A Human Operator Model for Medical Device Interaction Using Behavior-Based Hybrid Automata

Gerrit Niezen and Parisa Eslambolchilar

***Abstract*—This paper describes the design and implementation of a control-theoretic model that can be used to model both the discrete and continuous behavior of a human operator. The human operator model can be used to compare different device user interfaces in terms of human performance. The implemented human operator model combines an ON–OFF control model and a behavior-based hybrid automaton with three controllers. The controllers, defined as continuous, discrete, and fine-tuning behavior, simulate the user's conceptual model of the user interface. The device model used is that of a commercial syringe pump with chevron keys, described as a formal specification. Results of the human operator model simulation were generated for 20 different numbers obtained from syringe pump log files. The simulation results were compared over 33 trials to a lab study employing a device based on the formal specification. The result of the simulation shows a significant similarity to the result of the lab study for all the numbers used.**

***Index Terms*—Automata, control theory, drug delivery, formal verification, human computer interaction, safety, simulation.**

I. Introduction

UP–DOWN buttons are a ubiquitous user interface input method, used in everything from microwave ovens to safety-critical medical devices. When used to enter numeric data, like times or rates, up–down buttons and their extended version, chevron keys, are defined as an incremental number entry method [1]. This differentiates them from serial digit entry methods like numeric keypads, where numbers are entered sequentially from left to right.

A chevron-key interface, as shown in Fig. 1, is commonly used for entering values on interactive medical devices and consists of four buttons. A "small up" and a "big up" key increases the displayed value, with the "big up" key causing a larger change than the "small up" key. The "small down" and "big down" keys both decrease the displayed value, with the "big down" key causing a larger change than the "small down" key. The interface allows for two modes of interaction: The user can press the button for a discrete change in displayed value, or the user can press and hold the button to change the displayed value at a rate dependent on the duration of hold [2].

Manuscript received November 28, 2014; revised March 4, 2015 and July 1, 2015; accepted September 10, 2015. Date of publication October 26, 2015; date of current version March 11, 2016. This work was supported by the UK Engineering and Physical Sciences Research Council [grant number EP/G059063]. This paper was recommended by Associate Editor E. Mercer.

The authors are with the Future Interaction Technologies Laboratory, Department of Computer Science, Swansea University, Wales SA2 8PP, U.K. (e-mail: me@gerritniezen.com; p.eslambolchilar@swansea.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2015.2487509

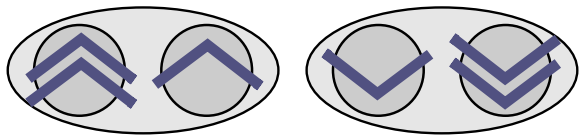

Fig. 1. Chevron-key interface, with the "big up" key on the far left, followed by the "small up" key on the middle-left and the "small down" key on the middle-right, with the "big down" key on the far right.

Number entry is a very common task in healthcare, and incorrect drug doses are a significant contributory factor to unnecessary fatalities [3]. Errors made using incremental number entry are less severe than numeric keypads [2]. That is, the difference between the intended number and transcribed number is lower, which makes them more suitable for safety-critical interfaces. To understand the impact of human erroneous behavior in interaction with number entry devices, we need to model human operator behavior. There are a number of well-explored approaches in human–computer interaction (HCI) to model human operator behavior based on, for example, device interfaces, cognitive models, and task-analytic models in interaction with numeric keypads. However, incremental number entry presents a unique problem: entering a number consists of both continuous and discrete interactions. Pressing the button once (or multiple times) for a short amount of time is a discrete interaction, while holding the button is a continuous interaction where the release of the button depends on the displayed value and the reaction time delay of the user. This continuous interaction can be modeled using a control-theoretic feedback loop.

Manual control theory offers a powerful and flexible approach for describing human behavior and analyzing human–machine systems [4]. Manual control theory is a discipline which cuts across traditional boundaries between scientific fields of study and uses the same language for systems of different hardware, whether physical or biological [5]. It has been applied to modeling human behavior and solving human factors problems for more than 60 years [4]; however, it has been largely overlooked outside the engineering arena, e.g., in HCI research. Manual control theory is a time-domain approach based around differential equations, which started emerging in the 1960s and has been widely used for modeling human pilots [6], [7].

Hybrid automata extends control theory to express both discrete and continuous aspects in the same formalism [8]. It has been used with success in areas like robotics [9] and aviation [10] to model both the continuous and discrete aspects of a system. In this paper, we will show how it can be used to model

both the continuous and discrete behavior of a human operator, using a chevron-key interface on a medical device.

The paper is organized as follows: First, we discuss the related work on user modeling, manual control theory, and hybrid automata. We then introduce our human operator model using behavior-based hybrid automata and manual control. A device model that interacts with the human operator model is then described. The results of a comparison study, where the human operator model is compared with a lab study, is shown to validate the model. This is followed by a discussion and conclusions.

II. Related Work

A. User Models in Human–Computer Interaction

In this section, we consider some of the approaches in HCI that attempt to model human behavior, i.e., the user. In these approaches, the human operator's behavior is determined by device interface models, cognitive models, or task-analytic models of human behavior.

1) Using Device Interfaces to Define User Models: With respect to formal modeling in HCI on device response to human behavior, the existing approaches use formal verification to determine four logic property categories for human–device interface models, identified by Campos and Harrison [11]: reachability [12], [13], visibility [13], [14], task related [12]–[14], and reliability [12], [14]. Campos' and Harrison's [11] work has been extended to heuristically assess usability using formal verification, or the output of formal verifications in HCI. For example, Hussey *et al.* [15] identified four usability properties: task efficiency, reuse, robustness, and flexibility. Kamel and Aït-Ameur [16] showed how four usability properties specific to multimodal human–device interfaces could be evaluated formally: complementarity, assignation, redundancy, and equivalence. Kamel *et al.* [17] also provided temporal logic patterns for verifying the "adaptability" of a multimodal interface: For a given initial state (which may encompass a condition where a particular modality is not available), the human operator will always be able to eventually find a way to reach a goal state.

Bolton *et al.* [18] provided a review of formal verification approaches to evaluate human–automation interaction. They identified two broad categories of formal verification approaches: those that focus on the analysis of the user interface of the system, and those that focus on the analysis of how the system is (supposed to be) used. Approaches in the first group typically use a model of the user interface under analysis, proving properties of the interface that are relevant to the operation of the system. Examples of properties include usability principles, mode confusion properties, and user-related safety requirements. To help focus on user relevant issues and behavior, the inclusion of mental models or knowledge models has been used to augment the analysis. Approaches in the second group work either with task models (see Section II-A3) of how the users are supposed to use the system or with cognitive models (see Section II-A2) of the mental process that drive that behavior.

Using a formal model of a device–user interface can inform whether certain situations and certain human behavior may contribute to a failure in interaction between a device and a user. Although the verification models are extremely powerful, they suffer from certain limitations [18]. One of the limitations is that traditional model checking is applied to systems that can be modeled with discrete variables. However, systems can have continuous quantities, and current techniques can handle systems models with no more than a half-dozen continuous variables.

Campos *et al.* [19] presented a systematic formal method for analyzing interactive systems that was based on resources rather than prescribed behavior. They argued that a *resource-based approach* can help to identify potential usability problems by exploring what should be available at the interface to support users. Using two commonly used infusion pumps as examples, they showed that this approach provides a means of comparing devices designed to support the same activities iteratively. Campos *et al.* showed that the presence of resources introduced a notion of plausibility and a more realistic conception of user behavior, which was based neither on rigid plan following nor random behavior.

Another approach is to encode assumptions about the user directly into the model, that is, joint models of user and device [20]. The approach could be characterized as embedding elements of a user model into the device model and thereby constraining the behaviors of the system being analyzed. In this case, the separation between device model and user assumptions is less clear potentially, and this can bias the user assumptions toward those that are needed to make the system work. By working with assumptions at a resource level [19], a clear separation is made between models and assumptions about users as expressed in terms of resources. These models are also relatively easy to build and the outputs are straightforward, providing a way for the HCI expert to contribute to a more rigorous analysis.

The approach we describe in Section II-C, similar to the resource-based approach, does not "prove" usability of the system, but rather identifies and investigates plausible and interesting behaviors to find and to fix usability problems and to investigate the effectiveness of different user strategies for achieving goals.

2) Using Cognitive Models to Define User Models: Cognitive models in HCI take into account the user's capabilities. Human cognition can be modeled as part of a formal system and then verified, i.e., what actions the user will use to interact with the system. These methods let the analyst formally verify that the system will always allow the operator to achieve their goals with a set of cognitive behaviors. These methods can also identify situations where the human operator fails to achieve his desired goals, or drives the system into dangerous operating conditions.

