Reasoning about When to Provide Explanation for Human-in-the-loop Self-Adaptive Systems

Nianyu Li
Peking University
li_nianyu@pku.edu.cn

Javier Cámara
University of York
javier.camaramoreno@york.ac.uk

David Garlan
Carnegie Mellon University
garlan@cs.cmu.edu

Bradley Schmerl
Carnegie Mellon University
schmerl@cs.cmu.edu

***Abstract*—Many self-adaptive systems benefit from human involvement, where a human operator can provide expertise not available to the system and perform adaptations involving physical changes that cannot be automated. However, a lack of transparency and intelligibility of system goals and the autonomous behaviors enacted to achieve them may hinder a human operator's effort to make such involvement effective. Explanation is sometimes helpful to allow the human to understand why the system is making certain decisions. However, explanations come with costs in terms of, e.g., delayed actions. Hence, it is not always obvious whether explanations will improve the satisfaction of system goals and, if so, when to provide them to the operator. In this work, we define a formal framework for reasoning about explanations of adaptive system behaviors and the conditions under which they are warranted. Specifically, we characterize explanations in terms of their impact on a human operator's ability to effectively engage in adaptive actions. We then present a decision-making approach for planning in self-adaptation that leverages a probabilistic reasoning tool to determine when the explanation should be used in an adaptation strategy in order to improve overall system utility. We illustrate our approach in a representative scenario for the application of an adaptive news website in the context of potential denial-of-service attacks.**

I. Introduction

A self-adaptive system is designed to be capable of modifying its structure and behavior at run time in response to changes in its operational environment and the system itself (e.g., faults, changing requirements and attacks) [1], [2]. Although automation is one desirable characteristic of self-adaptation, many adaptive systems also benefit from human involvement: e.g., when a human operator has contextual information that is not available to the system. In these cases, the system may be able to adapt more effectively when its adaptation strategies can exploit human actions [3], [4], [5]. For example, in a security setting, it is common to combine automated actions for intrusion detection and remediation with human-guided decision making to defend the system against potential attackers [6].

Enabling effective human-system coordination, however, is challenging mostly because humans may not fully understand the system state, goals, and rationale for carrying out their actions [7]. To deal with this problem, there has been recent considerable interest in using system-generated "explanation" as a way to improve human understanding and ability to cooperate with the system [8], [9].

While explanations have been shown to be effective in many circumstances, they also come with costs typically in the form of delayed human actions (since the human will need to read and understand the explanation before carrying out an action). Thus, it is important for the system to be able to reason about when to use explanation: too little explanation, and the human may not be as effective; too much, and the system will be slow to respond to adaptation needs.

In this paper we propose a formal framework for such reasoning. Specifically, we see explanation as an action (or *tactic*) that the system can use judiciously in strategies involving human participation. Such tactics come with associated context-sensitive costs and benefits that affect overall system utility, and allow a probabilistic planner to determine the optimal use of explanation under an uncertain environment (where uncertainty is induced, among other factors, by human behavior and other external conditions –like the amount of malicious clients attacking the system). Key to this framework is the use of explicit models of human capability and willingness [10], which are affected positively by explanation, as well as the notion of *delays* induced by explanation which may negatively affect responsiveness to adaptation needs.

As we elaborate, using our framework allows an on-line planner to determine when explanation will be effective. Important features of the framework are that it can be tuned as needed to accommodate: (a) different levels of capability of the human operator, (b) the impact that explanation will have on an operator's willingness and capability to perform an action, (c) the ability to decide dynamically on a per-tactic basis whether explanation is warranted, and (d) different costs that arise from timing delays when using explanation.

The main contributions in this paper are:

- A formal framework for designing self-adaptive systems where explanation can be used as a tactic to aid the human operator in improving overall system utility;
- The use of probabilistic model checking to analyze trade-offs of using explanation in an adaptation strategy;
- An illustration of the applicability and benefits of our approach in collaborative mitigation of distributed denial-of-service (DDoS) attacks on an enterprise system.

The rest of the paper is structured as follows. Section II describes a motivating scenario of a DDoS attack on an adaptive news website. Section III introduces the background on probabilistic model checking and discusses related work. Section IV illustrates how the formal modeling of human involvement can be leveraged through probabilistic model

checking to reason about explanation as a potential tactic, while Section V presents analysis results. Section VI concludes the paper with potential directions for future work.

II. Motivating Scenario

To illustrate our explanation framework, we employ Znn.com [11], [6], an adaptive news website portraying a representative scenario for self-adaptive systems. Pages in the website can render multimedia content to enhance user experience, or could show only text content which requires less computation. A load balancer distributes stateless requests from users to a pool of replicated servers, which deliver the requested contents.

Sometimes, Znn.com might experience spikes in requests resulting from either legitimate client traffic caused by a popular event (slashdot effect), or denial-of-service (DoS) attacks in which malicious clients try to exhaust system capacity to render system services unavailable. In these cases, the system cannot serve all the requests adequately and users will experience unexpected long response time. The organization needs to take actions to maintain responsive service provision while keeping the cost of operating the infrastructure within budget (ideally, this includes not incurring operating costs that correspond to resources consumed by malicious clients). In addition, the actions taken should minimize interference with legitimate user experience. In short, we identify four concerns that can be mapped to the following quality attributes:

- Response Time (R), the time elapsed from receiving the request to sending the response from the server;
- User Annoyance (A), the percentage of disturbed users due to defensive tactics;
- Cost (C), the resources being operated in the system (e.g., number of active servers);
- Client Maliciousness (M), the percentage of malicious clients.

Znn.com can generally employ the following four tactics for dealing with the spikes in requests:

- *enlistServer*, commissioning a new replicated web server to share the load;
- *lowerFidelity*, reducing the level of service to text only;
- *blackhole*, adding the IP addresses of clients that are deemed to be attacking the system to a blacklist that blocks their requests;
- *throttle*, limiting the rate of requests accepted from potentially malicious clients.

The first two tactics are fully automated and could absorb the excess of user traffic without increasing the annoyance to legitimate users. However, they cannot handle malicious clients and can result in an unnecessary increment of the cost of operating the system. On the contrary, the last two tactics could suppress the spikes due to a DoS attack and require a human operator to accurately identify attackers. A well-trained operator will be effective at eliminating traffic from malicious clients, but a poorly trained one might increase user annoyance by misidentifying attackers and causing service disruption to legitimate clients.

Tactic	Response Time		Malicious Clients		Cost		User Annoyance	
	Δ(ms)	ΔU_R	Δ(%)	ΔU_M	Δ(usd/hr)	ΔU_C	Δ(%)	ΔU_A
enlistServers	-1000	$\uparrow\uparrow\uparrow$	0	$=$	+1.0	$\downarrow\downarrow\downarrow$	0	$=$
lowerFidelity	-500	$\uparrow\uparrow$	0	$=$	-0.1	$\uparrow$	0	$=$
blackhole	-1000	$\uparrow\uparrow\uparrow$	-100	$\uparrow\uparrow\uparrow$	0	$=$	+50	$\downarrow\downarrow$
throttle	-500	$\uparrow\uparrow$	0	$=$	0	$=$	+25	$\downarrow$

TABLE I: Tactic cost/benefit on utility dimensions.

