State University, Ames, IA, 50011, USA

ABSTRACT

This study applies formal methods utilizing probabilistic reasoning to integrated system models containing a human cognitive component. As the line between user, environment, and computational system becomes increasingly blurred by immersive technologies, being able to prove safety properties of the full, integrated system, instead of just the automation component increases in importance. We present a model of a human-machine system based on augmented reality and use it to reproduce results from human-centered studies of cognitive load. Specifically, we formulate a model of feature integration theory of visual search as probabilistic timed automata. The model satisfies properties related to task performance such as identify whole system. The properties, in turn, provide system design and implementation guidance. Our end-to-end modeling, verification, and validation presented here corroborates the applicability of probabilistic model checking to the human/environment/automation system in an augmented reality (AR) application. These formal models permit designers to guarantee system properties, as well as assess the impact of design choices without conducting full user studies for each decision.

Keywords: Formal modelling, Augmented reality, Probabilistic timed automaton, System design

INTRODUCTION

Computational system design has traditionally focused narrowly on the computing hardware and software, accounting for the user through restrictions on the possible inputs and outputs. However, immersive technologies such as augmented and virtual reality (AR and VR) make it more difficult to control user inputs and, in the case of AR, the information presented to the user. In such cases, effective designs must consider not just the hardware and software, but also human cognitive behaviors and their interactions with the broader operating environment through the AR/VR system.

Formal methods, such as probabilistic model checking, are a powerful tools for conducting analyses of human-in-the-loop systems, given that very few human cognitive behaviors are deterministic. Formal methods involve the mathematical specification of a system, typically as some type of automaton describing system states and transitions, and the actions that transform one

Received September 26, 2026; Revised October 21, 2026; Accepted October 27, 2026; Available online December 1, 2026

© 2026 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

state to another. This allows for rigorous mathematical verification of desired properties of that system, such as avoiding failure states or traversing states in a particular order (NASA, 2024).

Model checking techniques, implemented in software such as PRISM (PRobabilistIc Symbolic Model checker) (Kwiatkowska *et al.*, 2011), verify model properties by assessing the reachability of states of the formal model. PRISM checks a variety of probabilistic models, which are largely reducible to a probabilistic state transition system. The exact form of the transition system varies by model type, and admits different representational (e.g., explicit, symbolic, or abstract) and algorithmic (e.g., search-based, statistical, or optimizing) techniques. PRISM checks properties expressed in PCTL, a logical language for asserting the probability that the model can reach specific states (e.g., that the probability of reaching a failure state is less than 1e-6).

Probabilistic Timed Automata (PTAs) (Beauquier, 2003) are especially attractive for modeling software+environment+human AR/VR systems because they incorporate non-determinism and probabilistic uncertainty. PTA non-determinism captures uncertainty in what transitions will be selected by software or the human and when the transition will occur. PTA probabilities model uncontrollable outcomes due to the environment. The transition delays correspond to intervals; for example, a system might take between 50 and 100msec to compute the next frame for display. Transition non-determinism models software choice of annotating a frame, or human choice of focusing attention on objects in the frame. Probabilistic outcomes govern how objects in the environment behave. An alternative, simple example is applying PTAs to modeling flipping a fair or a biased coin – the choice of coin and delay associated with the flip is non-deterministic and the outcome is probabilistic (i.e., either 0.5 or 0.99 heads depending on the coin). PTA models of AR/VR systems provide a valuable surrogate for evaluating alternative systems designs. Such model-based engineering is able to rule out system designs that cannot perform to specification before they are fully developed.

This paper presents an example of applying a long-standing cognitive theory of visual search to an augmented reality task, building a probabilistic model of the theory and AR task, using this model to make predictions about system performance using the PRISM model checker. We also validated model predictions in experiments with human participants. These validated can dictate design properties for human-machine systems.

A FORMAL MODEL OF VISUAL SEARCH

Task Domain Background

We model a checkpoint surveillance task (Figure 1). The observer is using a head-mounted augmented reality system to observe the scene around the checkpoint from an elevated vantage point. The augmented reality headset provides the observer with an automated target recognizer (ATR) that aids in identifying attackers. The observer's goals are to: (1) tag all attackers before they reach the checkpoint and (2) maximize checkpoint throughput. Anyone

tagged as unsafe will be denied passing through the checkpoint. Figure 2 shows a model of the task.

