Robustness For Protection Envelopes with Respect to Human Task Variation

Ayesha Yasmeen and Elsa L. Gunter
Department of Computer Science
University of Illinois
Urbana, IL 61801
Email: yasmeen@illinois.edu, egunter@illinois.edu

***Abstract*—Safety critical systems can suffer severe and even fatal consequences due to aberrant behavior of human operators. Human operators are unique in their decision making capability, judgment and nondeterminism. There is a need for analyzing the interactions among computer systems and human operators where the operators are allowed to deviate from their prescribed behaviors for executing a task. In this paper we wish to examine the ability of a system to remain safe under broad classes of variations of the prescribed human task. To facilitate this concept we consider the concept of a protection envelope giving a wider class of behaviors than strictly prescribed by the human task while providing guarantees of restrictions on human operator to the system. We develop methods for addressing two issues. The first issue is: given a human task specification and a protection envelope, will the protection envelope properties still hold under standard variations as described by Hollnagel [10]. The second issue is: in the absence of a protection envelope, can we approximate a protection envelope that will at least have the property of being robust against the aforementioned variations. We present methodology and tool for assisting in this regard.**

***Index Terms*—task analysis, human operator, system robustness, protection envelope, human action variations**

I. Introduction

Computerized automated systems are omnipresent in today's world. Everyday humans interact with computers in almost every sector of life: through using personal computers at home; serving as operators for computer systems in the workplace; withdrawing money from a bank's ATM and so on. Automated systems are designed to assist in and improve our day to day lives. The impacts of failures of these systems in our personal, social, economic, national, and global lives are an extensively studied topic. However, there exists another side of the story about the interactions among automated systems and their human users. Although computers are used to simplify human lives, the humans themselves can complicate and even jeopardize the operation of a computerized system by using the system in a manner that is unexpected by the designers of the system. While the behaviors of these computer systems are predictable and repeatable, each human operator's behavior can be unique in their capability of decision making, autonomous judgment and improvising actions. Practical formal modeling techniques and tools can help validate and possibly aid in improving the tolerance provided by the operator dependant computer systems against operator errors. Interest in this type of contribution has extended beyond specialized areas like pilots of jets and operators of nuclear plants to diverse roles like: customers in e-commerce transactions or automated retail checkouts, air-traffic controllers, managers of smart warehouses, manufacturing systems and smart office buildings. In most of the scenarios the human operators are given specific recommended task descriptions to follow. Of course the recommended task specifications themselves need to be correct and the bulk of research activity focuses on proving the safety and effectiveness of recommended task specifications. However, deviating from safe and effective task specifications can lead not only to severe losses but to great financial and, in some safety-critical systems like in health care sector, even fatal consequences. Hence specifying and analyzing the combination of computerized systems and their human operators where the operators can deviate from their suggested task specifications is an area of intense interest.

Essentially, all computer systems have some sort of built-in safeguard mechanism to protect against loss. In systems which depend heavily on human operators, providing tolerance of human errors becomes more critical. Fault tolerance analysis [8] considers tolerance against both hardware and software failure and malfunctions. We intend to complement that research by focusing on an area that is often either ignored or oversimplified. That area is the uncertainty contributed by deviations of operators from procedures recommended to them. Human operators have the capability to subjectively introduce errors in the system. They may unpredictably vary their behavior based on frame of mind, time, presence and influence of other operators etc. Computerized systems must be designed to be not only user-friendly but also safe from users. Current formal frameworks sometimes omit any assumptions about the human operators, sometimes they are considered as all powerful antagonists who can do absolutely anything, and at some other times they are assumed to behave perfectly and always operate within the guidelines provided to them. We suggest providing a middle ground, where the human operators are neither neglected, nor are they allowed to be unrealistically powerful antagonists. Instead the human operators can deviate within "tolerable" bounds from their recommended task specifications. The focus of this work is analyzing the level of tolerance: are the common realistic human operator behavior variations included in the "tolerable" bounds? Every human interaction-intensive system will benefit from being modeled and analyzed against variations of human

task execution manners. We need to mention that we do not consider the cognitive reasoning behind atypical human behaviors. We focus on the effects of their actions not on why the actions were chosen to be performed in the first place.