The impact on different quality attributes of the four tactics is shown in Table I. The number of upward or downward arrows is proportional to the magnitude of utility increments and decrements, respectively. For example, tactics *enlistServers* and *blackhole* cause a drastic reduction of response time (-1000 ms), which results in a much better utility in response time (U_R). Regarding the presence of malicious clients, tactic *blackhole* is the most effective one, whereas the other three tactics do not have any impact.

III. Background and Related Work

This section introduces some background on model checking of stochastic multi-player games (SMG), human involvement in self-adaptation, as well as related work on explanation for human involvement.

A. Model Checking Stochastic Multiplayer Games

Probabilistic model checking is a technique for formally modeling and analyzing systems that exhibit stochastic behavior, allowing quantitative reasoning about probability and reward-based properties (e.g., resource usage, time, etc.) [12]. Our approach to reasoning about explanation for human involvement in adaptation builds upon a recent technique for modeling and analyzing stochastic multiplayer games [13].

In our approach, systems are modeled as a turn-based SMG, meaning that in each state of the model, only one player can choose between several actions, the outcome of which can be probabilistic. Players can follow strategies to either cooperate to achieve the same goal, or compete to achieve their own (possibly conflicting) goals.

Reasoning about strategies enables checking for the existence of a strategy that is able to optimize an objective expressed as a quantitative property in a logic called rPATL [14], which extends ATL [15], a logic extensively used to reason about the ability of a set of players to collectively achieve a particular goal. Properties written in rPATL can state that a coalition of players $\langle\langle C\rangle\rangle$ has a strategy which can ensure that the probability of an event's occurrence $P_{\triangleright\triangleleft q}$ or an expected reward measure meet some threshold $R^r_{\triangleright\triangleleft x}$. Moreover, extended versions of the rPATL reward operator $\langle\langle C\rangle\rangle R^r_{max=?}[F^*\phi]$ and $\langle\langle C\rangle\rangle R^r_{min=?}[F^*\phi]$, enable the quantification of the maximum and minimum accrued reward r along paths that lead to states satisfying ϕ that can be guaranteed by players in coalition C, independently of the strategies followed by the rest of players. Model checking of rPATL properties supports optimal strategy synthesis for a given property.

B. Human Involvement in Self-Adaptation

Self-adaptive systems (innermost boundary labeled as "Machine" in Figure 1) were developed to autonomously adapt to changing circumstances. System dynamics (which results from the occurrence of certain events or changes in the environment state or target system –i.e., the system under control– state) is periodically monitored by a set of *sensors*. Given these sensor readings, the *controller* performs an analysis of available actions and their potential impact on the satisfaction of system goals based on the information available in the *knowledge base*, and plan corresponding adaptation decisions to be enacted via *actuators*.

The different activities in the feedback loop can benefit from human involvement in a variety of ways: Monitor can receive information from humans (acting as sophisticated sensors) that would be otherwise difficult to automatically monitor or analyze (e.g., humans can indicate whether there is an ongoing anomaly based on context information that is not captured by the models included in the knowledge base). The controller can incorporate inputs (e.g., recommendations, validation) into the decision-making process from application domain experts who can have additional insight about the best way of adapting the system. Execution can employ humans as system-level actuators to execute adaptations when changes to the system cannot be fully automated, or as a fallback mechanism. Beyond that, the role of the human can be supervisory, observing the activities carried out by the system and determining whether they are *appropriate* or potentially *erroneous* (e.g., likely to lead the system into an unsafe state, or degrade the satisfaction of system goals). In this paper, we focus on the case in which human operators act as sophisticated system-level actuators.

To capture the attributes of human agents that might affect interactions with the system, we employ the OWC [10] (opportunity-willingness-capability) model, which categorizes attributes into: 1) Opportunity: captures the applicability conditions of the actions that can be carried out by human actors upon the target system as constraints (e.g., is there an operator physically located on site?); 2) Willingness: captures transient factors that might affect the disposition of the operator to carry out a particular task (e.g., load, stamina, stress); and 3) Capability: captures the likelihood of successfully carrying out a particular task, which is determined by fixed attributes of the human actor, such as training level. OWC has been employed in several works that study human involvement in self-adaptation [16], [6], [4].

The activities generated by the autonomous machine will usually be interpreted by the human with respect to his model. When the machine model in a human's mind does not match reality, the machine is not behaving according to operator expectations. This phenomenon is known as *automation surprises* [17], [18]. Explanation is a mechanism that a machine can use to align a human's mental model of the system and its adaptive behavior with the machine's. This, in turn, can help to increase the willingness and capability of operators in performing different tasks assigned by the machine.

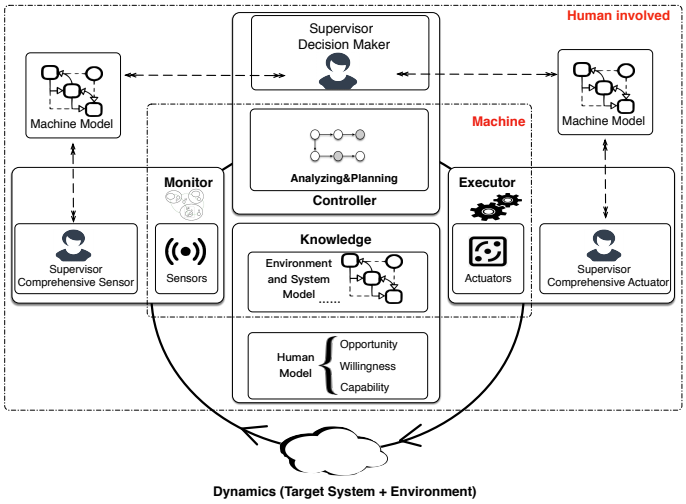


Fig. 1: Human Involvement in Self-Adaptation.

C. Explanation

For over a hundred years, many works in the fields of philosophy, social psychology, and cognitive psychology, have studied what constitutes an explanation, how they are structured, as well as how people generate and evaluate explanation [19]. Over three decades ago, there was extensive research on explanation in the context of expert systems [20]. Recently, the role of explanation has seen a resurgence in the field of artificial intelligence with the notion of eXplainable Artificial Intelligence (XAI) and in autonomous agents and robots as an important capability [19]. Currently, self-adaptive systems are being used as foundation to develop applications in many domains like autonomous driving, smart office and e-health. The work in [21] describes how the state of the machine is captured in a human's mind. When the behavior of the system is not explained, the state in mind may not be consistent with the real state, which could lead to dangerous situations. Also, the lack of a mental model for the human to estimate the actions of system may lead to safety risks [22], [23]. Therefore, explanation is necessary to support human-involved self-adaptive systems, as ratified by the General Data Protection Regulation (GDPR) law, which underlines the right to explanations [24]. Although some results from prior work can be be exploited by research on self-adaptive systems, such as explanation structures and explanation evaluation, new challenges arise, like understanding how timely explanation comprehension works, and its influence on human behaviors, which in turn affect adaptation behaviors.