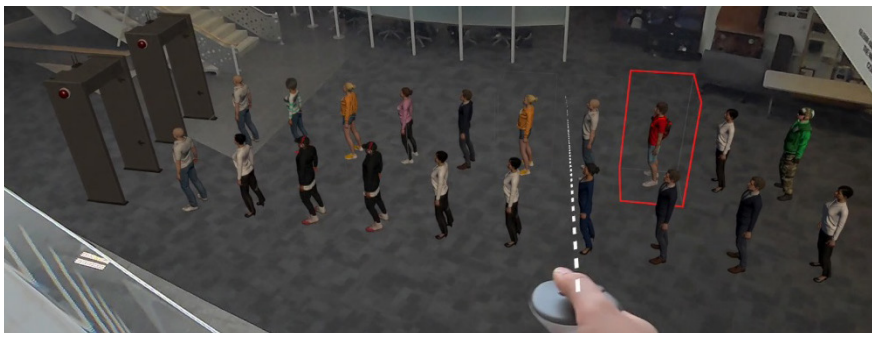


Figure 1: Checkpoint surveillance task domain.

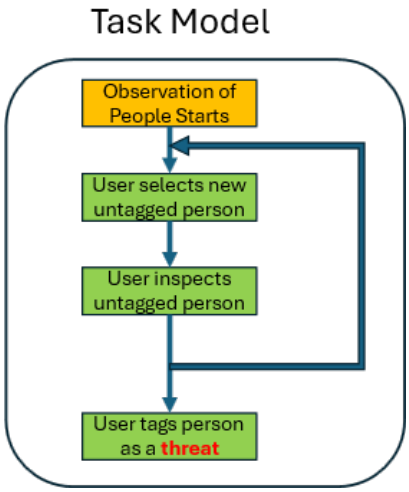


Figure 2: Task model.

The specific version of the task being modelled is based on Kelley *et al.* (2024). The study involves subjects in a VR environment looking at a wall with 32 windows. There were 20 people in the 32 windows, one of whom was wearing a uniform identifying them as an attacker. The study explored which of a set of target indicators helped people find an attacker most quickly.

Kelley *et al.* applied Triesman's (1982) *feature integration theory* (FIT) to VR. FIT predicts that searching for an object with a highly salient and distinguishing feature should be simple, *i.e.* it should "pop out", and take essentially constant time, regardless of how many distractor objects there are. Under FIT, the expected time to find the target object (E{*TT*}) should be:

$$E\{TT\} = C + I \tag{1}$$

where C is a constant that reflects the amount of time it takes the user to interact with the environment to select the target object (in our case, tagging

with a laser pointer), which they only do once, and I is the average amount of time it takes the user to inspect an object to determine if it is in fact the target object. If the target pops out, then the user only needs to inspect the target object to verify it before engaging the selection process.

In contrast, searching for an object *without* distinguishing features should be more time-consuming as people search through distractors looking for the target object. This type of search is called *serial self-terminating search*. When there is only one attacker to be found, the user examines each person one at a time until they find the attacker. The expected tagging time in a task with n objects/people to search is:

$$E\{TT\} = C + \frac{nI}{2} \tag{2}$$

Kelley *et al.* (2024)'s research objective was to determine which of a set of target indicators provided by ATR most allowed the target to "pop out," enabling participants to find the attacker most quickly. In their experiments the ATR was perfectly accurate – the target indicator was always applied to the true target. In all cases, participants were faster than without a target indicator, demonstrating that FIT still holds in a virtual reality environment.

Current Task Environment

The task environment for this study differs from Kelley *et al.* (2024) in several ways. First, instead of a building façade in VR our study uses an AR scene of two lines approaching a checkpoint. Second, we fix the target indicator (a bounding box in the AR headset), rather than testing different indicators. Third, we use $P(tp)$, the probability that the ATR correctly recognizes the target (where tp denotes true positive) as a proxy to the target inspection time I. We assume that for a given indicator, I and $P(tp)$ are correlated, i.e., I ~ $P(tp)$. By varying $P(tp)$, we are indirectly varying I, but the interpretation changes slightly. Instead of the salience of the target indicators, we are changing the reliability of the target recognizer.

The advantage of studying this task as a PTA, given that we are converting between time and probability (per above) is that we can quantify how salience, time, and probability interact (e.g., modeling how increasing target salience decreases the mean I, and how increasing I also increases reliability). However, in order to validate our approach with respect to Kelley *et al.*, we limit our current focus to a fixed, implicit salience and inspection time.

Our AR environment has the user standing on a balcony overlooking an atrium. In the atrium are twenty (20) virtual avatars organized into two lines in front of a pair of checkpoints. The lines do not move, and one avatar is indicated by the ATR with a bright red bounding box, as shown in Figure 3. Participants were tasked with finding the avatar wearing a backpack. We defined the ATR reliability $P(tp)$ in the interval [0.7, 1.0]. For example, if $P(tp) = 0.75$ the red bounding box correctly surrounded the avatar with the backpack in 0.75 of the trials. We also included a set of trials with no bounding box at all.