Programmable user models (PUMs) [21] are cognitive models that capture the knowledge and cognitively plausible behavior an operator is able to use when interacting with an interactive device, and implement them as part of a formal system model [22], [23]. PUMs take into account the goals of interaction with the device that the operator wishes to achieve, the operator's knowledge of the device, the information available to the operator through the interface (feedback), and a set of actions the operator can perform to interact with the device. The operator, at any moment of interaction with the device, uses knowledge

about the device and currently available feedback to decide on the next action(s) to achieve his goals. PUMs have been evaluated using formal verification with both theorem provers [24] and model checkers [25].

These models have at least one advantage over Goals, Operators, Methods, and Selection rules (GOMS), a description of how to calculate the time to accomplish tasks [26], [27], where it assumes the participants involved in interaction are well practiced and make no errors during task execution. With GOMS models, reliable estimated times must be available for all components in the task [28].

PUMs have been used to model different classes of human operator (expert versus novice) [29] in order to investigate when different types of operators may perform different errors when interacting with an automated teller machine (ATM). Keystroke-level timing analysis [26], [27] have been added into Curzon *et al.*'s [29] framework and used to evaluate timing performance of a human operator interacting with the ATM [30]. Similar to GOMS, KLM timing analyses estimate time with the provision that the sequence of actions required to perform a task is executed without error.

Cognitive models of users interacting with devices model human behavior explicitly and represent the cognitive basis for erroneous behavior, for example, they provide additional insights into safety, usability, skills, cognitive load, and salience. However, these analyses require each cognitive mechanism to be incorporated into the model. As such, they are likely to overlook certain behaviors in interaction.

3) Task Analytic Models of Normative Human Behavior: Task analytic models are commonly used to model human task behavior as sequences of activities to fulfill the goals of interaction with an interactive device [31]. Although these models are not concerned with modeling cognitive concepts such as attention and memory, they can model abstractions of these in order to model the user as a simple input–output system, where inputs can come from the user goals, the environment, or interfaces; and outputs are user actions [32].

These models have been used in the evaluation of human operator performance for a variety of purposes including usability evaluation [27], [33], timing analysis of human tasks [27], and alerting systems [34], [35]. Researchers have shown the usability of formal methods in verifying human operator behavior encompassed by task analytic models accomplishing their desired goals and/or avoiding dangerous system operating modes [35], [36]. Some models have been extended to incorporate erroneous human behavior into task models so that their impact can be evaluated as part of the formal verification [31], [36].

Task analytic models are computational structures and have been represented by some researchers as communicating state charts [37]. For example, Degani and Heymann [38] incorporated human task models into state chart models of a human–device interface and used them to explore human operator behavior during an irregular engine start on an aircraft. A Petri net-based formalism has been used in modeling human task behavior in a waste fuel delivery system to test the system's safety [39].

Bolton *et al.* [31] argue that task analytic models, such as operator function model [40] and enhanced operator function model [41], can be represented using discrete graph structures and that they can include human behavior in formal system models. A number of researchers have taken this approach forward [13], [39], [42], [43], and these works show that in the context of safety for example, the system will always operate safely if the users adhere to the modeled behavior. This highlights one of the limitations of these models: The analyses provide little or no insight into the impact of erroneous human behavior (unless manually incorporated into the task behavior model) [31]. Moreover, continuous interaction behavior and even the devices are not taken into account.

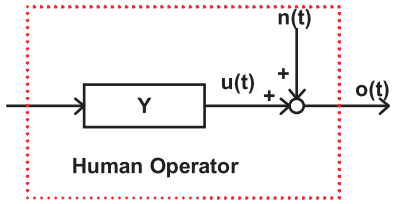

Fig. 2. Quasi-linear model of the human operator. Y is the linear transfer function; *u(t)* is the linear response; *n(t)* is internal noise (reflected noise in the perceptual and motor systems of the operator); and *o(t)* is the quasi-linear response. The noise is generally presumed to be uncorrelated with any input signal. Adapted from [45].

B. Manual Control Theory and Human Operator Models

Manual control is the study of humans as operators of dynamic systems. Early research focused on the human element in vehicular control [44]. In order to predict the stability of the full system, designers included mathematical descriptions of the human operators along with the descriptions of the vehicle dynamics. Applications of manual control theory have been modified and extended to applications in HCI. Humans are regularly asked to position the cursor on a menu, drag the scrollbar and other tracking and positioning tasks. Thus, the human operator can be modeled using the tools of manual control theory. It provides insights into the basic properties of human performance and facilitates the ability to predict the performance of human–machine systems [45].

Human behavior is nonlinear, but linear analysis still provides important insights into human performance and linear models may be able to give reasonable predictions for some situations. Research in manual control theory has led to the development of a quasi-linear model of the human operator, shown in Fig. 2. The quasi-linear model is an attempt to represent the human operator as a linear differential equation with internal noise, which is assumed to arise from perceptual or motor processes internal to the human operator [45].

Fig. 3 illustrates a typical 1-D tracking experiment. The human operator, represented as Y with internal noise $n(t)$, is instructed to follow a quasi-random input signal, $r(t)$. The error, $e(t)$, is displayed in a compensatory tracking task. Control responses, $o(t)$, are typically made with a joystick or more generally a controller, *C*, and these control responses are input to a plant (e.g., computer) Y_p. The output of the system, $y(t)$, is the response of the computer.

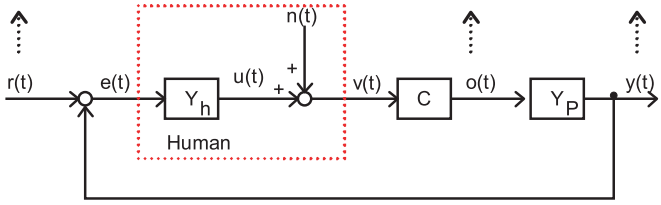

Fig. 3. Typical 1-D compensatory tracking task. Adapted from [45].

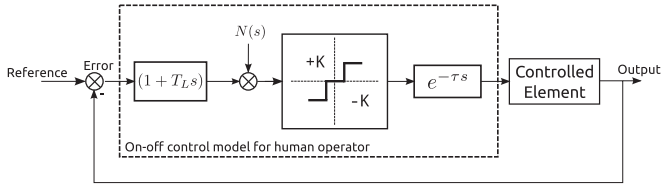

Fig. 4. Young and Meiry's ON–OFF control model for the human operator.

C. Hybrid Automata and Human Operator Models

Hybrid automata are formal models to describe systems that contain both continuous dynamics and discrete switch logic [46]. A hybrid automaton is a finite-state machine with a set of continuous variables whose values are described by ordinary differential equations.

In behavior-based robotics, a robot has a library of useful controllers, called behaviors [9]. The robot can switch between these behaviors based on how its environment changes, for example, switching from going-to-goal behavior to avoiding obstacles. These switched systems are modeled as hybrid automata. This suggests that human operator behavior can also be modeled using hybrid automata.

Oishi *et al.* [47] modeled a hybrid system for semiautomated aircraft landings. It was used to verify that the cockpit interface provides the pilot with enough information to safely decide between landing a plane or performing a go-around manoeuvre. Hybrid automata were used to model the nonlinear aircraft dynamics, but the pilot as controller did not form part of the model.

Doherty and Massink [48] explored the use of hybrid automata for the specification and analysis of interactive systems, modeling the discrete and continuous aspects of the system itself, but only the discrete aspects of the user. They suggested that dynamic systems theory could be used to describe the complex dynamic behaviors of both the system and the user, using a combination of hybrid automata and manual control theory. This is the approach we explore in this paper. Even though automata theory is commonly used in the field of HCI [49]–[51], hybrid automata have not yet been explored for its applications in HCI to model both discrete and continuous interaction.

In this paper, we present a technique based on hybrid automata where discrete and continuous quantities of an interactive system (human operator and device) are taken into account. Our technique uses well-established controllers in manual control theory [45], [52] to simulate the user's conceptual model of the user interface, i.e., an infusion pump with four chevron keys. We define these automata as *behavior-based hybrid automata*, where each state in the automaton describes a different type of behavior used by the human operator, defined as continuous, discrete, and fine-tuning behavior.

In a closed-loop interaction between a user and a device, when simulating human operator behavior, the device behavior also needs to be simulated. Device models of infusion pumps created for software verification purposes can be plugged into our model. In the remainder of this paper, we discuss the implementation of our human operator model, a device model that interacts with the human operator model and the results of a model validation study, where the human operator model is compared with a lab study.