The authors in [25] distinguish three explanation phases: explanation generation, explanation communication, and explanation reception. Explanation generation aims to generate two categories [26]: 1) "what-explanation", a description of the solution of a planning problem; 2) "why-explanation", a justification of why that particular solution is selected. Goal-based requirements models can be adopted to offer explanation of how a system is meeting its requirements [7]. Lin et

al. contributed an automatic explanation for the different explanation types and decision model types [27]. Explanation is also treated as different levels of reflective capabilities from forensic self-explanation to autonomous history-aware decision-making [28]. Elizalde et al. contributed an approach that identifies factors that are most influential to decision making with MDP [29]. Khan et al. present an approach for explaining an optimal action in a solution by counting the frequency of reaching a goal by taking the action [30]. Sukkerd et al. emphasized contrastive justification based on quality attributes and presented a method for generating an argument of how a solution is preferred to other rational alternatives [31].

These existing works either focus on the explanation content in the explanation generation stage or explanation effects on the human in the communication and reception stages. None of them, to the best of our knowledge, capture the explanation cost or the latency explanation induces in adaptation behavior. Our previous work [6], [4] applies an analysis technique based on model checking of SMGs to reason about human involvement without any explicit consideration about the role of explanation in such involvement. In this work, we build on this analysis technique, which is particularly suitable for our purposes since it enables reasoning quantitatively about trade-offs and under uncertainty about "when" explanations should be provided. Other prior work has investigated the role of explanation as a mechanism to improve the understanding of human agents when they are involved in a supervisory role, either approving or rejecting the autonomous system's proposed course of action [32]. In contrast, the aspect of explanation analyzed in this paper is in the context of a human operator carrying out tasks for the system, where the main trade-off analyzed is between the potential improvement due to increased levels of willingness and capability, and the cost incurred by adaptation delays caused by the latency of explanation comprehension.

IV. Reasoning about Explanation for Human-in-the-loop Adaptation

Deciding whether explanations should be provided for human-involved self-adaptive systems is not an easy task, since their influence on humans can be affected by factors such as the level of expertise in carrying out particular tasks. Also, the cost incurred by the explanation may reduce the satisfaction of systems goals due to, for example, the delayed adaptation behaviors and human confusion. These factors build as additional sources of uncertainty affecting the behavior of the self-adaptive system [33].

In this context, we want to answer the following questions: (Q1) How can the outcome of adaptation with explanation be predicted for human-involved adaptive systems? and (Q2) How can a self-adaptive system determine whether explanation should be provided to a human operator in a given situation?

The key idea of our approach to enable automated reasoning about explanation is to: (i) consider it as one of the possible actions or *tactics* that the system can enact as part of an adaptation strategy, and (ii) leave the choice of using such explanation tactics under-specified in the SMG model (encoded as a non-deterministic choice). The probabilistic model checker PRISM-games [34] is then used to analyze the model, resolving the nondeterminism to produce a strategy that maximizes the expected utility by selecting when explanation should be given (i.e., at which choices in the model explanation tactics should be used).

Our system is modeled as a SMG with three players, illustrated in Listing 1. Player *env* is in control of all the (asynchronous) actions that the environment can take out of system's and human's control while player *ope* specifies the actions controlled by the human actor. Player *sys* controls the actions that belong to the system, whose behavior is encoded in the processes *ha_system* (which captures the behavior of the system under control), as well as *Outgun*, *Eliminate* and *Eliminate_Explanation*, which are processes that capture different adaptation strategies of the controller for absorbing, eliminating and eliminating with explanation excess traffic, respectively. Moreover, the system player controls the synchronization of actions between adaptation strategies, the target system and the human operator, thus modeling the triggering of adaptation tactics. The global variable *turn* (line 5) is used to explicitly encode alternating turns between the environment, system and operator players. The following subsections present the models of the three players.

```
1 player sys
      ha_system, Outgun, Eliminate, Eliminate_Explanation,
      [enlistServer], [lowerFidelity], [explain_bA], [blackholeAttacker],
      [explain_tS], [throttleSuspicious] endplayer
2 player env environment endplayer
3 player ope operator endplayer
4 const ENVT=0; const SYST=1; const OPER =2;
5 global turn:[ENVT..OPER] init ENVT;
```

Listing 1: Player Definition for the Znn.com SMG.

The environment process *env* (Listing 2) models potential evolution of environment variables. For simplicity, we assume our environment model only keeps track of time, although additional behavior controlling other elements (e.g, network delay) can be encoded (please refer to [35] for further details illustrating the modeling of adversarial environments in turn-based SMGs). Variable *t* (line 2) keeps track of execution time (the time frame for the system's execution is determined by [0, *MAX_TIME*]). During its turn, the environment checks that the end of the time frame for the execution has not been reached yet, and if that is the case, it increments the value of *t* one unit, yielding the turn to the system player (line 3).[1]

[1]We model the SMG using the syntax of the PRISM language, encoded as commands $[action]\ guard \rightarrow p_1 : u_1 + ... + p_n : u_n$ Where guard is a predicate over the model variables. Each update u_i describes a transition that the process can make (by executing action) if the guard is true. An update is specified by giving the new values of the variables, and has an assigned probability $p_i \in [0, 1]$. Multiple commands with overlapping guards (and probably, including a single update of unspecified probability) introduce local nondeterminism.

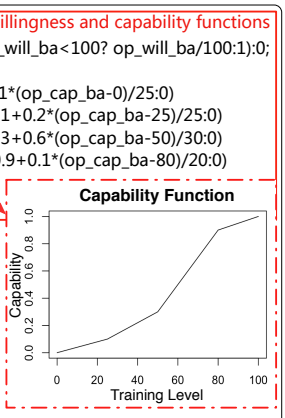
```
1  module environment
2      t : [0..MAX_TIME] init 0;
3      [          ] (turn=ENVT) & (t<MAX_TIME) -> (t'=t+1) & (turn'=SYST);
4  endmodule
```

Listing 2: Environment Model.