A. Protection Envelope

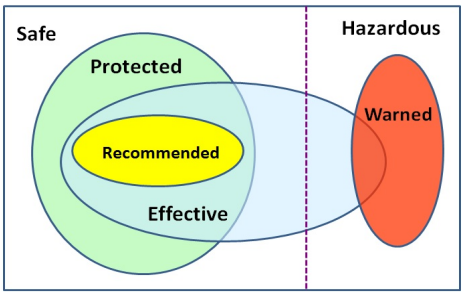

Fig. 1. Protection Envelope

We now present a categorization of human behaviors. We categorize the set of all possible human behaviors among subsets shown in the Venn diagram in Figure 1. This decomposition is defined with respect to the behaviors of a complimentary set of players, in this case the computer system, the operating platform etc. For any such analysis, we need a domain-dependent concept for progress and loss. Then, the *Safe* behaviors are those in which the actions of the operator never lead to any *loss*. Behaviors that are not safe are *hazardous*. *Effective* behaviors are ones in which some progress is made. Among the effective behaviors are some desired behaviors which we refer to as *recommended* behaviors in which the operator exactly follows the steps in his task description. There may be ways to make progress that are not recommended, perhaps because the recommended procedures are just meant to describe one of many ways to get the job done or because other ways of doing the job may be hazardous. The *warned* behaviors, often specified in the warnings section of a user manual, are the recognized set of hazardous behaviors. The *protected* behaviors, the focus of this work, are ones in which the operator may vary from recommended or effective behaviors without causing any hazardous consequence. The "protection envelope" is provided by an engineered set of properties of the system that form a specified subset of safe behaviors of the human operators. The "protection envelope" enforces less stringent guidelines for the human operators at the cost of a reduction in its characteristics. While the recommended behaviors are both effective and safe, the protected behaviors are guaranteed to be safe only. In any scenario, the designers of a system would aim at increasing the robustness of a system against atypical human behaviors by enlarging the protection envelope. An idealized situation would be where the protection envelope represents all possible safe behaviors.

A protection envelope is a measure of the robustness of the automated system against human action variations. The system has safeguard mechanisms to *protect* itself against the deviant behaviors in the protection envelope. Ideally each and every aberrant human action variation should be a protected one. It is usually infeasible to identify and protect against all possible atypical human behaviors. The protection envelope provides a trade-off: a collection of erroneous behaviors that are safe for the system. If the humans guarantee to behave in a protected manner the system can guarantee safety. Once a protection envelope has been provided by the system designers, there arises the question of whether the protection envelope is a sufficient one. The aim of this work is to provide a framework for assessing the *quality* of protection envelopes. As a heuristic measure of the quality of the protection envelope, we take help of existing characterizations of human action variations. Norman categorized human errors as slips (unintentional) and mistakes (intentional) in [20]. Swain *et al* proposes four human error categories: (i) errors of omission, (ii) errors of commission, (iii) sequence error (out of order execution), (iv) time error: performing actions too fast, too slow etc. in [24]. Hollnagel's phenotypes of erroneous actions [10] provide us with a thorough listing of usual patterns of atypical task executions by humans. We propose studying variations of the recommended behavior according to the error patterns provided by Hollnagel to assess the robustness of the protection envelope. If a protection envelope contains all possible human action variations suggested by him then we state that the protection envelope does a *quality* job of tolerating all standard human action variations. The possible erroneous behaviors suggested by Hollnagel are presented below.

Hollnagel's Characterization of Erroneous Human Actions Hollnagel's characterization of the erroneous human actions manifested while executing the subtasks or actions belonging to a task is comprised of two broad categories: *phenotypes* and *genotypes*. The genotypes are based on the reasons leading to erroneous actions and is not considered by us. The phenotypes of erroneous actions is based upon the way erroneous actions gets manifested. Then based on the level of detection he provides three classes of erroneous actions: (a) Zero-order: The erroneous actions that can be detected by comparing with an expected action or a prescribed action sequence, (b) First-order: detecting first-order erroneous actions involves combinations of several zero-order erroneous actions and may need to consider history of zero-order erroneous actions that have been performed, (c) Second-order: detection of second-order erroneous actions involves detection of nested or limited number of first order detections. Zero-order erroneous action detections can be categorized based on:

- timing involved in their execution:
  - Premature/Delayed start/finish of an action: A subtask is started/finished much earlier/later than the expected finishing time.
  - Omission of an action: A subtask is never started.
- action sequence manifested during execution:
  - Jump forward/backward: An action much later/earlier in a task sequence is performed earlier/later.
  - Omission: This is actually a repeat of the time-based omission phenotype presented earlier. Here an action from exactly the next step than the currently expected

step is executed.
- Repetition: A subtask is executed a second time.
- Intrusion: An action from another task sequence gets executed.

First-order detection phenotypes are: (i) Spurious intrusion, (ii) Jump, skip: several steps are omitted or performed ahead or performed later, (iii) Place loosing: A more involved permutation of the actions in a task sequence almost giving the impression of random execution of task, (iv) Side-tracking: An expected part of the task gets replaced by another part of the task. A special case is *restart*, (v) Recovery: A forgotten action or a group of actions may get performed, (vi) Capture: Here a part of another task sequence is inserted in place of a part of the desired task sequence. A special case of capture is *branching*, (vii) Reversal: It is the permutation of two adjacent actions: constituted of a jump forward and jump backward of actions, (viii) Time compression: A sub-sequence of actions get performed in a time much less than their combined expected execution time. Every human-intensive system should be capable of remaining safe against these standard behavior variations.

The first issue that we want to solve is: Is the protection envelope capable of protecting the system from typical human operator action variations? More specifically, given a protection envelope and a recommended task description, are the standard atypical executions of the recommended task contained within the protected behaviors? If not, what are the erroneous behaviors that are not covered by the protection envelope? The system designers should at least be aware of the specific atypical behaviors that can compromise safe operation of the system.