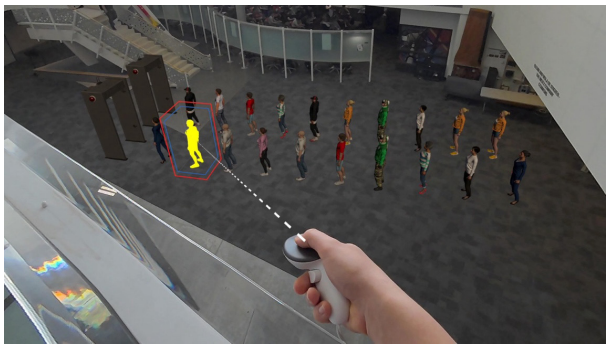


Figure 3: Screenshot of AR task with user selecting potential attacker.

We predict the time it should take the user to inspect people until they locate the attacker as a function of the reliability of the single bounding box provided by the ATR (i.e., its true positive rate, *P(tp)*). With probability *P(tp)*, the user correctly locates the attacker (that is boxed by the ATR) according to Equation 1 and with probability *(1–P(tp))* they locate the attacker (not boxed by the ATR) according to Equation 2 so that:

$$\begin{aligned} E\{TT\} &= P(tp)[C+l] + (1-P(tp))\left[C+\frac{nl}{2}\right] \\ &= C + I\left[P(tp) + \frac{n}{2}(1-P(tp))\right] \end{aligned} \tag{3}$$

Formal Model

We modeled the above task as a *probabilistic timed automaton* (PTA) (Beauquier, 2003) using the notation of the PRISM model checking system (Kwiatkowska *et al.*, 2011). Timed automata are finite automata where state transitions depend on, and modify the value of, several clock variables. PTAs extend timed automata so that transitions out of a state are probabilistic.

PRISM models specify a set of *modules* (i.e., component automata). PRISM composes these modules into an automaton and formally analyzes whether it satisfies properties such as reachability. The PRISM model of the surveillance task consists of four modules:

- A *user* module that controls the search behavior of the user, including accounting for the time it takes the user to conduct operations, such as the time *I* to inspect an individual.
- An *attacker* module initializes which person in the environment is the attacker either with probability *P*(*tp*) if boxed, or with uniform probability otherwise.
- A *tagging* module that decides whether the person currently being examined should be tagged (i.e., recognized at the attacker).
- A *deadline* module that keeps track of total elapsed time and signals if the deadline has been reached without the attacker being tagged.

The user module describes the main behavior of interest, and the other modules are largely used for bookkeeping about the environment. This specific implementation of the model has 20 people in the environment, one of whom is the attacker. We consider cases where there is no ATR and where the reliability of the ATR ranges from 0.7 to 1.0. There is always exactly one person with a bounding box and one attacker, but they may not be the same person. An unreliable ATR can mark the wrong person as the attacker.

The user module (see Figure 4) always (without loss of generality) inspects people in order from 0 to N-1, as this allows PRISM to implicitly track who has already been inspected. It follows from this that we can model the no-ATR case by randomly assigning the attacker to one of the indices. This is equivalent to random assignment and random inspection: the probability of finding the attacker within k inspections is identical in either case.

When there is an ATR bounding box in the model, the box is always placed around person 0, and that person is assigned as the attacker with a probability equivalent to the ATR reliability being tested and randomly assigned to one of the other 19 people otherwise. This way, the user always inspects the marked person first, as expected. If the boxed person is not the attacker, the user starts inspecting the other people sequentially as in the No Box case until they find the attacker. This is equivalent to the user first examining a boxed person and then searching for the attacker among the non-boxed people, if the cue is incorrect.

```
module user            // Model of user choosing to examine people
  t1 : clock;                           // reaction time
  curr_per_id: [0..N] init 0;           // id of current person
  curr_per_box: [0..2] init use_box;    // is current person boxed

  invariant  // user must examine the current person immediately
    (curr_per_id < N & curr_per_box=0 & tagd_attkr=0 & sat_deadln=0
     => t1 <= unbox_exam_tm) &
    (curr_per_id < N & curr_per_box=1 & tagd_attkr=0 & sat_deadln=0
     => t1 <= box_exam_tm)
  endinvariant

  // Transition 1: allow exam_box transitions when curr_per_box
  [exam_box] curr_per_box=1 & curr_per_id=0 & t1 >= box_exam_tm ->
             (t1'=0) & (curr_per_id'=curr_per_id+1) & (curr_per_box'=0);
  // Transition 2: allow unboxed tagging when not curr_per_box
  [exam] curr_per_box=0 & t1 >= unboxed_examine_time ->
         (t1'=0) & (curr_per_id'=curr_per_id+1);
  // Transition 3: disallow unboxed tagging when boxed
  [exam] curr_per_box=1 & t1 > deadline ->
         (t1'=0) & (curr_per_id'=curr_per_id+1);
  // Transition 4: disallow exam_box transitions when not curr_per_box
  [exam_box] curr_per_box=0 & t1 > deadline ->
             (t1'=0) & (curr_per_id'=curr_per_id+1);
endmodule
```