A. Human Model

Listing 3 shows the encoding of behaviors corresponding to an operator in the Znn.com system. Opportunity elements (lines 4-5) are used to guard the execution of tactics such as *explain_ba* and *blackhole* (lines 12 and 15) in the process. Willingness (line 1) is initialized with some constants (line 6) that represent the initial willingness a human has based on his current state. Capability denotes the likelihood of successfully carrying out a particular task, such as training level (line 7) and is defined by an explicit set of value pairs (with intermediate points linearly interpolated in line 2) as shown in the dotted red rectangle.

```
                                                  Willingness and capability functions
1  formula op_f_w_ba = op_will_ba>0? (op_will_ba<100? op_will_ba/100:1):0;
2  formula op_f_c_ba =
       (op_cap_ba>0 &op_cap_ba<=25? 0.1*(op_cap_ba-0)/25:0)
     +(op_cap_ba>25&op_cap_ba<=50? 0.1+0.2*(op_cap_ba-25)/25:0)
     +(op_cap_ba>50&op_cap_ba<=80? 0.3+0.6*(op_cap_ba-50)/30:0)
     +(op_cap_ba>80&op_cap_ba<=100?0.9+0.1*(op_cap_ba-80)/20:0)
     +(op_cap_ba>100? 1:0);
   .....
3  module operator
4    op_onLocation: bool init true;
5    op_busy:bool init false;
6    op_wil_ba: [0..100] init INIT_WIL;
7    op_cap_ba:[0..100] init INIT_TRAIN;
8    exp_ba_done: bool init false;
9    cnt_compExpba : [0..MAX_TIME] init 0;
10   ba_done:bool init false;
11   cnt_ba : [0..MAX_TIME] init 0;
     .....
12   [explain_ba] (cnt_compExpba = 0) & (op_onLocation) &
                  (!op_busy) & (exp_ba_done= false)
               -> (op_busy'= true) & (cnt_compExpba'=1);
13   [             ] (turn=OPER) & (cnt_compExpba>0) &
                  (cnt_compExpba<=exp_ba_latency)
               -> (cnt_compExpba'=cnt_compExpba+1) & (turn'=ENVT);
14   [             ] (turn=OPER) & (cnt_compExpba=exp_ba_latency+1)
               -> (exp_ba_done'= true) & (cnt_compExpba'=0) &
                  (op_cap_ba'=min((cap_ba+DELTA_CAP),100)) &
                  (op_wil_ba'= min((will_ba+DELTA_WIL),100))&(turn'=ENVT)
                  &(op_busy'= false);          Blackhole Explanation Tactic
15   [ blackhole ] (op_onLocation) & (!op_busy) &
                          (cnt_ba=0) & (ba_done= false)
               -> (cnt_ba'=1) & (op_busy'=true);
16   [             ] (turn=OPER) & (cnt_ba>0) & (cnt_ba<=ba_latency)
               -> (cnt_ba'=cnt_ba+1) & (turn'=ENVT);
17   [             ] (turn=OPER) & (cnt_ba=ba_latency+1)
               -> (cnt_ba'=0)&(turn'=ENVT)&(op_busy'= false)&
                  (ba_done'= true);            Blackhole Tactic
     .....
18   [             ] (turn=OPER) & (cnt_ba=0) & (cnt_compExpba=0) & ...
               -> (turn'= ENVT);
19 endmodule
```



Listing 3: Human Model.

Variables *cnt_compExpba* and *cnt_ba* are counters used to keep track of the latency of tactics *explain_ba* (i.e., the time needed to comprehend the explanation with respect to blackholing tactic) and *blackhole* (i.e., the time to manually block traffic from malicious clients) respectively while *exp_ba_done* and *ba_done* indicate the end of tactics execution in avoid of tactic repetition. Moreover, the module includes commands that model the effect of executing the different tactics as updates on its variables. In particular, there are three different commands per tactic in the module. We focus on tactic *explain_ba* to illustrate how tactic execution is modeled:

- Tactic trigger (line 12) Triggers tactic execution when: (i) an operator is on location and not busy, (ii) the tactic has not been executed before, and (iii) the latency counter for the tactic is zero, meaning that the tactic is not being executed. As a consequence, the operator is flagged as busy and the latency counter is activated (i.e., $cnt\_compExpba' = 1$);
- Tactic latency counter update (line 13). If the tactic counter is active, but still has not reached the tactic's latency value, the counter is incremented in one unit;
- Tactic completion (line 14). When the tactic's latency counter expires, the command updates variables *op_cap_ba* and *op_wil_ba* according to the encoding of the impact of the explanation on the willingness and capability of the designated human operator. The latency counter and task completion identification is reset, and the busy status of the operator is set to false.

Ordinarily an explanation will motivate the operator with more willingness to complete the task and reduce the probability of making mistakes when identifying the attackers, denoted as the effect of *DELTA_WIL* and *DELTA_CAP*. Correspondingly, the value of the formula in lines 1-2 is updated to reflect these changes, which affect the probabilistic behavior of the system, described in Section IV-A.

The encoding used for the tactic *blackhole* (lines 15-17) follows the same structure. Every command in this module, except two with synchronized actions (line 12 and 15) initialized by the system, includes a predicate in the guard to ensure that the command is triggered only during the operator player's turn (*turn=OPER*), and an additional predicate in the post state that yields the turn to the environment player (*turn'=ENVT*). Moreover, an additional command (line 18) lets the process progress without any variable updates when none of the latency periods for the tactics are active. Note that in our model, we assume sequential execution of tactics[2].

B. System Model

The system behavior module (Listing 4) incorporates a collection of variables encoding the different system qualities of concern, as well as the aspects relevant to the applicability conditions of tactics.Variables *rt*, *as*, *mc*, and *ua* encode the response time, number of active servers, percentage of malicious clients, and level of user annoyance in the system, respectively and lines 6-9 illustrate how the different variables are initialized. Variable *ba_fail* (line 10) denotes the failure of

[2]We do not describe the commands corresponding to tactics *explain_ts* and *throttle* related to human operator in Listing 3 for the sake of clarity.

performing tactic *blackhole*. The encoding used for the tactic related to the system module follows the same structure and is quite similar to those described in operator module.

```
1  formula ba_f_rt =                                  Blackhole Tactic Impact on Utility Dimensions
     rt-1000*op_f_c_ba >=0 ? ( rt-1000*op_f_c_ba <= MAX_RT ?
     ( floor(rt-penalty_ba*1000*op_f_c_ba) : MAX_RT) : 0);
2  formula ba_f_mc = mc-op_f_c_ba*mc >= 0 ? ( mc-op_f_c_ba <= 100 ?
     ( floor(mc-penalty_ba*op_f_c_ba*mc):100):0;
3  formula ba_f_ua = ua+50>=0 ? (ua+50<=100 ? ua+50:100):0;
4  formula penalty_ba = pow(0.99,pow(exp_ba_latency,2));
5  module ha_system
6     as : [0..MAX_SERVERS] init init_as;
7     rt : [0..MAX_RT] init init_rt;
8     mc : [0..100] init init_mc;
9     ua : [0..100] init init_ua;
10    ba_fail : bool init false;
      ...
11    [explain_ba] (turn=SYST) & (cnt_compExpba=0)&(exp_ba_done= false)
                -> (turn'=OPER);
12    [             ] (turn=SYST) & (cnt_compExpba>0) &
                    (cnt_compExpba <= exp_ba_latency)
                -> (turn'=OPER);
13    [             ] (turn=SYST) & (cnt_compExpba = exp_ba_latency+1)
                -> (turn'=OPER);                    Blackhole Explanation Tactic
14    [blackhole] (turn=SYST) & (cnt_ba=0) & (ba_done= false)
                -> (turn'=OPER);
15    [             ] (turn=SYST) & (cnt_ba>0) & (cnt_ba<=ba_latency)
                -> (turn'=OPER);
16    [             ] (turn=SYST) & (cnt_ba=ba_latency+1)
          -> op_w_prob_ba :
              (rt'=ba_f_rt) & (mc'=ba_f_mc) & (ua'=ba_f_ua) & (turn'=OPER)
            1-op_w_prob_ba :
              (ba_fail'=true) & (turn'=OPER);           Blackhole Tactic
      ...
17    [             ] (turn=SYST) & (cnt_ba=0) &... -> (turn'=OPER);
18 endmodule
```

Listing 4: System Model.