Even in the absence of a protection envelope, we should be able to assess which variations of the recommended behavior are safe. This is the second issue that we intend to resolve. There are situations where system designers develop a system without striving to make it robust against the environment. There can be cases where determining the expectations of the system about human action variations is rendered unimportant by the extreme difficulty of developing a simple non-robust system. There are situations where precisely formulating the protection envelope is not an easy undertaking. In these types of situations the developers are tempted to design a system without putting any thorough emphasis on the environmental effects. In all these scenarios although having a protection envelope would still be desirable, it will not be available for analysis. We still need to assess the robustness of the system against all standard human action variations in such situations. This can be achieved by falling back to the safety properties. We can assess whether standard varied action sequences conform to the safety property instead of the protection envelope. The advantage of our study in this case will be that the analysis results will help identify the safe aberrant human behaviors. We will provide a collection of human behaviors deviating from the recommended manner that can be demonstrated to be safe. This collection of deviant but safe human behaviors is in fact *a* protection envelope.

A positive impact that we intend for our work to have is that in case where a possible human error can be identified as a source of threat to a system, the system developers will get interested enough that they will attempt to prevent such an aberrant behavior from occurring in the first place. They will investigate the possible reasons for aberrant human behavior. If there is a lack of proper guidelines or recommended behavior they will try to improve upon them; if there is an ambiguous human-computer interface they may either improve the interface or provide rigorous training to eradicate any confusion to disambiguate it. Consider the Three Mile Island Accident [14]. There the expectation was that in the case of a valve problem, the operators will take immediate steps to seal the reactor so that coolants do not leak. This should have been part of the recommended behavior for the operators in case of an emergency. However, in this case ambiguous alarm system design caused the operator to omit the action of containing the coolants. The consequences of such a system is of such enormous proportions that the system designers could have been encouraged by a formal analysis delineating the dangers of the human operators not behaving in the expected manner. They could have incorporated measures to ensure safety like: (i) detecting that operators have failed to perform leak containment actions, (ii) some unambiguous secondary alert is set off which is extremely hard to ignore or (iii) some drastic secondary safety measure is built into the system to contain coolants.

A system may not be safe against all possible human action variations. Our analysis will provide the designers with a comprehensive list of all possible human action variations that can be harmful to the system. The designers can now make informed decisions about which action variations can be or should be reconsidered and possibly be made safe against. Before presenting our framework for achieving this goal we first present other works related to ours and an example scenario that we will use to illustrate our work.

II. Related works

Human workflow specification and verification works are very related to ours. Typical workflow verification entails determining whether a recommended workflow is deadlock free, live, livelock free and conforms to the desired goals. Formalisms used for modeling workflow include CSP [27], [17], Petri nets [25] etc. We used CSP to model and analyze human operator behaviors [9]. In [23] Shin *et al* use graphs and deterministic finite state automata to model and analyze human operator behavior. Clarke *et al* show in [5] how a collaborative procedure for humans can be specified using a process specification language Little-JIL and analyzed for satisfaction of desired properties. Their process descriptions can include exceptional situation handling task descriptions and hence is very close to our work. Tasks have been modeled using trees [16], real time logic [12] and predicate logic [13]. Operator Function Model (OFM) [19] provides a finite state machine based analytical tool to give a task analytic structure of operator behavior. All these works are mostly about verifi-

cation of only the recommended behavior of human operators. In our case we study robustness against various mutations of the recommended behavior.

Bolton *et al* extend OFM to EOFM [2] with task sequencing and conditional constraints to better model human task behavior. In one of their work with EOFM [3], they use the phenotypes of erroneous actions of Hollnagel to predict possible source of errors for a system. Given a recommended task EOFM for a human operator their approach is to (i) create another EOFM model which has all possible error phenotypes embedded in it, (ii) model check the new EOFM against system requirements to find a counter example. These counter example will demonstrate how an erroneous action might be problematic for a system. Our approach is different in that at one time we only introduce a single phenotype. Then we present which phenotypes the system is robust against and which phenotypes the system is *not* robust against. The system designers will now have a thorough analysis of all possible erroneous actions and their possible effects on the system. Our technique of progressively introducing errors in a task description through the different orders of erroneous action phenotypes is similar to *fuzz* testing of software reliability [18]. Fuzz testing is performed by providing unexpected, erroneous, random inputs to a computer program.

A sequence of works on human cognitive process modeling by Curzon *et al* [6], [22], [7] is also related. They focus on all possible correct and incorrect cognitive interpretations of user interfaces. We focus on how a human executes a task like operating a nuclear power plant using a computer system. They model the outcomes of human cognitive process as nondeterministic choice between all possible decisions. Although we do not focus on the human decision process but on the actual decision, our nondeterministic choice model for decision among possible actions is the same as their approach.