III. Modeling the Human Operator

A. Modeling Continuous Behavior

In conventional human operator models, the operator observes a continuous output display and controls the device using a continuous input device like a joystick or a steering wheel. This makes sense for tasks like piloting an aircraft or driving a car and is handled well by the quasi-linear model described in Section II-B. However, a large number of modern tasks consist of short discrete inputs using buttons that are either ON or OFF. In the 1960s, Meiry [52] discovered that a human operator using an ON–OFF control input behaves like an ideal relay, and that the same theory developed for relays and servos [53] could be used to describe and analyze discrete human behavior. Young and Meiry [54] proposed the ON–OFF control model for the human operator as shown in Fig. 4. The operator has a reaction time delay ($e^{-\tau s}$) and the ability to generate lead ($1 + T_L s$), but his/her output is restricted to a three-level switch operation. The remnant noise term $N(s)$ accounts for uncertainty in the triggering of the switch. The output of the human operator model ($+K$, $-K$ or 0) is then fed to the controlled element, that is, the device input. The device output is fed back to the human operator, where the error is the difference between the displayed value and the desired target value.

A delay in a control-theoretic model means there is no response for some initial time interval after the control input is applied, and as such can be used to model reaction time responses. A lag, on the other hand, means that there is a gradual response after a control input is applied. All objects in the physical world exhibit some form of lag or delay [55], as changes in the world are continuous. Interactive devices do not exhibit lags, but they do exhibit delays, usually referred to as latency.

The three-level switch component shown in Fig. 4 can be used to model a single button, but what if there is more than one button, as is the case with a chevron-key interface? As part of their work on developing the theory behind relay controllers, Flügge-Lotz and Taylor [56] created a discontinuous controller that can switch between four discrete levels, as shown in Fig. 5. We can use this second-order controller to simulate the human operator switching between different device inputs, i.e., the small up, small down, big up, and big down buttons, where each device input is mapped to one of the four discrete levels of the controller.

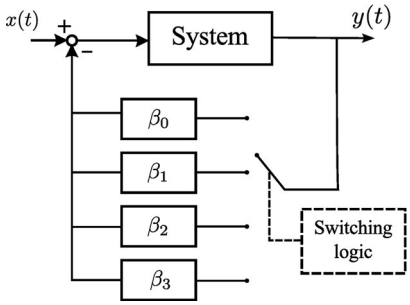

Fig. 5. Discontinuous controller with four levels.

TABLE I
STEPWISE SWITCHING FUNCTIONS MAPPED TO CHEVRON-KEY INPUTS

β_m	β_m value	Chevron key
β_0	-10	"big down"
β_1	-1	"small down"
β_2	1	"small up"
β_3	10	"big up"

In order to represent four discrete values, we make use of a switching criterion incorporating two signum functions as defined by Flügge-Lotz and Taylor [56]:

$$\beta_m = k_1 \text{sgn}(e(t)) - k_2 \text{sgn}\left(\dot{e}(t)\right) \tag{1}$$

where k_1 and k_2 are positive constants, such that $\beta_3 = -\beta_0 = k_1 + k_2$, $\beta_2 = -\beta_1 = k_1 - k_2$, and $\dot{e}(t)$ is the derivative of the error signal $e(t)$. This switching rule has been shown experimentally [53] to give good performance for a variety of inputs including step functions, triangular waves, and random inputs. It switches between levels as to maintain a small instantaneous error between input and output. That is, if the sign of the instantaneous error and the rate of change is the same, small levels (β_1, β_2) will be selected, while a difference in sign will cause the large levels (β_0, β_3) to be selected.

To model the difference in effect that a large chevron key has in comparison with a small chevron key, we set $\beta_3 = 10$ and $\beta_2 = 1$. Solving for β_3 and β_2 in (1) gives

$$k_1 = \frac{11}{2},\ k_2 = \frac{9}{2}. \tag{2}$$

This gives us four stepwise switching functions that we map to the four chevron keys as shown in Table I.

B. Switching Between Continuous and Discrete behavior

We can model both the discrete and continuous behavior of a human operator using a behavior-based hybrid automaton, as shown in Fig. 6.

The error signal $e(t)$ is the difference between the displayed value and the reference value x_{ref}. When the system is initialized, the error signal is as large as the set point. Switching between continuous behavior and discrete behavior depends on the size of the error signal. We only consider switching at specific intervals, that is, when $t \bmod \tau = 0$, as the simulation depends on the interval period of the device model (see Section IV for more information).

To prevent oscillations for smaller set points, the point at which we switch to discrete behavior needs to be proportional to the size of the set point. For large initial set points, that is, where $x_{\text{ref}} > 100$, we switch from continuous behavior to discrete behavior when $e(t) \leq 100$. For smaller set points, the moment at which we switch changes proportionately to a specified switch sensitivity α and the set point, that is we switch from continuous behavior to discrete behavior when $e(t) \leq \alpha x_{\text{ref}}$.

When operating under continuous behavior, we use the dynamics

$$u(t) = k_p\Big(k_1 \text{sgn}\big(e(t)\big) - k_2 \text{sgn}\big(\dot{e}(t)\big)\Big) \tag{3}$$

where $u(t)$ is the input signal, k_p is the controller gain, and the Flügge–Lotz equation in (1) is used as the stepwise switching function. As derived in the previous section, we set $k_1 = \frac{11}{2}$ and $k_2 = \frac{9}{2}$. Under these conditions, switching between the four buttons is equally likely and depends on both the current and the previous error.

When operating under discrete behavior, we use the dynamics

$$u(t) = k_p\Big(k_1 \text{sgn}\big(e(t)\big) - k_2 \text{sgn}\big(\dot{e}(t)\big)\Big)\Pi(t - n\tau) \tag{4}$$

where we introduce the term $\Pi(t - n\tau)$. The rectangle function $\Pi(x)$, also called the gate function, pulse function or window function, is defined by

$$\Pi(x) = \begin{cases} 0, & \text{for } |x| > \dfrac{1}{2} \\ \dfrac{1}{2}, & \text{for } |x| = \dfrac{1}{2} \\ 1, & \text{for } |x| < \dfrac{1}{2}. \end{cases} \tag{5}$$

$\Pi(x)$ is 0 outside the interval $[-\frac{1}{2}, \frac{1}{2}]$ and unity inside it. We use a periodic version of $\Pi(x)$ to generate a pulse train (also called a pulse wave) that simulates the human operator pressing and releasing a button in rapid succession. n is incremented for each iteration of the simulation.

We define $c(t)$ to be the number of times the output $y(t)$ crosses the set point value x_{ref}. If the output value overshoots the target value three or more times (such that the number of crossings $c(t) > 2$), and the error is small ($e(t) < 1.0$), a third kind of behavior is simulated which we call *fine-tuning*. It is simulated using the dynamics

$$u(t) = k_p(k_1 - k_2)\text{sgn}\big(e(t)\big)\Pi(t - n\tau). \tag{6}$$

This means only the small chevron buttons are utilized, as $k_1 - k_2 = \beta_2 = -\beta_1$ correspond to the small chevron buttons in Table I, and corrections are made based only on the current error. This is to simulate the heuristic used by human operators to use finer-grained control when overshooting multiple times.

C. Modeling Noise and Delay

Any reasonable mathematical model of the human operator must include the various psychophysical limitations, or