As shown in *ha_system* module, tactics trigger when: (i) the tactic has not been executed before, and (ii) the latency counter for the tactic is zero. Since the commands (lines 11, 14) are synchronized with the commands for human operator in listing 3, the conditions/guards should also satisfy such as human operator on location and not busy to simultaneously trigger both commands. When the latency counter of tactic *blackhole* expires (line 16), the command can (1) update variables *rt*, *mc* and *ua* with willingness probability *op_w_prob_ba* from the human model with respect to a successful activation of tactic shown in Table I (defined as formulas in lines 1-3), discounted by the capability probability *op_c_prob_ba* and penalty *penalty_ba* that is positively correlated with explanation latency (line 4); or (ii) fail to blackhole malicious attacker with probability 1-*op_w_prob_ba*, flagging the failure on variable *ba_fail*. Beside, we assume penalty is applicable to both variables *rt* and *mc* with positive tactic impact as we model the fact that a small periods of latency with high response time and malicious clients are tolerable while longer periods are much worse.

Unlike *blackhole*, the tactic *explain_ba* will only affect the willingness and capability of a human operator not directly the system variables (line 13). The encoding used for the *enlistServers* tactic and *lowerFidelity* follows the same structure of tactic *blackhole*, but without any OWC elements encoded in the guards or updates of the commands. We do not describe these code in listing 4 for the sake of clarity.

Module Eliminate (Listing 5) models the strategy to eliminate excess traffic with the help of a human operator. It first notifies an operator to manually block traffic from malicious clients. The command on line 3 encodes the triggering of tactic *blackhole*, which sets the value of the timestamp variable *ba_trigger_t* that indicates at which time point the tactic was triggered. If the settling time expires (i.e., $t > trigger\_t + ba\_latency$) and the intended effect of the tactic is not observed (i.e., ba_fail), the strategy notifies another operator to execute the *throttle* tactic as a fallback, throttling suspicious clients. Module *Eliminate_Explanation* follows a similar structure except explicitly providing the explanation as a tactic *explain_ba* and *explain_ts* before assigning tasks to the human, while module Outgun models the automatic strategy to absorb excess traffic in Znn.com.

```
1  module Eliminate
2     trigger_t:[0..MAX_TIME] init 0;
3     [blackhole] (turn=SYST) -> (trigger_t'=t);
4     [ throttle  ]  (turn=SYST) & (ba_fail) & (t>trigger_t+ba_latency)
                -> true;
5  endmodule
6  module Eliminate_Explanation
7     trigger_t:[0..MAX_TIME] init 0;
8     [explain_ba] (turn=SYST) -> (trigger_t'=t);
9     [ blackhole]  (turn=SYST) & (t>trigger_t+exp_ba_latency)
                ->  (trigger_t'=t);
10    [ explain_ts] (turn=SYST) & (ba_fail) & (t>trigger_t+ba_latency)
                -> (trigger_t'=t);
11    [   throttle  ] (turn=SYST) & (ba_fail) & (t>trigger_t+exp_ts_latency)
                ->  true;
12 endmodule
13 module OutGun
      ...
14 endmodule
```

Listing 5: Strategy Model.

C. Utility Profile

Utility functions and preferences are encoded using formulas and reward structures that enable the quantification of the utility of a given game state. Formulas compute utility on the different dimensions of concern, and reward structures weigh them against each other by using the utility preferences of a given scenario. Listing 6 illustrates in line 1 the encoding of utility functions using a formula for linear interpolation based on the points defined for utility function UM from the domain knowledge. Lines 2-5 show how a reward structure can be defined to compute a single utility value for any state by using the utility preferences defined for a particular scenario.

V. Analysis Results

In this section, we demonstrate how our approach can produce decisions about when to involve humans in adaptation and when to provide them with explanations in a DoS scenario. In particular, we exploit SMG models of human-involved adaptation to determine: (i) the expected outcome of human involvement in adaptation, (ii) the conditions under which

```
1  formula uM = (mc>=0 & mc <=5? 1 : 0)
     +(mc>5  & mc <=20? 1+(0.80-1)*((mc-5)/(20-5)) :0)
     +(mc>20 & mc <=50?0.80+(0.40-0.80)*((mc-20)/(50-20)) :0)
     +(mc>50 & mc <=70? 0.40+(0.00-0.40)*((mc-50)/(70-50)) :0)
     +(mc>70 ? 0:0);
   ...
2  rewards "rGU"
3     turn = SYST & scenario=1: 0.15*uR+0.6*uM +0.1*uC+0.15*uA;
4     turn = SYST & scenario=2: 0.3*uR+0.3*uM +0.1*uC+0.3*uA;
      ...
5  endrewards
```

Listing 6: Utility Reward Structure for Znn.com SMG.

explanation as a tactic could improve the expected outcome of human involvement, and (iii) the conditions under which human involvement facilitated by explanation improves over fully automated adaptation or human involvement without explanation.

To explore our scenario, we statically analyze a discretized region of the state space of our problem. Each state of the discrete set requires a run of the model checker per adaptation alternative (i.e., automated, human-involved without explanation, human-involved with explanation) that quantifies the accrued system utility encoded in a temporal logic formula. For PRISM-Games, this property is expressed in rPATL as $u_{mau} \equiv R^{rGU}_{max=?}[F\ end]$ where "rGU" is the reward structure specified in listing 6, and *end* is a predicate that indicates the end of the adaptation decision-execution period.

Once we have quantified the utility of every alternative in each state of the discrete space S, we rank the different adaptation alternatives and select the one that maximizes the expected accrued utility. Hence the selected adaptation strategy for a state $s \in S$ from a set of strategy alternatives Γ can be determined according to: $\gamma_{\uparrow}(s,\Gamma) \triangleq arg\max_{\gamma\in\Gamma} u_{mau}\ (s,\gamma)$ where $u_{mau}(s,\gamma)$ is the value of property u_{mau} evaluated in a model instantiated with with an initial state s and the adaptation logic of strategy γ.

A. Experimental Results

We analyze our results in two scenarios that differ in the priority given to the different concerns: eliminating malicious clients (Scenario 1) and optmizing the experience of legitimante clients (Scenario 2).

Scenario 1. Figure 2 depicts utility analysis results for the different adaptation strategies in a DoS scenario in which the priority is eliminating malicious clients. The initial state of the scenario corresponds to, a response time 2000 ms, 0% of user annoyance, and 2 active servers. The overall time frame for the scenario is 30 minutes, and the latency value employed for tactics *blackhole* and *throttle* is 10 minutes. This tactic latency models the time that the human operator requires to decide (without an explanation being provided) which clients have to be blackholed or throttled. Note that tactic latency is different from the latency of explanation comprehension, which will be introduced in the following.

Plots 2 (a) and (b) illustrate the maximum accrued utility the system can achieve for varying initial levels of malicious clients and training of a human operator with strategies *Eliminate* and *OutGun*, respectively. As expected, the utility for *OutGun* is not affected by the level of training of the human operator because the tactics employed by the strategy are fully automated. Moreover, the utility decreases progressively with increasing levels of malicious clients as strategy *OutGun* employs tactics (e.g., adding server or lowering fidelity) that do not deal with malicious users. In contrast, we can observe how *Eliminate* yields significantly better results when both the percentage of malicious clients and training level are high. On the contrary, the utility gap between *OutGun* and *Eliminate* is small when the percentage of malicious clients is low as there are few or no malicious clients to deal with, and when the operator is poorly trained and is less efficient at reducing malicious clients, increasing user annoyance in legitimate clients who might be incorrectly blacklisted or throttled.