III. Example Scenario

Correct identification and correlation of data and patients is very important in health care. Everyone has heard horror stories about patients who got an operation intended for another patient [4], died because they got the wrong medication [15]. As a result, health care facilities such as hospitals, labs, and clinics have rigorous task specifications for clinical personnel that involve careful identification. There are efforts to improve efficiency in hospitals by using *A*utomated *I*dentification and *D*ata *C*apture (AIDC). Traditionally this task is comparatively manual. For instance, a nurse collects the weight of a patient on a mechanical scale and writes it on into a paper file. With the emergence of Electronic Health Record (EHR) databases, the nurse then enters this information into an EHR at a later stage. But if wireless medical devices were used, the task could have been streamlined. In the case of a prototype system built at our university, a pulse oximeter connects by Blue-tooth to a PDA (which we will subsequently refer to as a *medical mediator*), which uses an RFID tag scanner to identify the patient and the clinician before entering the results into an EHR via a WiFi

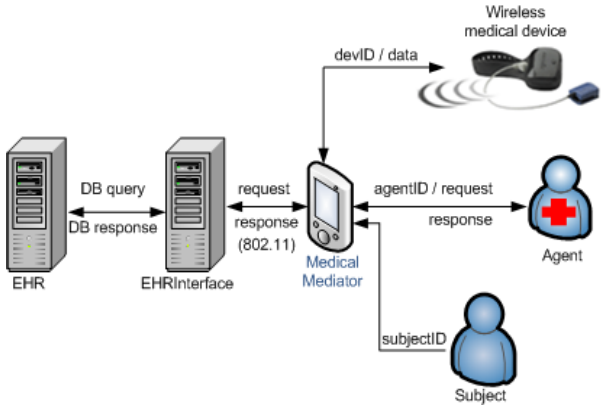

Fig. 2. AIDC system design

link. We add the additional step of identifying the device taking the reading to obtain the following nurse task description:
(i) Identify the patient by scanning the patient's identification tag. (ii) Identify the device to be used to take a reading by scanning its identification tag. (iii) View the information to verify whether the correct entities were identified. (iv) Take a reading and (v) Check the reading to see if it is acceptable for entry into the EHR and (vi) Approve the reading if it is.

This task description can be augmented by some simple recovery instructions like using a reset function if the entities are not correctly identified or the reading is not satisfactory. The goal of the overall system is to ensure that there is correct association of patient and data in the EHR. If the computer system operates safely then the onus lies on the nurse who collects the data from the patient. Throughout a data collection sequence, it must be the case that the nurse always correctly asserts that the patient from whom a data was collected is *the* patient whom she had identified. Intuitively: *for all patients* p_1 *and* p_2. *took reading from* p_2 *using a device implies did not identify* p_1 *in the previous step*. However, one can notice that the computer system cannot provide a guarantee like this. This is a constraint that has to be imposed on the nurse because she is the one physically facing the patient and knows exactly which patient got a device attached to. No matter how many checks are included at every step, if a nurse negligently asserts that an incorrect combination of data is correct, the computer system has to accept that.

IV. Framework for analyzing operator action variations

The framework we propose is comprised of four steps: (i) Construct a model for the interactions among the automated system components and the human operators, (ii) Obtain the recommended task specification for the human operators and safety and protection envelope definitions, (iii) Introduce well established variations into the recommended task specification, (iv) Perform analysis of the various mutant task specifications and provide results to the system designers about safety conformance or containment in protection envelope for each of the mutated task specifications.

We now need to resolve the following issues: How to encode the interactions between the human operator and the computer system? How to model the recommended task specification? How to introduce variations into the recommended task specification? How to analyze the different transformed tasks?

A. Human Computer Interaction Model

We need a model that can identify different entities, the actions performed by different entities in different situations and is able to capture the evolution of the system through the combined actions of the entities. The model offering all these characteristics is the *C*oncurrent *G*ame *S*tructures(CGSs) [1].

1) Concurrent Game Structures: A CGS is an 8-tuple, $\langle P, Q, q_0, \Sigma, \Pi, \pi, e, \delta \rangle$ where the components are as follows: P is a finite set of players. Q is a finite set of states and $q_0 \in Q$ is the initial state. Σ is a finite nonempty set of abstract actions for the players where the silent action $\tau \in \Sigma$. Π is a finite set of propositional atoms that help identify the states. $\pi : Q \rightarrow 2^{\Pi}$ is a function that gives the set of propositional atoms that hold of a given state. The function $e : P \times Q \rightarrow 2^{\Sigma}$ gives which actions are enabled for each player in each state. We denote the action choices available to a player p in a state q by $e_p(q)$. We enforce that the silent action τ is enabled for each player in each state: $\forall q \in Q. \forall p \in P. \tau \in e_p(q)$. The function $\delta : Q \times \ e_{p_1}(q) \times e_{p_2}(q) \times \ldots e_{p_{|P|}}(q) \rightarrow Q$ defines state transitions. Given a state q, and a vector of actions $\bar{v} \in \Sigma^P$ where $\bar{v}|_p \in e_p(q)$ for each p, the value $\delta(q, \bar{v})$ is the next state q'. We require that $\forall q \in Q. \delta(q, \tau \times \ldots \times \tau) = q$. The reason behind enabling the τ action for each player in each state and $\bar{\tau}$ self-loops in each state is based on the perception that human operators cannot be assumed to always be synchronized with the computer systems. They may idle away for some time before performing an action. Similarly a system may become unresponsive at times and fail to respond within an expected time frame.