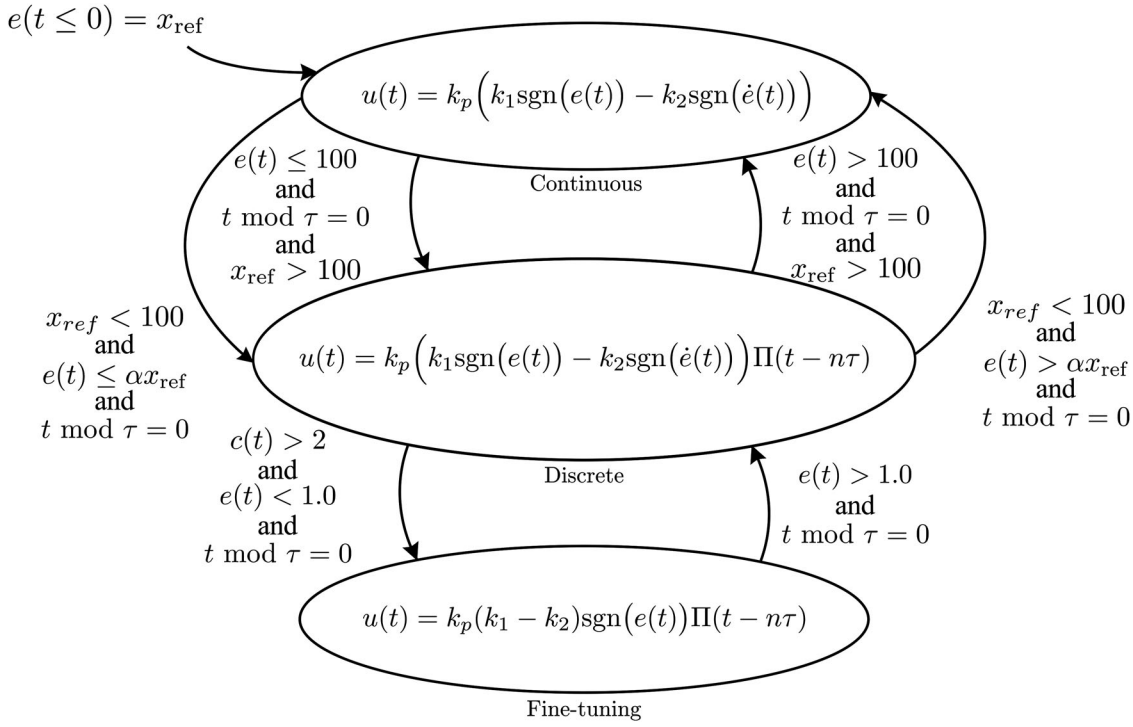

Fig. 6. Behavior-based hybrid automaton describing discrete, continuous, and fine-tuning behavior of human operator using a chevron-key interface.

errors, in observing the display output and executing control inputs [57]. *Noise* in this context means the random fluctuation or disturbance in a signal. As described in Section II-B, a noise signal is used to model the imperfect perceptual processes, called observation noise, and the imperfect motor processes, called control noise, that are internal to the human operator. This noise is assumed to be uncorrelated with the input signal. Sheridan and Ferrell [58] reported observation noise to be typically about 3% and control noise to be typically around 1% of their respective signal variances.

There are various time delays associated with the visual central processing and neuromotor pathways of the human operator [57]. There is a finite time during which information is processed, given as the time for the perceived signal to be transformed and communicated to the effector, that is the hand on the input device [45]. To model this reaction time delay, we delay $u(t)$ by a variable number of time steps within a specified range. Sheridan and Ferrell [58] reported the time delay to be typically around 150 ms, with Kleinman *et al.* [57] describing values between 150 and 250 ms.

The implemented human operator model, combining the ON–OFF control model from Section III with the behavior-based hybrid automaton from Section III-B, is shown in Fig. 7. The reference signal, x_{ref}, is the target value that the operator is trying to reach. This target value is usually static, but can also change over time. The error $e(t)$ is the difference between the perceived value on the output display and x_{ref}. The noise source described above is added to the error signal, which is then passed into the hybrid automaton. The value generated by the hybrid automaton and delayed by the reaction time delay described above is the control signal that is sent to the device. The output of the device completes the feedback loop.

IV. Modeling Device Behavior

When simulating human operator behavior, the device behavior also needs to be simulated. We wanted to model user interaction with medical devices utilizing chevron-key interfaces. Fortunately, device models of infusion pumps created for software verification purposes can be plugged into our model.

Modeling of devices first involves developing a version that can be analyzed using model checking, and then transformed systematically into a form that is analyzed using theorem proving [59]. This second analysis is done using Prototype Verification System (PVS), a state-of-the-art theorem prover. The device model, a PVS specification, is then validated against the physical device using a combination of plausibility properties and simulation.

PVSio-web [60] is a tool used for the rapid prototyping of device user interfaces in PVS. It makes use of PVSio, a component of PVS that allows for the exploration of the behavior of a PVS specification. The backend of PVSio-web executes PVS and PVSio on-demand, type-checking both the PVS specification of the device interface as well as executing the PVS specification according to PVSio commands.

To connect the device model to our human operator model, we use the standardized WebSocket protocol [61], enabling bidirectional communication between the human operator model and the PVSio-web tool, as shown in Fig. 8. As such, the device behavior is simulated within PVSio-web and the user behavior

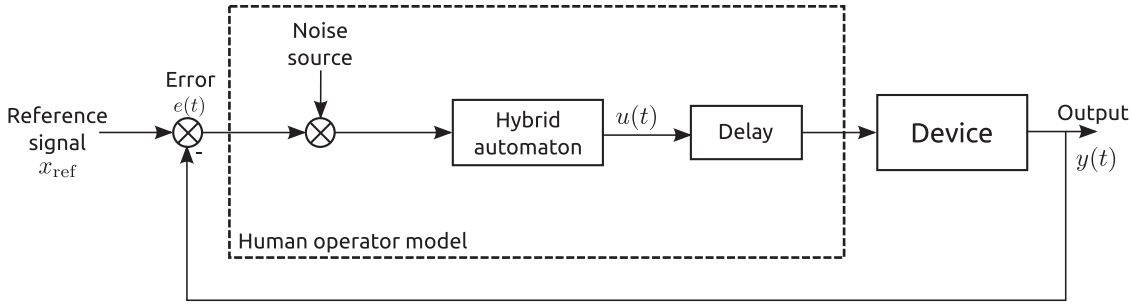

Fig. 7. Human operator model, combining the ON–OFF control model with a behavior-based hybrid automaton.

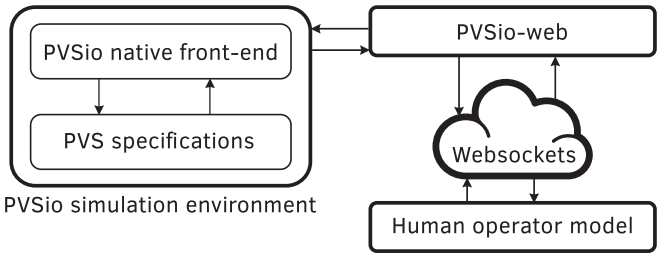

Fig. 8. Architecture of device model, showing links to human operator model using WebSockets.

is simulated within our human operator model. This allows us to make use of different device models that have been developed for use with PVSio-web. Key presses from the human operator model are translated into PVSio commands and used as input to the device model, with the device display output generated by the device model used as input to the human operator model. PVSio-web performs the functionality required by the *Device* component in Fig. 7.

V. Experimental Setup

The model was implemented in the Python programming language using a minimalistic software framework for feedback control by Janert [55]. The implementation is made available as an open-source tool [62].

To initialize the human operator model, we set $u = 0$ and the switch sensitivity $\alpha = 0.1$. We set $k_p = 1$, $k_1 = 5.5$, and $k_2 = 4.5$ using the results from (2). Parameters were selected based on the experimental results of an arbitrarily chosen number. The noise follows a normal (Gaussian) distribution with mean $\mu = 0$ and standard deviation $\sigma = 10$. To smooth the noise, a finite impulse response filter with nine filter coefficients and a cutoff frequency of 0.1 Hz is used. For setpoints with two decimal places, this noise level is too high and the standard deviation is reduced to $\sigma_{\text{small}} = \frac{\sigma}{10} = 1$. A variable delay of maximum two time steps was used, and the simulation ran for a maximum of 160 time steps.

The device model used is that of a commercial syringe pump with chevron keys described in [60] and derived as a PVS specification in [63]. Each time step in the device model is assumed to be 220 ms. The device model was running on an instance of PVSio-web, on the same computer running the human operator model simulation.

Fig. 9. Physical prototype used in the lab study.

The numbers used in the experiment were obtained from the log files of 60 syringe pumps located in the university hospital. The log files were anonymous and contained no personal information. Twenty numbers were selected randomly from the logs, where all numbers had a decimal part and ranged from 0.26 to 83.3.

A. Validation Study

In [2], Oladimeji *et al.* performed a lab study where the participants entered the numbers from the log files on a physical prototype shown in Fig. 9. It serves as a comparison for the human operator model. Thirty-three participants (22 female) took part in the study, with five participants (15%) indicating that they were familiar with the chevron interface from use in digital stop watches and alarm clocks. Each participant had a training session where they tried using the interface by entering ten numbers. The experiment involved entering 20 numbers,

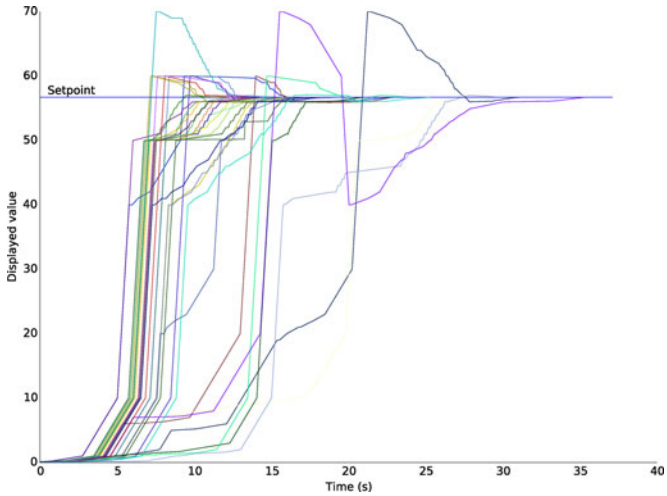

Fig. 10. Entering a large set point (56.7) by 33 participants during the lab study, plotted on same axes.