Plot 2 (c) shows the delta in utility between strategy *Eliminate_Explanation* (which includes explanation as potential tactics) and strategy *Eliminate* without explanation. Explanation latency (i.e., the average time that it takes for the human operator to go through and understand the explanation) is fixed to a value of 4 minutes for explanation tactics *explain_ba* and *explain_ts*. The effectiveness of the explanation tactics (i.e., their influence on willingness and capability encoded by *DELTA_CAP* and *DEL_WIL* in our model, respectively) is fixed to 30%. Plot 2 (e) is analogous to (c), but explanation latency is fixed to a value of 7 minutes.

These two plots illustrate how explanation affects the expected outcome of adaptation for strategy *Eliminate*. In particular, we observe that the impact is negligible or even negative when the operator is poorly or well trained. This is because an experienced operator (i.e., training level above 80%) is already close to the top of her capability, whereas a novice (i.e., training level below 30%) obtains modest improvement in capability in the low-medium training level range as shown in capability function at Listing 3. However, the improvement obtained from explanation tactics in situations where the operator training level is around 50% is significant due to the higher capability-training ratio in the range 50-80% (cf. capability function in Listing 3). Moreover we observe that the delta in utility also improves progressively with increasing levels of malicious clients. This is consistent with the fact that such situations require a better trained operator to successfully identify malicious IP address ranges.

Comparison of plots (c) end (e) also reveals that higher explanation latency is detrimental to obtaining better system utility, as one might expect. Concretely, the delta in plot 2 (c) is always above positive with 4 minutes explanation latency and could be as high as 20. However, the maximum delta for with 7 minutes of explanation latency (e) halves with respect to that with lower latency. In fact, the delta can even become negative when the operator is well trained. This also aligns with the intuition that a well trained operator is already effective at dealing with malicious clients regardless of the explanation provided, which is only going to delay enacting adaptation and reflect negatively on utility.

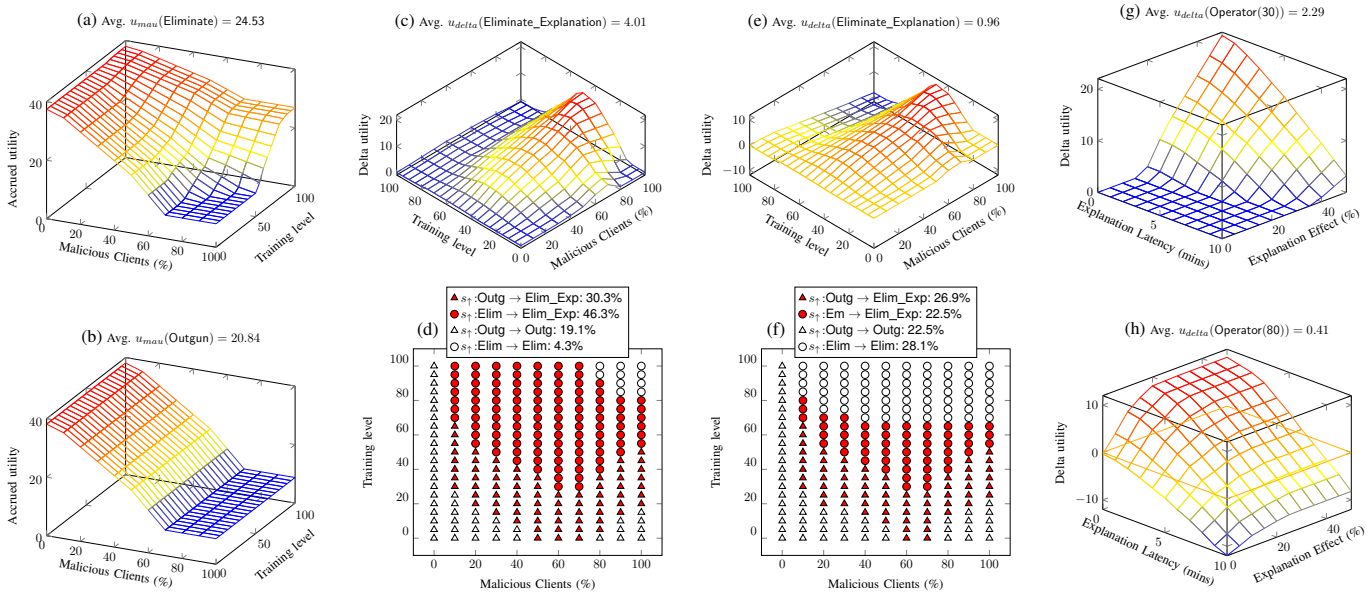


Fig. 2: Results for Scenario 1: minimizing percentage of malicious clients

Plots 2(d) and (f) present the results of strategy selection among *OutGun*, *Eliminate* and *Eliminate_Explanation* that correspond to the two values of explanation latency employed in the two upper plots. Without explanation, the states in which human involvement is chosen are indicated by a circle, whereas the automated strategy is indicated by a triangle mark. The plots show that higher levels of malicious clients make human involvement preferable even when the training level of the operator is limited to the range 30-50. However, it is also worth noting that when the training level is very low, the improvement by *OutGun* can outweigh the moderate improvement provided by *Eliminate*. This can be observed in the area where the training level is below 40 and the malicious clients are in the range 80-100.

Regions of the space in which explanation is selected are colored in red, with triangles and circles denoting states in which *OutGun* and *Eliminate* have been replaced by *Eliminate_Explanation*, respectively. The plots reveal the areas where explanations improve the expected outcome of adaptation (in red) and show how the areas where human involvement can outperform automated adaptation are broadened (red triangles). The white circles denote explanation being detrimental due to its cost, such as the points in the upper right corner in plot (d) where the well-trained operator should act quickly in serious cases with near 100% malicious clients, instead of waiting for an ineffective explanation. Also, an experienced operator ($\geq 80\%$ training level) should not wait for explanation at all if the latency is up to 7 minutes as shown in (f) as the comprehension delays of explanation will outweigh its benefit.

To provide further context about under what conditions an explanation should be provided, we present two plots that examine the trade-off between explanation cost and effect. Explanation cost is associated here with the explanation comprehension latency and effect is the improvement on the capability of the operator with a fixed 20% willingness improvement. Our analysis assumes a penalty on tactic effect directly proportional to the latency value.

Plot 2 (g) represents a novice with initial training level 30 while (h) is for an experienced expert with 80. Plot (g) clearly shows that the delta utility with respect to strategy *Eliminate* without explanation increases progressively with higher values of explanation effect and decreasing explanation latency. It is also worth noting that the delta will not become negative even if the tactics are enacted poorly and late. This is because the reference (i.e., utility for Eliminate) is also low due to the low capability. Things are totally different for the expert. The delta utility is decreasing down to -10 as the explanation latency reaches 10 minutes. This aligns with the intuition that trying to make an expert understand something that she already knows is just a waste of time and is only going to degrade the overall system utility. Moreover, the delta utility improves with the explanation effect ranging from 0-20, but with no improvement beyond that range because the operator is already at the top of her capability.