2) Human Computer Interaction Model using CGS: The CGSs provide a modular, uniform formal technique for modeling both the systems and their human operators. In any scenario under consideration, the human operators and the computer system can constitute the set of players. The vocabulary of these players can be captured by Σ. The enabled action function e can indicate which actions are physically possible for the human operators to perform. The combined actions from the operators and the computer system can help the system transition from one state to another. Thus the transition function δ will capture the interactions among the system components and the human operators. In the AIDC scenario, the players can be: the nurse, the RFID tags, the patients, the devices, the medical mediator and the EHR. The nurse and the medical mediator can have vocabularies like: {`PressStart`, `ScanPatID(i)`, `VerifyIDs`, ...} and {`AllowPressStart`, `AllowDataCapture`,...} respectively. In the initial state, the nurse is allowed to press a start button to initiate the AIDC process. Then in the initial state, the enabled action function e for the nurse and the medical mediator will contain the actions `PressStart` and `AllowPressStart` respectively. Let there be a state *AIDCstarted* where a proposition $PressedStart$ is true indicating that the start button has been pressed. From the initial state, if the nurse and the medical mediator performs the actions `PressStart` and `AllowPressStart` then the system can go to the *AIDCstarted* state.

3) Recommended Task Specification: Recommended procedures will be a sequence of human operator actions where the actions will be in the vocabulary Σ of a CGS corresponding to the procedure. For example a recommended task description for the nurse for taking reading from a patient can be: {`PressStart`, `ScanPatID(i)`, `ScanDevID(i)`, `VerifyIDs`, `TakeReadingWithDevice(i)`, ...}.

4) Protection Envelope Model: The protection envelopes can be most efficiently and succinctly specified by using logical properties. Each human operator behavior satisfying the property will be a protected behavior. Since the protection envelopes will involve guarantees like "Step B is *not* taken *until* step A has been finished", the properties will need to be temporal in nature too. We will use Linear Temporal Logic [21] formulae for this purpose. LTL formulae are usually checked against finite state automata. The transition system of CGS is inherently like that of a finite state automaton. Specifying the protection envelope using LTL will allow us to verify inclusion of a behavior in the protection envelope by simply checking whether the CGS corresponding to that behavior satisfies the protection envelope property. Similar reasoning works for specifying safety properties using LTL. Also the existence of efficient LTL model checkers is an added incentive for this approach. In the AIDC scenario, the nurse is responsible for every pairing of patient ID, device ID and data that is stored in the EHR. The nurse validates the IDs scanned from the patients and the devices. A part of the protection envelope for the nurse can be: $\Box(\texttt{patIDOked(i)} \wedge \texttt{devIDOked(j)} \rightarrow (\diamond^{-1} \texttt{patIDScanned(i)} \ \wedge \diamond^{-1} \texttt{devIDScanned(j)}))$. This property states that if the nurse validates the identities of the ith patient and j-th device then exactly that i-th patient and j-th device were scanned in the previous steps. This past time LTL property can be converted to an LTL property. Here `patIDScanned(i)`, `devIDScanned(j)`, `patIDOked(i)` and `devIDOked(j)` are CGS state propositions that are set to true when the nurse has scanned or approved the identity of the i-th patient and j-th device. This property encompasses those nurse behaviors where there exists correct correspondence among the scanned and validated patient and the device identities. A nurse while undertaking training for using the medical mediator might ask whether she can scan the RFID tag of one patient multiple times. What if she scans the RFID tag of the intended device before scanning the RFID tag of the intended patient? Will she have to restart the whole process if she makes this type of mistake? It is not readily comprehensible from the provided protection envelope whether it contains such variations. This illustrates yet another utility of our framework: a situation where the protection envelope is too convoluted to explicitly express closure under different

human action variations. Our framework will be capable of providing the answers to such queries.

B. Obtaining Deviations from Recommended Procedure

We consider a subset of the phenotypes of erroneous actions suggested by Hollnagel. Other phenotypes can be handled similarly. We first show how we can attain new task specifications from a given recommended task specification. Then each new task specification can be analyzed to determine whether this is physically possible and safe or protected.

- k times repetition. Let a task T be linearly composed of n subtasks T_i, $T = T_0, T_1, \ldots, T_{n-1}$. Then each subtask is repeated k times to create new tasks. For example: $T' = T_0, T_1, \ldots, T_i, T_i, \ldots, T_i, \ldots, T_{n-1}$.
- Omission: Let a task T be linearly composed of n subtasks T_i, $T = T_0, T_1, \ldots, T_{n-1}$. Then for each subtask, we can create a new task specification by omitting it. For example: $T' = T_0, T_2, \ldots$.
- Reversal: Reversal involves permutations of two subtasks. Let a task T be linearly composed of n subtasks T_i, $T = T_0, T_1, \ldots, T_{n-1}$. Then for each subtask T_i we permute it with all other subtasks and create new task specifications. For example: for subtask T_1, we consider $T' = T_1, T_0, \ldots, T_{n-1}$, $T'' = T_0, T_2, T_1, \ldots, T_{n-1}$, $\ldots$, $T''' = T_0, T_2, T_1, \ldots, T_{n-1}, T_1$ and so on.
- Delayed and premature execution: We do not have a timed model and hence are incapable of directly analyzing such modifications. However we will show later how delayed task execution can be approximated due to allowing each entity to stall temporarily using the silent τ action in each state in our CGS models.
- Inclusion: Let a task T be linearly composed of n subtasks T_i, $T = T_0, T_1, \ldots, T_{n-1}$. Then for each subtask T_i we insert it in front of all other subtasks to create new task specifications and analyze them. For example: for subtask T_1, we consider $T' = T_1, T_0, T_1, \ldots, T_{n-1}$, $T' = T_0, T_1, T_1, T_2, \ldots, T_{n-1}$, $\ldots$, $T' = T_0, T_1, T_2, T_1, \ldots, T_{n-1}, T_1$.
- Intrusion: Given two recommended tasks $T = T_0, T_1, \ldots, T_n$ and $T' = T'_0, T'_1, \ldots, T'_{n'}$ we obtain new tasks T'' by inserting each subtask $T'_i \in T'$ after each subtask T_j in T. For example: $T'' = T_0, T'_0, T_1, T_2, T_3, \ldots, T_n$.