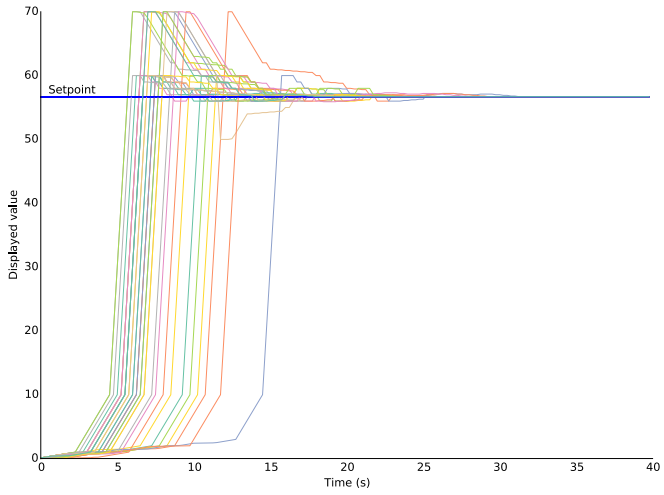

Fig. 11. Entering a large set point (56.7) with 33 iterations of the simulation, plotted on same axes.

with instructions given to emphasize either speed of entry or accuracy having no statistically significant effect.

Interactions logged on the interface included button presses and releases as well as incremental changes of the numeric value, while a button is being held down. For each interaction logged, a timestamp, the button receiving the interaction and the value on the screen were recorded. This allows us to compare the human operator model with participants over the duration of entering a number using the chevron-key interface.

VI. Results

In Fig. 10, the results of the lab study for a large set point (56.7) is shown. Thirty-three participants are plotted on the same axes, showing a maximum overshoot of 70 and a mean time to finish of 19.7 ± 5.9 s. For the same set point, a simulation of 33 iterations is shown in Fig. 11, also showing a maximum overshoot of 70 with a mean time to finish of 21.4 ± 5.5 s.

In Fig. 12, the results of the lab study for a small set point of 6.7 is shown. The maximum overshoot is 8 and the mean time

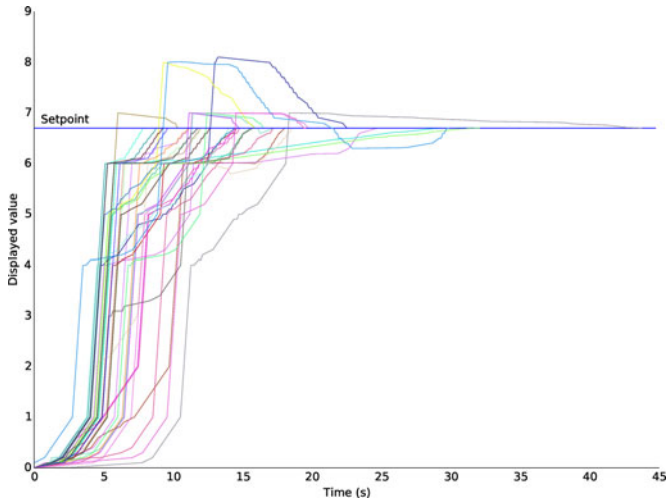

Fig. 12. Entering a small set point (6.7) by 33 participants during the lab study, plotted on same axes.

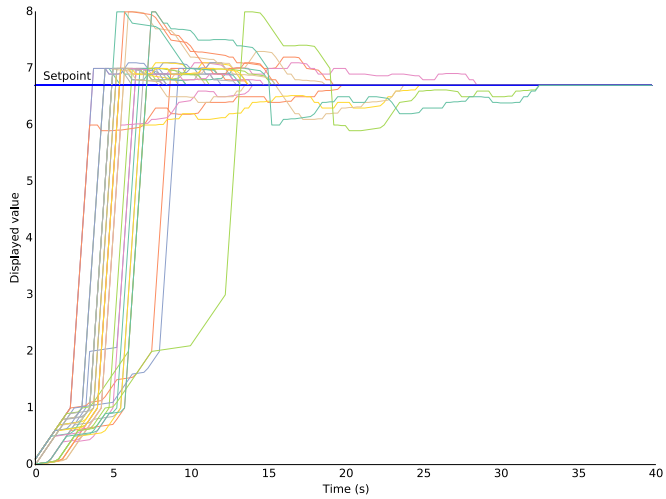

Fig. 13. Entering a small set point (6.7) with 33 iterations of the simulation; plotted on same axes.

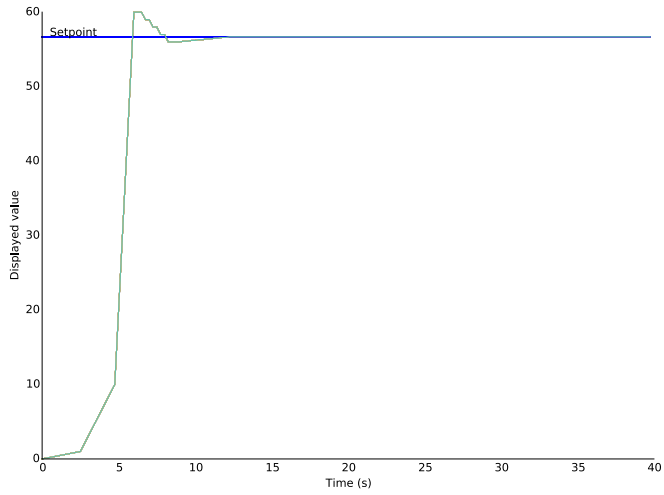

Fig. 14. Entering a large set point (56.7) with 33 iterations of simulation; no simulated delay or noise.

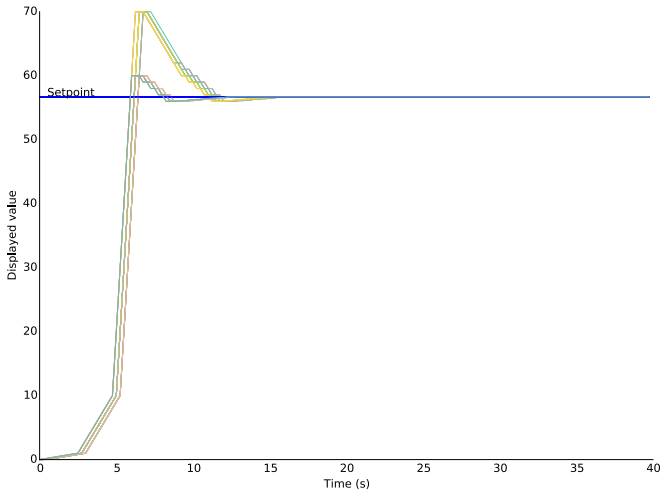

Fig. 15. Entering a large set point (56.7) with 33 iterations of simulation; with delay and no noise.

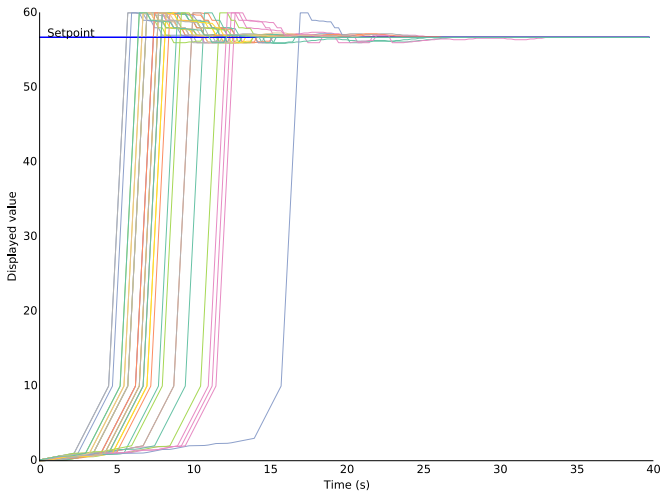

Fig. 16. Entering a large set point (56.7) with 33 iterations of simulation; with noise and no delay.

to finish is 18.2 ± 7.7 s. For the same set point, the simulation run is shown in Fig. 13, with the same maximum overshoot of 8 and a mean time to finish of 16.8 ± 7.5 s.