Scenario 2. Figure 3 shows the analysis results in Scenario 2, in which the top priority is optimizing the experience of legitimate clients. Plot (b) shows how strategy Outgun still experiences a reduction in the utility with increasing levels of malicious users (similarly to Scenario 1). However, in this scenario the reduction in utility and the gap between the human-driven and the automated strategy is less pronounced than in Scenario 1 because in this case the main contribution to utility results from optimizing legitimate client experience, and efficiency at reducing the percentage of malicious clients is not as relevant. The delta utility with an explanation is also

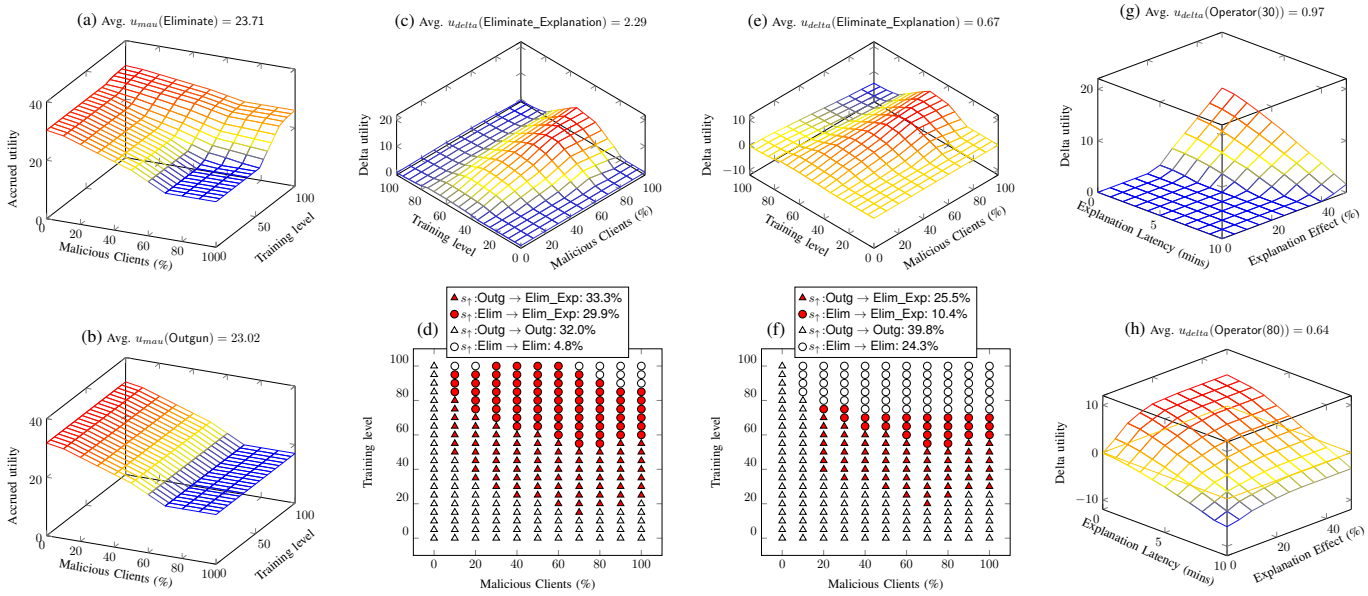


Fig. 3: Results for Scenario 2: optimizing experience of legitimate clients

selected in less situations as shown in plots 3 (d) and (f) (i.e., red points including circles and triangles accounting for 63.2% and 35.9% respectively compared to 76.6% and 49.4% in Figure 2). Despite this reduction, explanation can still broaden the range of situations under which the human-involved strategy is selected (i.e., red triangles). Figure 3 (g) and (h) follow the same trend of Scenario 1, although the maximum and minimum delta utility obtained with explanation is reduced to almost half of its magnitude in Scenario 1 in most cases. This is again explained by the fact that the top priority in Scenario 2 is optimizing experience of legitimate clients and not dealing effectively with malicious clients that does not detract from utility as much as in Scenario 1.

In summary, the results of analyzing these two scenarios have shown that: (i) explanation can enhance the performance of the system when involving human operators, especially when they have intermediate training levels, (ii) incurring the cost of explanation is not valuable, and can even be detrimental effects when operators are close to the top of their capacity, and (iii) although different priorities result in variations in magnitude of the increment of system utility and the range of situations under which human-driven adaptation is selected, explanation improves system utility across large regions of the state space.

VI. Conclusions and Discussion

In this paper we presented an approach based on probabilistic model checking to determine when an explanation should be selected as an adaptation tactic in a human-involved self-adaptive system. Although one limitation of our approach is that we did not directly correlate our analytical results with actual systems through an empirical study, our findings are supported by and consistent with those obtained by Sukkerd et al [36], who conducted an empirical user study to determine if explanations can help users gain more understanding of the rationale behind system decisions and become more likely to cooperate with the system once they understand the need to act, thus increasing user willingness. Their experimentation also supports our model, which assumes that explanation increases operator awareness of the situation and capability [36].

A second limitation is the lack of a direct procedure for obtaining values that characterize the effect of explanation on willingness and capability (i.e., *DELTA_WIL* and *DELTA_CAP*). While this is an area for future work, prior work suggests that the impact on willingness can be obtained through unobtrusive measurement of human mental states with, e.g., facial expression [5], while the delta in capability can be traced through historical behaviors [27]. Moreover, sensitivity analysis can also be useful when the explanation effect cannot be determined with precision but lies within a known range. Another limitation, and also a topic for future work, is that comprehension time for explanation is not easy to predict across different operators. One way to overcome this challenge is by assigning the latency based on the complexity of information in the explanation content, e.g., the amount of the information. Qualitative estimates of time for the operator to understand explanation could also be explored [27].

Our initial investigation suggests a number of additional research directions, such as formulating the problem as a multi-objective optimization with Pareto-optimal solutions (in contrast with a single utility function which is a linear combination of different contributions, as in this paper); tailoring the optimal amount of information to be provided as an explanation candidate to maximize overall utility; taking the time delay of explanation comprehension in consideration and anticipating human-involved adaptation to proactively

make decisions about explanation; and finally, analyzing and planning strategies by explicitly considering explanation and automated tactics as potentially concurrent adaptation tactics.

REFERENCES

[1] B. H. C. Cheng and et al., "Software engineering for self-adaptive systems: A research roadmap," in *Software Engineering for Self-Adaptive Systems. Lecture Notes in Computer Science, vol 5525. Springer, Berlin, Heidelberg*.

[2] R. de Lemos and et al., "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II. Lecture Notes in Computer Science, vol 7475. Springer, Berlin, Heidelberg*.

[3] R. Sukkerd, D. Garlan, and R. G. Simmons, "Task planning of cyber-human systems," in *Software Engineering and Formal Methods - 13th International Conference, SEFM 2015, York, UK, September 7-11, 2015. Proceedings*, 2015, pp. 293–309.