C. Analyzing Mutant Task Specifications

We introduce a broad class of variations into the recommended task description to constitute new task descriptions. We first need to build CGS models of human computer interactions for each new task description to be able to analyze it. Each new task corresponds to a human behavior. The behavior is exhibited by executing steps of the task while interacting with the computer system in the specific order suggested by the task specification. Using this view of human tasks we describe how we can create CGSs corresponding to the varied task descriptions.

1) Model for Recommended Procedure and its Variations: In order to facilitate the process of building a new CGS for each atypical behavior created from the recommended behavior, we first construct a CGS we will refer to as the C_{all}. In all states in the C_{all}, each and every action that is physically possible for the operators to execute along with the response of the computer system (and hence all corresponding transitions) will be enabled in the e function. Referring to Figure 1, C_{all} contains all possible safe and hazardous human behaviors. Given a C_{all} and a task specification, we will create the CGS corresponding to the task specification by constricting the C_{all} by restricting the actions available to the human operator at each state. The first subtask of the task description will be searched for in the initial state. All other human operator actions (except for τ) will get disabled in that state. From then on, the task description will help restrict the e function for the operators. Starting from the initial state, the transition induced by the combination of human operator system action will be followed to ascertain the next state for inspection. At every state reached by following the provided task description, the e function and hence the transition function δ will be restricted. If at a state in C_{all}, the operator action suggested by the task description cannot be performed then we know that the task specification is not a physically possible one.

2) Analyzing Aberrant Behavior CGSs: Provided with a C_{all}, a mutant task specification, and a protection envelope or safety property, we can construct the CGS corresponding to that task specification using the procedure as described above. We then need to analyze the CGS with a suitable model checker. We have developed an extension to a tool Tutela (under construction) (`https://netfiles.uiuc.edu/yasmeen/www/tutela.html`) that assists in analyzing recommended task specification variations. We chose the model checker SPIN [11] for verifying LTL property satisfaction in the back-end of Tutela. The tool works as follows: it (i) takes

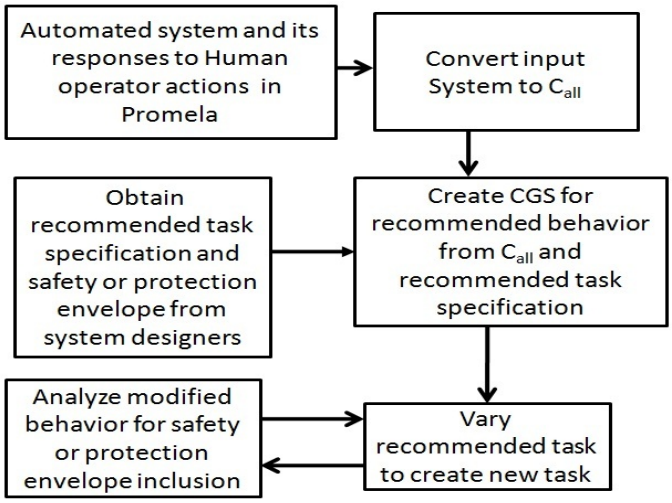

Fig. 3. Framework implementation in Tutela

a C_{all} and a recommended task specification as input, (ii)

automatically (a) derives new task specifications from the recommended task, (b) creates CGS corresponding to each new task, and (c) analyzes the CGS with a model checker for property conformance. A flow diagram for the relevant part of the software is presented in Figure 3. The C_{all} is provided in the Promela language which is the input language for SPIN. The recommended task is given as a sequence of human operator actions using C_{all} vocabulary. The protection envelope or safety property is specified using LTL. We describe the interesting mechanisms of the tool like C_{all} creation from Promela input, CGS creation for a task description, CGS encoding into Promela for analysis by SPIN in another work [26]. After a task specification model has been created, we first determine whether it is physically possible to be executed. Physical impossibility of execution is indicated by the task specification requiring an action to be performed by the operator at a state where it is not contained in the e function of that operator in that state in the CGS. A task variation that is not physically possible is inherently safe as the system is capable of barring the human operator from executing the task. Only feasible task specifications are analyzed for being protected or safe.

