Available online at www.sciencedirect.com
ScienceDirect
Cognitive Systems Research 12 (2011) 134–143
Cognitive Systems RESEARCH
www.elsevier.com/locate/cogsys

A memory for goals model of sequence errors

Action editor: David Peebles

J. Gregory Trafton [a,*], Erik M. Altmann [b], Raj M. Ratwani [a]

[a] AI Center, Naval Research Laboratory, United States
[b] Department of Psychology, Michigan State University, United States

Received 20 January 2010; accepted 26 July 2010
Available online 26 January 2011

Abstract

A model of routine sequence actions is developed based on the Memory for Goals framework. The model assumes that sequential action is guided by episodic control codes generated for each step, and that these codes decay with time and can be primed by contextual retrieval cues. These control codes serve a place-keeping function that allows the system to infer the correct next action after performance is interrupted. According to the model, perseveration (repeat) errors occur because an older episodic trace intrudes due to noise in the system. Anticipation (skip) errors occur because of failures in reality monitoring, in which the model believes that it has completed a step it has not. The model predicts that perseveration errors should occur more frequently than anticipation errors, and that perseveration errors should occur in a graded fashion away from the current step. Across two different experiments, these predictions were supported at both a qualitative and a quantitative level.
Published by Elsevier B.V.

Keywords: Routine sequential actions; Errors, Cognitive modeling

1. Introduction

Several researchers have described classes of errors that people make as they perform routine sequential actions (Baars, 1992; Norman, 1981; Reason, 1990). The classes have mainly been derived from diary studies (Reason, 1990) and from neurologically damaged patient studies (Schwartz, Montgomery, Buxbaum, & Lee, 1998), but also in some instances from experimental studies (e.g., post-completion errors: Byrne & Bovair, 1997; Ratwani, McCurry, & Trafton, 2008). This report will focus on *sequence errors*.

Sequence errors occur during routine action and consist of perseverations, omissions/anticipations, and intrusions (Reason, 1984). Perseveration errors are repeats of a previous action, such as putting cream in a cup of coffee more than once. Perseveration errors can be further classified into continuous perseveration errors, which occur when an action is performed over and over, and recurrent perseveration errors, which occur when a previously completed subtask is performed again, usually with one or more intervening subtasks (Sandson & Albert, 1984). Omissions are skipped steps, while anticipation errors are skipped steps that are quickly rectified. For example, an omission error would be completely forgetting to put cream in a cup of coffee, while an anticipation error would be attempting to pour from an unopened container. In practice, it can be quite difficult to differentiate omission and anticipation errors (Cooper & Shallice, 2000). Intrusion errors (sometimes called capture errors) occur when an action comes from a different, usually related, task. For example, a capture error would occur when a person attempts to make a cup of coffee but gets distracted by a tea bag and instead makes tea.

1.1. Previous models of sequential behavior

There are two computational models of routine sequential behavior: the interactive activation network (IAN) model (Cooper & Shallice, 2000, 2006) and the simple

* Corresponding author. Address: NRL, Code 5515, Washington, DC 20375, United States. Tel.: +1 202 767 3479.
E-mail address: greg.trafton@nrl.navy.mil (J.G. Trafton).

1389-0417/$ - see front matter Published by Elsevier B.V.
doi:10.1016/j.cogsys.2010.07.010

recurrent network (SRN) model (Botvinick & Plaut, 2004, 2006).

In the IAN model, different schemas compete for activation. Activation comes from triggers (environmental or context) and source-schemas (related schemas), but a schema will not be activated if it is not over a specific threshold. Thus, while working on a routine task, the selection of a schema is influenced by the current schema and the state of the world. The IAN model suggests that errors are caused by a lack of attentional resources or distraction in normal populations (Cooper & Shallice, 2000; Norman & Shallice, 1986). Variability in attentional resources is instantiated in IAN by noise. In the case of sequence errors, noise has two major effects. First, noise in the system can cause variability in the ordering of schemas that do not have ordering constraints. Second, noise can cause variability in the selection of which schema is selected when multiple schemas are applicable. Both these forms of variability can cause various sequence errors.

The SRN model (Botvinick & Plaut, 2004) has a set of input units that are activated by features of the environment. Activation is passed along the input units to a set of hidden units, which receive recirculated activation. The hidden units then pass activation to a set of output units that then perform an action (fixating an object, pouring an object, etc.). The connection weights encode series of sequential attractors which the trained model tends to follow (Cooper & Shallice, 2006). Errors in the SRN model are made because of noise, which can cause the network to drift to a related task sequence that contains a state (i.e., a sequential attractor) that resembles the next correct one. Thus, an error is made by the SRN model not because an attentional operation has been omitted, but because the model's internal representations have lost information about a previous or current state (Botvinick & Bylsma, 2005; Botvinick & Plaut, 2004).

1.2. The memory for goals model

To understand sequence errors following task interruption, we adapted a model that we have previously used to interpret how subgoals are suspended and resumed in a problem-solving task (Altmann & Trafton, 2002; Hodgetts & Jones, 2006), and which has since been used to interpret the time costs of interruption (Altmann & Trafton, 2002; Hodgetts & Jones, 2006; Trafton, Altmann, Brock, & Mintz, 2003) and factors affecting post-completion error (Li, Blandford, Cairns, & Young, 2008).