For a simulation run of 33 iterations and a set point of 56.7, with no reaction time delay and no noise introduced, the results are shown in Fig. 14. This simulates an "ideal" operator that performs with no error and with no reaction time delay and shows the minimum time and steps required to enter the value, for the dynamics used. To further demonstrate how the model works, the results of a simulation run of 33 iterations with only a delay, as well as only noise, are shown in Figs. 15 and 16, respectively.

Fig. 17 compares the simulation results to the lab study results for all 20 numbers, showing the mean time to finish and its standard deviation over 33 trials in a grouped bar plot.

Table II shows the *maximum overshoot range* of each number, which is the maximum value displayed by the device while entering each number over the range of 33 trials.

VII. Discussion

With black-box validation, the overall behavior of the simulation model is considered, and validation is performed by comparing the model to the real world. If confidence is to be placed in a model, the outputs should be sufficiently similar to a real-world system when the same inputs are used [64]. When we compare the output of the model for a specific input, like the value 56.7, we see that 33 iterations of the simulation (see Fig. 11) are qualitatively similar to the real-world setup shown in Fig. 10. This is also the case for the smaller value 6.7, by comparing Fig. 13 (simulation) with Fig. 12 (real world). To calculate a confidence interval, we use

$$\overline{X}_S - \overline{X}_R \pm t_{2n-2,\alpha/2}\sqrt{\frac{S_S^2 + S_R^2}{n}} \tag{7}$$

where $\overline{X}_S$ and $\overline{X}_R$ are the simulation and real-world output mean, respectively, S_S and S_R are the standard deviations of the simulation and real-world output, respectively, n is the number of observations, and $t_{2n-2,\alpha/2}$ is the value from the Student's t-distribution with $2n-2$ degrees of freedom and a significance level of $\alpha/2$ [64]. This gives us a 95% confidence interval of -1.105 to 4.505 when using time to finish as the metric.

Comparing the mean time to finish of the simulation results with that of the lab study indicates similarity for the numbers used. When comparing 33 trials of a single number, both large numbers such as 56.7 (shown in Figs. 10 and 11), as well as smaller numbers like 6.7 (shown in Figs. 12 and 13) show very similar results over the 33 trials.

Looking at the lab study results in isolation, one may consider that some users are using a strategy of intentionally overshooting the target value and then reducing the displayed value. As the simulation model does not have any such programmed behavior or "intention," one would expect the amount of overshoot to be less. For example, one may consider that overshooting to 30.0 and then reducing to 12.5 may be a strategy. However, Table II shows that for almost half of the numbers, the maximum overshoot range is the same for both the lab study and the simulation. Comparing Figs. 10 and 11 for 56.7, as well as Figs. 12 and 13 for 6.7, shows the same amount of maximum overshoot for both values (70.0 for 56.7 and around 8.0 for 6.7). This could indicate a flaw with the design of the user interface itself, as it looks to be quite common for someone to overshoot significantly, reaching 8 when the target value is 6.7, or from 50 to 70 instead of 60 when the target value is 56.7.

The mean maximum number of crossings for all the numbers in the lab study is $c_{\max} = 2.7 \pm 1.2$. This was calculated by counting the largest number of crossings of the 33 trials for each number. $c_{\max}$ was used to determine the transition condition $c(t) > 2$ as the maximum number of crossings before the simulation goes from discrete behavior to fine-tuning behavior, as in Fig. 6.

Chevron-key interfaces show a high degree of idiosyncrasy as implemented in current systems. When holding down a key, the rate of change or *velocity* of the displayed value changes quite abruptly, as can be seen during the first 5 s of the simulation in Fig. 14. During this time, there are two abrupt changes in how

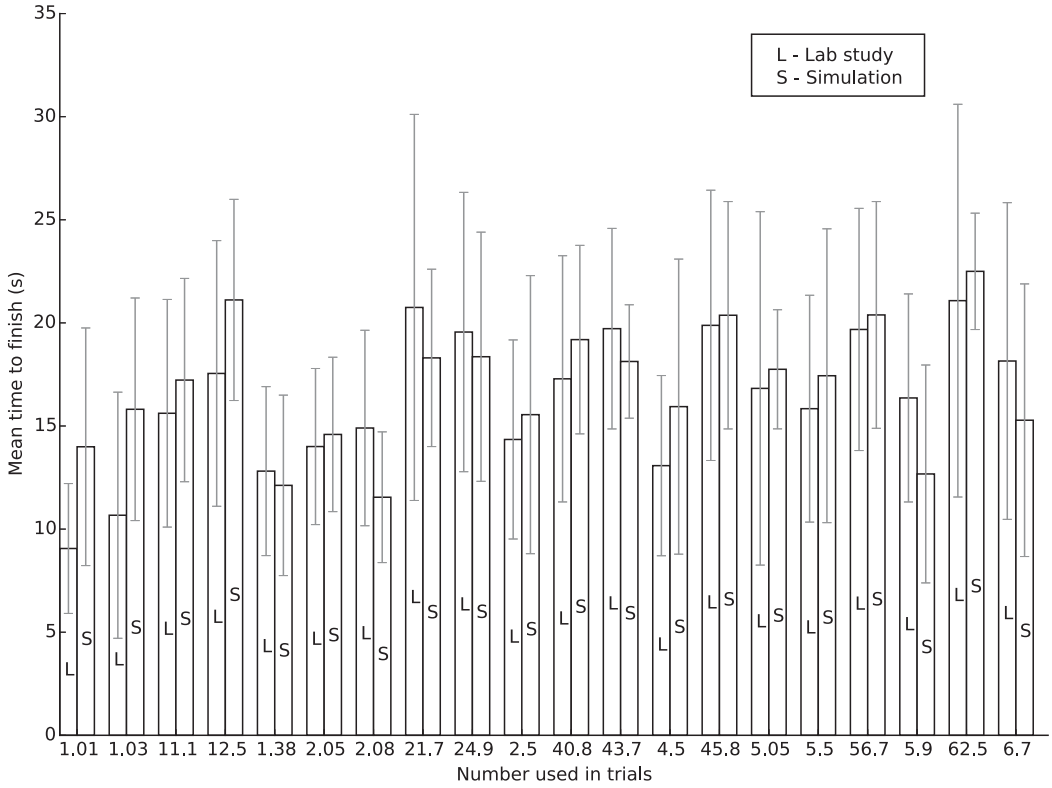

Fig. 17. Comparing simulation to lab study over 33 trials for 20 different numbers. Error bars represent standard deviation.

TABLE II
MAXIMUM OVERSHOOT RANGE FOR EACH NUMBER

Number	Lab study	Simulation
1.01	3.0	2.0
1.03	3.0	3.0
11.1	20.0	30.0
12.5	30.0	30.0
1.38	4.0	3.0
2.05	3.0	3.0
2.08	6.0	3.0
21.7	40.0	40.0
24.9	50.0	40.0
2.5	5.0	4.0
40.8	70.0	50.0
43.7	60.0	50.0
4.5	6.0	6.0
45.8	90.0	60.0
5.05	6.0	6.0
5.5	7.0	7.0
56.7	70.0	70.0
5.9	10.0	8.0
62.5	70.0	70.0
6.7	8.1	8.0

quickly the display changes, where the internal multiplier first switches from $0.1\times$ to $1\times$ (around the 2 s mark), and then from $1\times$ to $10\times$ (at the 5 s mark).

Figs. 14–16 reveal how the human operator model is able to simulate human behavior. With no simulated delay or noise (see Fig. 14), the model behaves like an "ideal" operator, with minimal overshoot and quickly settling on the correct value. When the variable reaction time delay is introduced (see Fig. 15), there is a larger tendency to overshoot, as is the case with actual human behavior (see Fig. 10). When noise is simulated (see Fig. 16), there is a larger variance in how fast the task is completed, similar to the effects of perceptual and motor noise inherent in human behavior.

VIII. CONCLUSION

Even a well-designed user study may not be able to capture all the potential issues with a specific user interface design. Running user studies for a sufficiently long time to uncover these issues can be prohibitively time-consuming and expensive. A combination of user studies and simulations during the design process could yield better designs. The human operator model presented here is intended to be used to evaluate the first iterations of a design, when the design space is still quite large, and there are many design choices to be considered. The model helps to fine-tune variable parameters on the user interface itself, for instance, to reduce the time required to enter numbers on the interface. Once the poor design choices have been eliminated, the best designs can then be evaluated using traditional user studies.