Figure 4: PRISM module for user behavior in surveillance task.

Finally, PRISM evaluates a PTA against a query to assess the probability of reaching a state where the attacker has been identified within a *deadline*. The minimum deadline is the time required to identify and tag the attacker before reaching the checkpoint. If we let $T_{i,CP}$ denote the time it would take the attacker to reach the checkpoint, then accounting for the time (C parameter in Equations 1-3) to use the AR interface to tag a person once they have identified the attacker, the deadline is defined:

$$deadline = T_{i,CP} - C \tag{4}$$

The success state in the model is thus only reached if the user picks the attacker prior to the deadline. From Equation 3, we see that the probability of success depends on both (1) the reliability of the ATR and (2) how many people they can inspect before the deadline relative to how many people there are to inspect. If there is no ATR, then this reduces to simply:

$$P(success) = \frac{1}{n}\frac{deadline}{I} = \frac{deadline}{nI} \tag{5}$$

If there is an ATR that is, say, 80% reliable, then 80% of the time the user will find the attacker after only inspecting one person – the person the ATR has marked with the bounding box[1]. The other 20% of the time the user will have to search, and their likelihood for success is governed by Equation 4. Combined, this yields:

$$P(success) = P(tp) + (1 - P(tp))\left(\frac{deadline}{nI}\right) \tag{6}$$

Figure 5 shows *P(success)* from our PRISM model with a variety of ATR reliabilities and deadlines, with an arbitrary choice of $I = 1$. What is important to understand for model validation and design is that the exact value of I is less relevant than the fact that the model predicts that the amount of time it takes a user to find the attacker will scale linearly with how many other people the user has to inspect before finding the attacker. This prediction can guide design principles such as by improving the ATR, and/or providing sufficient time for the user to inspect the number of people required to locate the attacker.

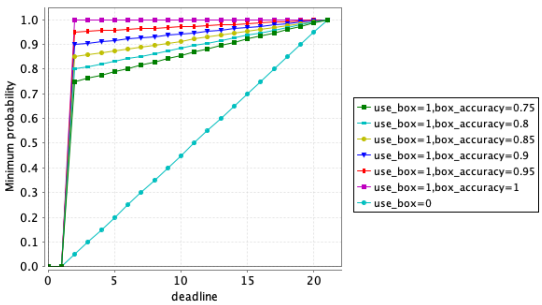


Figure 5: P(success) for PRISM model with N = 20 and I = 1 for different deadlines.

[1]This assumes the user does not make errors. Out of 11,030 total trials, there were only 72 errors, or roughly 0.65%. No individual participant made more than 3 errors out of roughly 240 trials, and almost 40% of participants made zero errors. Thus, we decided that this assumption was close enough for the proof-of-concept presented herein.

A graph identical to Figure 5 can be generated directly from Equation 5 by substituting the appropriate values for the parameters in that equation. As this is a proof of concept for demonstrating how probabilistic formal models can incorporate human cognitive models to help prove system properties, this match is intentional. It validates that the PRISM model is the correct task model *vis a vis* the underlying cognitive model it embodies, and thus can be used for future predictions and design.

MODEL VALIDATION WITH HUMAN PARTICIPANTS

We conducted a user study to validate the model's predictions and estimate the *I* and *C* parameters. The design was as follows: Participants stood on a balcony looking down one floor at two static lines of 10 avatars each and had to identify and tag the attacker avatar. Every participant did 20 trials of the task with no ATR to assist them and then did 60 trials with an ATR that was either 75%, 85%, or 100% reliable. Participants were told that the ATR might not be completely reliable.