D. Example Scenario Revisited

Let us consider a simple AIDC scenario with two patients `pat1` and `pat2` and two medical devices `dev1` and `dev2`. There is only one nurse and one medical mediator. The nurse is supposed to take a reading from `pat1` using `dev1`. Later on she is supposed to take a reading from `pat2` using `dev2`. Simple recommended task descriptions for the nurse for these two procedures are: (1) press start, scan `pat1` ID, scan `dev1` ID, verify IDs displayed, take reading, verify captured data for storage in the back end database; and (2) press start, scan `pat2` ID, scan `dev2` ID, verify IDs displayed, take reading, verify captured data for storage in the back end EHR database. The protection envelope for the first recommended task is: *if the nurse verifies `pat1ID` and `dev1ID` then previously she had scanned only `pat1ID` and `dev1ID`* and *if the nurse verifies `pat1ID`, `dev1ID` and data `d` then the data `d` WAS captured from `pat1` using `dev1`*. Now let us analyze the first recommended task description for the nurse.

- Repetition: Repeated scanning the ID tag of the patient and the device are found to be protected. This is a very important result, because scanning a tag multiple times can be a very common phenomenon while using the AIDC system. This assures that the nurses will not need to restart the procedure if they have inadvertently scanned the same ID multiple times.
- Reversal: We performed permutation of each pair of actions in the recommended task specification. Quite unsurprisingly, we find that the Press Start button action cannot be permuted with any other action. If a nurse wants to perform AIDC using the medical mediator she needs to press the start button as the first step of the entire procedure. The most significant finding for reversal is that scanning of the ID for the patient and the device can be swapped. Hence nurses will not need to restart the whole procedure if she has scanned the RFID tag of the device before scanning the RFID tag of the patient.
- Omission: Similar to the findings discussed above omission of the Press start button action is infeasible. Also the nurse needs to scan the IDs of the patient and device in some order for the process to progress. Hence omitting these actions will result in the entire process getting stalled.
- Delay: Since we do not perform timed analysis we are not capable of analyzing the effect of delaying on part of the operators. However, since we insist that the silent action τ be an enabled action for each player in each state in the CGSs it provides a mechanism for modeling delayed action. This is evident from the analysis results from the repetition of actions step. Although automated systems are supposed to be capable of almost immediate response, in the case that the system *is* late in responding, the operator might have gotten impatient and pressed a button twice. In future we will undertake inserting the silent τ action an arbitrary number of times into the recommended task specification to incorporate delayed human action analysis in our framework.
- Intrusion: For this analysis, we inserted subtasks from the recommended task specification for AIDC from the second patient into the recommended task specification for the first patient. We did this for two different implementations of the AIDC scenario. During the initial stage of the AIDC project development, the system under development was somewhat rigid in the sense that once the ID tag of a patient was scanned only a device ID tag could be scanned in the next step. Then the design was made more flexible by allowing the same ID tag of a patient or a device to be scanned multiple times. Ideally, an RFID tag scanner cannot distinguish among the tags belonging to different entities. The mobile mediator can determine whether the tag is a patient tag or a device tag after it has consulted with the back-end database. Hence the most flexible form of the system should allow all RFID tags to be scanned in the ID scanning steps. Then later on, when the nurse is asked to verify that the scanned IDs are acceptable, the nurse gets a chance to correct any errors. We analyzed the last two models. At this point we intend to focus on an unusual finding on our part. The finding occurred in the second version of the AIDC model (first version analyzed by us). In this model once the tag of a patient has been scanned, only that tag can be scanned multiple other times. Then the ID tag of a device needs to be scanned. Now, our findings surprisingly contained the result: `"insertion of scanPat2ID after scanPat1ID is physically possible"`(may not be safe). Although this may seem to be an erroneous judgment, the reasoning that leads to this is as follows:
  - Nurse presses the start button.
  - Nurse scans the ID tag of the first patient. The

combination of the RFID tag scanner and the mobile mediator is slow enough (modeled using the silent τ action) that the nurse decides that she was not able to scan the ID tag of the patient successfully.

– Nurse now scans the ID tag of another patient because she wants to proceed with the AIDC procedure of the second patient.

We had decided that the τ action be available to each entity at each step to enable modeling delayed response from them. This result alerts us that if it is possible for the automated systems to not respond to the operator actions in an immediate manner, the operators may get confused and decide to wait, re-perform the step or even worse insert a step from a different task specification. A modeling of the scenario where computer system components are not allowed to stay idle through the τ actions would not have uncovered this issue. The analysis results are available on the Tutela web site.

E. Protection Envelope Suggestion

If we perform the analysis with a safety property instead of a protection envelope property, then we will obtain a classification of the erroneous human behaviors as to whether they are safe or unsafe for the system. Then the collection of safe erroneous behaviors can be examined by the system designers to form a protection envelope. For example, consider the safety property for the AIDC system: *if a record of a data r said to be collected from a patient p with a device d is stored in the EHR, then only that patient and that device had been identified and data r had been collected from exactly that patient p using that device d*. Using this property we obtained that multiple scanning of the IDs of the patient and the device is safe, scanning the ID of the device before the patient is safe but the identification validation can not be performed before the identities have been scanned. The system designers can combine these information to form a nurse protection envelope (partial) like this: $\Box(\neg\texttt{patdevIDOked}(\texttt{i},\texttt{j})\ \mathcal{U}\ (\ \texttt{patIDScanned}(\texttt{i}) \wedge \texttt{devIDScanned}(\texttt{j})))$.