The human operator model presented in this paper allows for the modeling of both discrete interaction events, for example, short button presses, as well as continuous interaction, where the user exchanges input and output of dynamic information with the device constantly over a period of time. The human operator model can be used to compare existing interfaces against

simulations of interfaces that more closely mimic real-world physics, as these continuous and discrete dynamics can be simulated in the model.

Chevron-key interfaces as currently implemented have quite abrupt changes in how quickly the displayed value changes. These abrupt changes in the velocity at which the value increases does not fit with most users' conceptual models, where velocity increases gradually based on real-world physics. Human operators are well adapted to sensing and predicting physical changes. Human perception of velocity is enhanced when a control device has viscous resistance, as viscous resistance is linearly related to velocity [65]. While this is described in terms of proprioceptive feedback with input device like joysticks and computer mice, these concepts can potentially also be applied to visual or auditory feedback. One simple low-cost solution to the problems with existing chevron-style medical devices would be to modify the firmware to use a model that more closely mimics real-world physics [66], where the increase in velocity is gradual and fits better with the user's conceptual model.

Lab studies help in making sure that users understand an interaction design, while simulations help in designing the fine details of the user interface that are essential in safety-critical design, such as those of medical devices [3]. The human operator model presented here combines manual control theory with behavior-based hybrid automata to simulate both continuous and discrete interaction, enabling us to simulate aspects of user interaction at a high resolution that compares well to real-world data. It can be extended and modified for different use cases and can be connected to a variety of device models, including ones based on formal specifications.

ACKNOWLEDGMENT

The authors would like to thank P. Oladimeji and P. Masci for their comments on an earlier draft of the article and R. Oliveira for her comments on the formal methods and model checking approaches in HCI.

REFERENCES

[1] P. Eslambolchilar, J. Webster, and G. Niezen, "The evolution of number entry: A case study of the telephone," in *Proc. 13 Int. Conf. Human-Comput. Interaction*, 2013, pp. 538–545.

[2] P. Oladimeji, H. Thimbleby, and A. L. Cox, "A performance review of number entry interfaces," in *Proc. 13 Int. Conf. Human-Comput. Interaction*, 2013, pp. 365–382.

[3] A. Cauchi, P. Oladimeji, G. Niezen, and H. Thimbleby, "Triangulating empirical and analytic techniques for improving number entry user interfaces," in *Proc. ACM SIGCHI Symp. Eng. Interactive Comput. Syst.*, Rome, Italy, 2014, pp. 243–252.

[4] W. B. Rouse and D. Gopher, "Estimation and control theory: Application to modeling human behavior," *Human Factors*, vol. 19, no. 4, pp. 315–329, 1977.

[5] D. McFarland, *Feedback Mechanisms in Animal Behaviour*. London, U.K.: Academic, 1971.

[6] G. Leacock and D. Thomson, "Estimation of pilot model parameters for helicopter handling qualities studies," presented at the 21st Congr. Int. Council Aeronaut. Sci., Melbourne, Australia, 1998.

[7] D. T. McRuer and H. R. Jex, "A review of quasi-linear pilot models," *IEEE Trans. Human Factors Electron.*, vol. HFE-8, no. 3, pp. 231–249, Sep. 1967.

[8] G. Faconti and M. Massink, "Continuous interaction with computers: Issues and requirements," in *Proceedings of Universal Access in HCI*. New Orleans, LA, USA: Lawrence Erlbaum, 2001, pp. 301–304.

[9] M. Egerstedt, "Behavior based robotics using hybrid automata," in *Hybrid Systems: Computation and Control* (ser. Lecture Notes in Computer Science), N. Lynch and B. H. Krogh, Eds. Berlin, Germany: Springer, Feb. 2000, pp. 103–116.

[10] E. J. Bass, K. M. Feigh, E. Gunter, and J. M. Rushby. (2011, Oct.). Formal modeling and analysis for interactive hybrid systems. [Online]. Available: http://journal.ub.tu-berlin.de/eceasst/article/view/659

[11] J. Campos and M. Harrison, " Formally verifying interactive systems: A review," *Proc. Eurographics Workshop Des., Specification Verification Interactive Syst.*, 1997, pp. 109–124.

[12] G. D. Abowd, H.-M. Wang, and A. F. Monk, "A formal technique for automated dialogue development," in *Proc. 1st Conf. Designing Interactive Syst.: Processes, Practices, Methods, Techn.*, 1995, pp. 219–226.

[13] F. Paternó, "Formal reasoning about dialogue properties with automatic support," *Interacting Comput.*. vol. 9, no. 2, pp. 173–196, 1997.

[14] J. Campos and M. Harrison, " Systematic analysis of control panel interfaces using formal tools," in *Interactive Systems. Design, Specification, and Verification* (ser. Lecture Notes in Computer Science), T. Graham and P. Palanque, Eds. Berlin, Germany: Springer, 2008, pp. 72–85.

[15] A. Hussey, I. Maccoll, and D. Carrington, "Assessing usability from formal user-interface designs," in *Proc. Proc. Australian Softw. Eng. Conf.*, 2000, pp. 40–47.

[16] N. Kamel and Y. Ait Ameur, "A formal model for care usability properties verification in multimodal HCI," in *Proc. IEEE Int. Conf. Pervasive Serv.*, Jul. 2007, pp. 341–348.

[17] N. Kamel, Y. A. Ameur, S. A. Selouani, and H. Hamam, "A formal model to handle the adaptability of multimodal user interfaces," in *Proc. 1st Int. Conf. Ambient Media Syst.*, 2008, pp. 3:1–3:7.

[18] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Using formal verification to evaluate human-automation interaction in safety critical systems, a review," *IEEE Trans. Syst., Man Cybern., Syst.*, vol. 43, no. 3, pp. 488–503, May 2013.

[19] J. C. Campos, G. J. Doherty, and M. D. Harrison, "Analysing interactive devices based on information resource constraints. *Int. J. Human-Comput. Stud.*, vol. 72, no. 3, pp. 284–297, 2014.

[20] J. Rushby, "Using model checking to help discover mode confusions and other automation surprises," *Rel. Eng. Syst. Safety*, vol. 75, no. 2, pp. 167–177, Feb. 2002.

[21] R. M. Young, T. R. G. Green, and T. Simon, "Programmable user models for predictive evaluation of interface designs," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1989, pp. 15–19.

[22] A. Blandford, R. Butterworth, and J. Good, "Users as rational interacting agents: Formalising assumptions about cognition and interaction. in *Proc. Eurograph. Workshop Des., Specification Verification Interactive Syst.*, 1997, pp. 45–60.

[23] R. Butterworth, A. Blandford, and D. J. Duke, "Demonstrating the cognitive plausibility of interactive system specifications." *Formal Asp. Comput.*, vol. 12, no. 4, pp. 237–259, 2000.

[24] R. Butterworth and A. Blandford, "The role of formal proof in modelling interactive behaviour." in *Proc. Eurograph. Workshop Des., Specification Verification Interactive Syst.*, 1998, pp. 87–101.

[25] R. Rukšėnas, P. Curzon, J. Back, and A. Blandford, "Formal modelling of cognitive interpretation," in *Proc. 13th Int. Conf. Interactive Syst.: Des., Specification, Verification*, 2007, pp. 123–136.

[26] S. Card, T. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*. Hillsdale, NJ, USA: Erlbaum, 1983.

[27] B. E. John and D. E. Kieras, "Using GOMS for user interface design and evaluation: Which technique?" *ACM Trans. Comput.-Human Interact.*, vol. 3, no. 4, pp. 287–319, Dec. 1996.

[28] P. Cairns and A. L. Cox, *Research Methods for Human-Computer Interaction*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[29] P. Curzon, R. Rukšėnas, and A. Blandford, " An approach to formal verification of human–computer interaction," *Formal Aspects Comput.*, vol. 19, no. 4, pp. 513–550, 2007.

[30] R. Rukšėnas, P. Curzon, A. Blandford, and J. Back, " Combining human error verification and timing analysis," in *Engineering Interactive Systems* (ser. Lecture Notes in Computer Science), vol. 4940, J. Gulliksen, M. Harning, P. Palanque, G. van der Veer, and J. Wesson, Eds. Berlin, Germany: Springer, 2008, pp. 18–35.

[31] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Generating phenotypical erroneous human behavior to evaluate human–automation interaction using model checking," *Int. J. Human-Comput. Stud.*, vol. 70, no. 11, pp. 888–906, 2012.