[4] J. Cámara, G. A. Moreno, and D. Garlan, "Reasoning about human participation in self-adaptive systems," in *10th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS, Florence, Italy, May 18-19, 2015*, 2015, pp. 146–156.

[5] E. Lloyd, S. Huang, and E. Tognoli, "Improving human-in-the-loop adaptive systems using brain-computer interaction," in *12th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2017, Buenos Aires, Argentina, May 22-23, 2017*, 2017, pp. 163–174.

[6] "Chapter 7 - evaluating trade-offs of human involvement in self-adaptive systems," in *Managing Trade-Offs in Adaptable Software Architectures*, I. Mistrik, N. Ali, R. Kazman, J. Grundy, and B. Schmerl, Eds. Boston: Morgan Kaufmann, 2017, pp. 155 – 180.

[7] K. Welsh, N. Bencomo, P. Sawyer, and J. Whittle, "Self-explanation in adaptive systems based on runtime goal-based models," *Trans. Computational Collective Intelligence*, vol. 16, pp. 122–145, 2014.

[8] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," in *IJCAI-17 workshop on explainable AI (XAI)*, vol. 8, 2017, p. 1.

[9] T. Nomura and K. Kawakami, "Relationships between robot's self-disclosures and human's anxiety toward robots," in *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03*. IEEE Computer Society, 2011, pp. 66–69.

[10] D. Eskins and W. H. Sanders, "The multiple-asymmetric-utility system model: A framework for modeling cyber-human systems," in *Eighth International Conference on Quantitative Evaluation of Systems, QEST 2011, Aachen, Germany, 5-8 September, 2011*, 2011, pp. 233–242.

[11] S. Cheng, D. Garlan, and B. R. Schmerl, "Evaluating the effectiveness of the rainbow self-adaptive system," in *2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2009, Vancouver, BC, Canada, May 18-19, 2009*, 2009, pp. 132–141.

[12] M. Kwiatkowska, G. Norman, and D. Parker, *Probabilistic Model Checking: Advances and Applications*. Cham: Springer International Publishing, 2018, pp. 73–121.

[13] A. Simaitis, "Automatic verification of competitive stochastic systems," Ph.D. dissertation, University of Oxford, UK, 2014. [Online]. Available: http://ora.ox.ac.uk/objects/uuid:68b5e2d8-ba04-419f-8926-4cd542121e2d

[14] T. Chen and J. Lu, "Probabilistic alternating-time temporal logic and model checking algorithm," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007, 24-27 August 2007, Haikou, Hainan, China, Proceedings, Volume 2*, 2007, pp. 35–39.

[15] R. Alur, T. A. Henzinger, and O. Kupferman, "Alternating-time temporal logic," *J. ACM*, vol. 49, no. 5, pp. 672–713, 2002.

[16] M. Gil, M. Albert, J. Fons, and V. Pelechano, "Designing human-in-the-loop autonomous cyber-physical systems," *Int. J. Hum. Comput. Stud.*, vol. 130, pp. 21–39, 2019.

[17] S. Combéfis, D. Giannakopoulou, C. Pecheur, and M. Feary, "Learning system abstractions for human operators," in *MALETS Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*, 2011, pp. 3–10.

[18] E. Palmer, "Oops, it didn't arm. - a case study of two automation surprises." in *Proceedings of the 8th International Symposium on Aviation Psychology*, 1996, pp. 227–232.

[19] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, 2019.

[20] B. Chandrasekaran, M. C. Tanner, and J. R. Josephson, "Explaining control strategies in problem solving," *IEEE Expert*, vol. 4, no. 1, pp. 9–24, 1989.

[21] T. Hellström and S. Bensch, "Understandable robots-what, why, and how," *Paladyn, Journal of Behavioral Robotics*, vol. 9, no. 1, pp. 110–123, 2018.

[22] C. L. Bethel, "Robots without faces: non-verbal social human-robot interaction," Ph.D. dissertation, 2009. [Online]. Available: https://scholarcommons.usf.edu/etd/1855

[23] J. Broekens, M. Harbers, K. Hindriks, K. Van Den Bosch, C. Jonker, and J.-J. Meyer, "Do you get it? user-evaluated explainable bdi agents," in *German Conference on Multiagent System Technologies*. Springer, 2010, pp. 28–39.

[24] P. Carey., *Data protection: a practical guide to UK and EU law.* Oxford University Press, Inc., 2018.

[25] M. A. Neerincx, J. van der Waa, F. Kaptein, and J. van Diggelen, "Using perceptual and cognitive explanations for enhanced human-agent team performance," in *Engineering Psychology and Cognitive Ergonomics - 15th International Conference, EPCE 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings*, 2018, pp. 204–214.

[26] R. Sukkerd, "Improving transparency and understandability of multi-objective probabilistic planning," *Thesis Proposal – School of Computer Science Institute for Software Research Software Engineering, Carnegie Mellon University*, pp. 1–41, 2018.

[27] B. Y. Lim, A. K. Dey, and D. Avrahami, "*Why and why not* explanations improve the intelligibility of context-aware intelligent systems," in *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, 2009, pp. 2119–2128.

[28] A. García-Domínguez, N. Bencomo, J. M. P. Ullauri, and L. H. G. Paucar, "Towards history-aware self-adaptation with explanation capabilities," in *IEEE 4th International Workshops on Foundations and Applications of Self* Systems, FAS*W@SASO/ICCAC 2019, Umea, Sweden, June 16-20, 2019*, 2019, pp. 18–23.

[29] F. Elizalde, L. E. Sucar, M. Luque, J. Diez, and A. Reyes, "Policy explanation in factored markov decision processes." in *In Proc European Workshop on Probabilistic Graphical Models (PGM)*, 2008, pp. 97–104.

[30] O. Z. Khan, P. Poupart, and J. P. Black, "Minimal sufficient explanations for factored markov decision processes," in *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*, 2009.

[31] R. Sukkerd, R. G. Simmons, and D. Garlan, "Towards explainable multi-objective probabilistic planning," in *Proceedings of the 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems, ICSE 2018, Gothenburg, Sweden, May 27, 2018*, 2018, pp. 19–25.

[32] N. Li, S. Adepu, E. Kang, and D. Garlan, "Explanations for human-on-the-loop: A probabilistic model checking approach," in *Proceedings of the 15th International Symposium on Software Engineering for Adaptive and Self-managing Systems (SEAMS)*, 2020, to appear.

[33] N. Esfahani and S. Malek, "Uncertainty in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, 2010, pp. 214–238.

[34] M. Z. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification - 23rd International Conference, CAV ,July 14-20, 2011. Proceedings*, 2011, pp. 585–591.

[35] J. Cámara, G. A. Moreno, and D. Garlan, "Stochastic game analysis and latency awareness for proactive self-adaptation," in *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, Proceedings, Hyderabad, India, June 2-3, 2014*, 2014, pp. 155–164.

[36] R. Sukkerd, R. Simmons, and D. Garlan, "Tradeoff-Focused Contrastive Explanation for MDP Planning," *arXiv e-prints*, p. arXiv:2004.12960, Apr. 2020.