Our primary measure was how long from the start of a trial it took participants to tag the attacker. We also attempted to indirectly determine how many avatars the user inspected before deciding who to tag as the attacker. This measurement was indirect because we were not using an eye tracking headset. The only indication we had of which avatars the user examined was which avatars the selection ray from the hand controller (the white dashed line in Figure 3) intersected with. Unfortunately, there will be incidental intersections as the participant moves the selection ray through the environment to select the avatar they have identified as the attacker. For this reason, we only counted an intersection as an inspection event if there was no additional event within 50ms. We assumed a longer delay between inspection events indicated that the user was focusing on that potential target.

RESULTS

Table 1 shows the mean tagging time and number of inspections for each of the various ATR levels. Note a ceiling effect where the perfect ATR case does not show significantly different performance from the 85% reliable ATR. In the perfect ATR case participants were told in the instructions that it might make mistakes, and these results would indicate that they never behaved as if the ATR were perfect. Due to this, we never saw a case where there was only one inspection as they seem to have examined other potential options, even if they ultimately selected the avatar indicated by the bounding box. Despite that, we can clearly see a trend where the less reliable the ATR is, the more inspections participants performed, and the longer it took them to find the attacker.

Table 1: Mean tagging time and inspections.

ATR Reliability	Trials	Tagging Time (ms)	Inspections
No ATR	2710	1883.75	5.78
70%	2896	1546.67	4.69
85%	2714	1325.35	4.01
100%	2710	1319.31	3.52

Furthermore, Figure 6 allows us to confirm the linear relationship between the number of inspections a user performs and how long it takes them to find the attacker, which was the core prediction of the model.

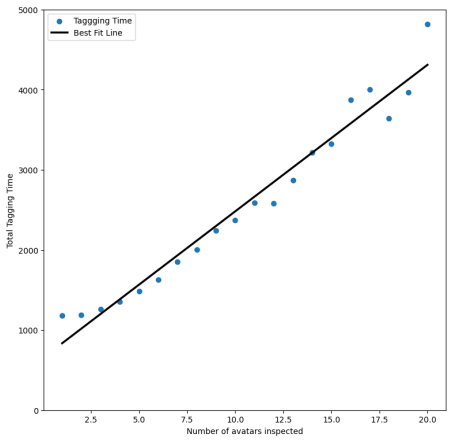


Figure 6: Tagging time by number of inspections.

CONCLUSION

From these results, we can extract design parameters for the full software-environment-human system. The only free parameters in the model are the deadline, the ATR reliability, and how long it takes a user to inspect a person and decide if they are the attacker (I). We can estimate I from human trials in with the AR system, but the effect of changes in I will be to change the slopes of the lines in Figure 5, not to change the overall pattern of system behavior. Presuming the ATR reliability is a known quantity (usually manufacturers tout some version of this statistic), we can take a graph like Figure 5 and choose a desired level of user performance. Then the point where the line for a given ATR reliability reaches that level of performance determines the minimum deadline needed to achieve a desired user performance.

For example, using Figure 5 we can see that if the ATR is 75% reliable (the green line) and we want users to, at worst, have time to identify 90% of attackers, then it must take people at least 14 clock ticks to reach the checkpoint. This

information can be used to design the physical space outside the checkpoint to ensure this minimum is reached, thus providing a guarantee about performance within the entire software-environment-human system.

Overall, this study demonstrates the utility of developing formal models that encompass all the elements of an augmented reality system – the headset software, the environment, and, importantly, the human user. Validating these models with human research and then using the predictions from those formal models informs design decisions. This ensures the full system can fulfill the desired performance properties.

ACKNOWLEDGMENT

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). This material is based upon work supported by the DARPA under Contract No. HR0011-24-9-04. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views, opinions and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of DARPA or the U.S. Government.

REFERENCES

Beauquier, D., 2003. On probabilistic timed automata. Theoretical Computer Science, 292(1), pp. 65–84.

Kelley, B., Wickens, C., Warden, A. C., Clegg, B., & Ortega, F. (2024, September). 2D and 3D Augmented Reality Attention Cueing Comparisons in 3D Target Search. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (Vol. 68, No. 1, pp. 1176–1180). Sage CA: Los Angeles, CA: SAGE Publications.

Kwiatkowska, M., Norman, G. and Parker, D., 2011, July. PRISM 4.0: Verification of probabilistic real-time systems. In International conference on computer aided verification (pp. 585–591). Berlin, Heidelberg: Springer Berlin Heidelberg.

National Aeronautics and Space Administration. (2024, April 12). Langley Formal Methods Program • What is Formal Methods. https://shemesh.larc.nasa.gov/fm/fm-what.html

Treisman, A. (1982). Perceptual grouping and attention in visual search for features and for objects. Journal of experimental psychology: Human perception and performance, 8(2), 194.