V. Conclusion

The manner in which a human operator uses an automated system can compromise safety of the automated system. We have presented a formal framework and a tool for modeling and analyzing an automated system against typical deviant human behaviors. The framework will assist system designers to understand the strengths and weaknesses of the assumptions they have made about human behaviors. In case such an explicit protection envelope around human behavior is absent, our study will help determine the protection envelope itself. This study of robustness against typical deviant human operator actions will hopefully aid the system designers in taking informed decisions about altering their dependance on "reasonable" human operator behavior.

Acknowledgment

This work was funded in part by NASA Contract number NNA10DE79C and NSF Award number 0917218. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NASA and NSF.

References

[1] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[2] Matthew L. Bolton and Ellen J. Bass. Enhanced operator function model: A generic human task behavior modeling language. In *SMC*, pages 2904–2911. IEEE, 2009.

[3] Matthew L. Bolton and Ellen J. Bass. Using task analytic models and phenotypes of erroneous human behavior to discover system failures using model checking. In *54th Annual Meeting of the Human Factors and Ergonomics Society*, page 992996, 2010.

[4] Mark R. Chassin and Elise C. Becher. The Wrong Patient. *Ann Intern Med*, 136(11):826–833, 2002.

[5] Lori A. Clarke, George S. Avrunin, and Leon J. Osterweil. Using software engineering technology to improve the quality of medical processes. In *ICSE Companion*, pages 889–898, 2008.

[6] Paul Curzon and Ann Blandford. From a formal user model to design rules. In *DSV-IS*, pages 1–15, 2002.

[7] Paul Curzon and Ann Blandford. Formally justifying user-centred design rules: A case study on post-completion errors. In *IFM*, pages 461–480, 2004.

[8] Catherine Dufourd, Alain Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *ICALP*, pages 103–115, 1998.

[9] Elsa L. Gunter, Ayesha Yasmeen, Carl A. Gunter, and Anh Nguyen. Specifying and analyzing workflows for automated identification and data capture. In *HICSS*, 2009.

[10] Erik Hollnagel. The phenotype of erroneous actions. *International Journal of Man-Machine Studies*, 39(1):1–32, 1993.

[11] G. J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, 2004.

[12] Farnam Jahanian and Aloysius K. Mok. Safety analysis of timing properties in real-time systems. *IEEE Trans. Software Eng.*, 12(9):890–904, 1986.

[13] C.W. Johnson and A.J. Telford. Extending the application of formal methods to analyse human error and system failure during accident investigations. *Software Engineering Journal*, 11(6):335–365, 1996.

[14] John G. Kemeny. *Report of The President's Commission on the Accident at Three Mile Island: The Need for Change: The Legacy of TMI*. Washington, D.C.: The Commission, 1979.

[15] J. M. Corrigan Kohn, L. T. and M. S. Donaldson. To err is human: Building a safer health system. http://www.nap.edu/catalog/9728.html.

[16] Nancy G. Leveson. Software safety: Why, what, and how. *ACM Comput. Surv.*, 18(2):125–163, 1986.

[17] Peter A. Lindsay and Simon Connelly. Modelling erroneous operator behaviours for an air-traffic control task. In *AUIC*, pages 43–54, 2002.

[18] Barton P. Miller, Lars Fredriksen, and Bryan So. An empirical study of the reliability of unix utilities. *Commun. ACM*, 33(12):32–44, 1990.

[19] C.M. Mitchell. Gt-msocc: A domain for research on human computer interaction and decision aiding in supervisory control systems. *IEEE Trans. on SMC*, 17(4):553–572, July 1987.

[20] D. A. Norman. Errors in human performance. Technical Report 8004, University of California, San Diego, Center for Human Infromation Processing Report, 1980.

[21] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[22] R. Ruksenas, Paul Curzon, Jonathan Back, and Ann Blandford. Formal modelling of cognitive interpretation. In *DSV-IS*, pages 123–136, 2006.

[23] Dongmin Shin, Richard A. Wysk, and Ling Rothrock. Formal model of human material-handling tasks for control of manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 36(4):685–696, 2006.

[24] A. D. Swain and H. E. Guttman. Handbook of human reliability analysis with emphasis on nuclear power plant applications. Technical report.

[25] W. M. P. van der Aalst. Verification of workflow nets. In *ICATPN*, volume 1248 of *LNCS*, pages 407–426. Springer, 1997.

[26] Ayesha Yasmeen and Elsa L. Gunter. Automated framwork for formal operator task analysis. In *ISSTA*, 2011(in press).

[27] Xiangpeng Zhao, Zongyan Qiu, Chao Cai, and Hongli Yang. A Formal Model for Human Workflow. In *International Conference on Web Services (ICWS)*, pages 195–202, 2008.