[32] M. L. Bolton, "Using task analytic behavior modeling, erroneous human behavior generation, and formal methods to evaluate the role of

human-automation interaction in system failure," Ph.D. dissertation, Univ. Virginia, Charlottesville, VA, USA, 2010.
[33] A. Lecerof and F. Paternò, "Automatic support for usability evaluation," *IEEE Trans. Softw. Eng.*, vol. 24, no. 10, pp. 863–888, Oct. 1998.
[34] A. R. Pritchett, "Reviewing the role of cockpit alerting systems," *Human Factors Aerosp. Safety*, vol. 1, pp. 5–38, 2001.
[35] S. Phansalkar, J. Edworthy, E. Hellier, D. L. Seger, A. Schedlbauer, A. J. Avery, and D. W. Bates, "A review of human factors principles for the design and implementation of medication safety alerts in clinical information systems," *JAMIA*, vol. 17, no. 5, pp. 493–501, 2010.
[36] M. L. Bolton and E. J. Bass, "Generating erroneous human behavior from strategic knowledge in task models and evaluating its impact on system safety with model checking," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 43, no. 6, pp. 1314–1327, Nov. 2013.
[37] P. A. Palanque, R. Bastide, and V. Sengès, "Validating interactive system design through the verification of formal task and system models," in *Proc. IFIP TC2/WG2.7 Working Conf. Eng. Human-Comput. Interaction*, 1996, pp. 189–212.
[38] A. Degani and M. Heymann, "Formal verification of human-automation interaction." *Human Factors*, vol. 44, no. 1, pp. 28–43, 2002.
[39] S. Basnyat, P. Palanque, B. Schupp, and P. Wright, "Formal socio-technical barrier modelling for safety-critical interactive systems design," *Safety Sci.*, vol. 45, no. 5, pp. 545-565, 2007.
[40] C. Mitchell and R. Miller, "A discrete control model of operator function: A methodology for information display design," *IEEE Trans. Syst., Man Cybern.*, vol. SMC-16, no. 3, pp. 343–357, May 1986.
[41] M. L. Bolton, R. I. Siminiceanu, and E. J. Bass, "A systematic approach to model checking human & automation interaction using task analytic models," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 41, no. 5, pp. 961–976, Sep. 2011.
[42] M. L. Bolton and E. J. Bass, "A method for the formal verification of human-interactive systems," in *Proc. Human Factors Ergonomics Soc. Annu. Meeting*, 2009, vol. 53, pp. 764–768,.
[43] E. L. Gunter, A. Yasmeen, C. A. Gunter, and A. Nguyen, "Specifying and analyzing workflows for automated identification and data capture," in *Proc. 42nd Hawaii Int. Conf. Syst. Sci.*, 2009, pp. 1–11.
[44] U. Kiencke and L. Nielsen, *Automotive Control System For Engine, Driveline, and Vehicle*. New York, NY, USA: Springer-Verlag, 2000.
[45] R. J. Jagacinski and J. Flach, *Control Theory for Humans: Quantitative Approaches to Modeling Performance*. Evanston, IL, USA: Routledge, 2003.
[46] T. Henzinger, "The theory of hybrid automata," in *Proc. 11th Annu. IEEE Symp. Logic Comput. Sci.*, 1996, pp. 278–292.
[47] M. Oishi, I. Mitchell, A. Bayen, C. Tomlin, and A. Degani, "Hybrid verification of an interface for an automatic landing," in *Proc. 41st IEEE Conf. Decision Control*, 2002, vol. 2, pp. 1607–1613.
[48] G. J. Doherty and M. Massink, "Continuous interaction and human control," in *Proc. XVIII Eur. Annu. Conf. Human Decision Making Manual Control*, 1999, pp. 80–96.
[49] H. Thimbleby, *Press on: Principles of Interaction Programming*. Cambridge, MA, USA: MIT Press, 2007.
[50] A. Dix, M. Ghazali, S. Gill, J. Hare, and D. Ramduny-Ellis, "Physigrams: Modelling devices for natural interaction," *Formal Aspects Comput.*, vol. 21, no. 6, pp. 613–641, Dec. 2008.
[51] H. Thimbleby, A. Gimblett, and A. Cauchi, "Buffer automata: A UI architecture prioritising HCI concerns for interactive devices," in *Proc. 3rd ACM SIGCHI Symp. Eng. Interactive Comput. Syst.*, 2011, pp. 73–78.
[52] J. L. Meiry, "The vestibular system and human dynamic space orientation," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 1965.
[53] D. Graham and D. T. McRuer, *Analysis of Nonlinear Control Systems*. New York, NY, USA: Wiley, 1961.
[54] L. R. Young and J. Meiry, "Bang-bang aspects of manual control in high-order systems," *IEEE Trans. Autom. Control*, vol. AC-10, no. 3, pp. 336–341, Jul. 1965.
[55] P. K. Janert, *Feedback Control for Computer Systems*. Newton, MA, USA: O'Reilly, 2014.
[56] I. Flügge-Lotz, and C. Taylor, "Synthesis of a nonlinear control system," *IRE Trans. Autom. Control*, vol. 1, no. 1, pp. 3–9, May 1956.
[57] D. L. Kleinman, S. Baron, and W. H. Levison, "A control theoretic approach to manned-vehicle systems analysis," *IEEE Trans. Autom. Control*, vol. AC-16, no. 6, pp. 824–832, Dec. 1971.
[58] T. B. Sheridan and W. R. Ferrell, *Man-Machine Systems: Information, Control, and Decision Models of Human Performance*. Cambridge, MA, USA: MIT Press, 1974.
[59] M. D. Harrison, P. Masci, J. C. Campos, and P. Curzon, "Demonstrating that medical devices satisfy user related safety requirements," presented at the 4th Int. Symp. Foundations Healthcare Inf. Eng. Syst., Washington DC, USA, 2014.
[60] P. Oladimeji, P. Masci, P. Curzon, and H. Thimbleby, "PVSio-web: A tool for rapid prototyping device user interfaces in PVS," presented at the 5th Int. Workshop Formal Methods Interactive Syst., London, U.K., 2013.
[61] I. Fette and A. Melnikov. (2011, Dec.). The WebSocket Protocol, RFC 6455 (Proposed Standard). [Online]. Available: http://www.ietf.org/rfc/rfc6455.txt
[62] G. Niezen. (2014, Nov.). InteractionLoops. [Online]. Available: http://dx.doi.org/10.6084/m9.figshare.1254311
[63] P. Masci, R. Ruksenas, P. Oladimeji, A. Cauchi, A. Gimblett, Y. Li, P. Curzon, and H. Thimbleby, "On formalising interactive number entry on infusion pumps," presented at the 5th Int. Workshop Formal Methods Interactive Syst., London, U.K., 2013.
[64] S. Robinson, *Simulation: The Practice of Model Development and Use*. New York, NY, USA: Wiley, 2004.
[65] B. Buxton. (2009, Sept.). "Human performance," in *Human Input to Computer Systems: Theories, Techniques and Technology*, ch. 8. [Online]. Available: http://www.billbuxton.com/inputManuscript.html
[66] P. Eslambolchilar and R. Murray-Smith, "Control centric approach in designing scrolling and zooming user interfaces," *Int. J. Human-Comput. Stud.*, vol. 66, no. 12, pp. 838–856, Dec. 2008.

Gerrit Niezen was born in Pretoria, South Africa, in 1982. He received the B.Eng., B.Eng. (Hons.), and M.Eng. degrees in computer engineering from the University of Pretoria, Pretoria, South Africa, and and the Ph.D. degree in industrial design from the Eindhoven University of Technology, Eindhoven, the Netherlands.

He is currently an Embedded Software Engineer with the Tidepool Project, San Francisco, USA, and previously a Researcher in human–computer interaction with the Department of Computer Science, Swansea University, Wales, U.K. His research interests include continuous interaction and manual control theory, interactive medical device design using open-source hardware, physical computing, and the Internet of Things.

Parisa Eslambolchilar was born in Tabriz, East Azerbaijan, in 1975. She received the B.Eng. degree in hardware engineering from the University of Amirkabir, Tehran, Iran, in 1999, the Masters in Engineering in robotics and artificial intelligence from the University of Tehran, Tehran, Iran, in 2002, and the Ph.D. degree in computer science from Hamilton Institute, National University of Ireland, Maynooth, Ireland, in 2007.

In March 2007, she joined the Computer Science Department, Swansea University, Wales, U.K., as a Lecturer, where in 2013, she became an Associate Professor. Her current research interests include mobile human–computer interaction, dynamic continuous interaction, promoting physical activity and mental wellbeing via ubiquitous computing, sonification, and brain–computer interaction.