The MFG model is based on the hypothetical construct of activation of memory elements—in particular, activation as construed in the ACT-R (Adaptive Control of Thought-Rational) cognitive theory (Anderson, 2007; Anderson et al., 2004). The MFG model inherits two basic processing assumptions from ACT-R. The first is that when central cognition queries memory, memory returns the item that is most active at that instant. The second is that the activation of a given memory element fluctuates noisily from moment to moment about a mean value. The MFG model makes additional assumptions that govern the activation of fine-grained episodic memory codes; these codes were task-related subgoals in the original application of the model, and here are codes linked to individual steps of a procedure that serve a place-keeping function during sequential performance. The first assumption is that, after an episodic code is encoded, its activation automatically decays, such that a retrieval request is more likely to return a recent code than a less-recent code (all else being equal). The second assumption, which is of less direct relevance here, is that an episodic code can also be primed by contextual retrieval cues, thereby overcoming effects of decay (Altmann & Trafton, 2002).

The episodic codes that serve the place-keeping function of interest here are what we refer to as **control codes**, meaning codes that guide performance for the duration of a single procedural step. These codes were introduced in a model of cognitive control that incorporated many of the same activation-related assumptions as the MFG model (Altmann & Gray, 2008). The Altmann and Gray model simulates performance in a task switching context, where on each trial, a task cue is presented that tells the system how to interpret the imperative stimulus that follows. The model interprets the task cue by retrieving its meaning from semantic memory, creates a control code from that meaning, and uses the control code as a basis for interpreting the stimulus and selecting a response. This model takes an unusually fine-grained perspective on moment-to-moment cognitive control, in which even a simple and closely-spaced process like interpretation of a task cue and interpretation of an imperative stimulus must communicate through episodic codes, which are then subject to architectural processes like activation noise and decay. Models constructed in the parent theory, ACT-R, generally do not dissect cognitive control to this level of detail, but this approach accounts for a wide range of behavioral data (Altmann & Gray, 2008). Moreover, performance from a wide range of tasks, ranging from perceptual-motor tasks (Hommel, Müsseler, Aschersleben, & Prinz, 2001) to navigation through information spaces (Altmann & John, 1999), tend to converge on a similar implication, which is that the system encodes fine-grained episodic memories in large numbers as an integral part of ordinary processing.

Here, we assume that processing on each step of a procedure is similarly governed by a control code. On each procedural step, the system consults its declarative knowledge, which we assume is well-learned, uses that knowledge to create a control code in episodic memory, and uses that control code to guide its processing for the duration of that step, including its parsing of the external environment and any responses it must make. This control code essentially represents the information, "this is the step I am currently on", and serves as a reference point for any component process that may have to run in the course of that step. Because control codes decay, the current one will always be the most active (modulo effects of activation noise), so

any process can reliably assume that whatever code it retrieves is the one it should act on.

After an interruption, the system can use a control code to regain its place in the procedure. Here we assume that the interruption most often occurs cleanly between steps—that is, after one step has completed and before the next has begun. This assumption is incorporated in the model, such that, after an interruption, the model assumes that the control code it retrieves was for the most recently **completed** step, and therefore infers that the correct **current** step is the one after that. Occasionally, however, an interruption will occur after the system has created the control code for a new step, but before it can execute that step. In these cases, after an interruption, the model again infers that the control code it retrieves was for the most recently completed step—but in such cases will be incorrect (as described below).

1.3. Accounting for sequence errors

All three models have different process explanations and capabilities for accounting for sequence errors.

1.3.1. Perseveration errors

The IAN model does occasionally repeat steps, resulting in a continuous perseveration error. This occurs when, due to too much self-activation or lack of inhibition, a schema is not deselected at the appropriate time, causing a schema to be repeatedly selected. The IAN model cannot, however, account for recurrent perseveration errors because once a goal is completed it is "ticked off" and not a candidate for later selection (Botvinick & Plaut, 2004; Cooper & Shallice, 2000).

The SRN model does make both continuous and recurrent perseveration errors. However, one interesting aspect of the original SRN model was that virtually all errors were due to a capture process. For example, with a small amount of noise, the network would occasionally drift to a similar sequential attractor (a capture process) and repeat a step (a perseveration error) (Botvinick & Plaut, 2004; Cooper & Shallice, 2006). The fact that all errors use a capture process is both a strength and a weakness for SRN. It is a strength because it shows that a single process can account for a wide range of error types (Botvinick & Plaut, 2004). It is a weakness because there is some evidence that some different error types have different processes (Cooper, 2007; Cooper & Shallice, 2006; Li et al., 2008; Schwartz et al., 1998).

The MFG model explains perseveration errors as follows. After an interruption, the system retrieves an old control code and assumes that it represents the most recently completed step. Most of the time, this assumption will be correct. Sometimes, however, an older control code will intrude, even though it is more decayed than the most recent old code, due to noise in activation levels. In this case, the model will repeat at least one step. For example, if the second-most recent control code is retrieved, the step corresponding to the most recent control code will be repeated.

This mechanism predicts that perseveration errors after interruption will form a gradient in which the most common one is to repeat the most recent step, the second-most common one is to repeat the second-most recent step, and so on.

1.3.2. Anticipation and omission errors

The IAN model also makes anticipation and omission errors. Omission errors occur because a schema may not have a high enough activation due to low self-activation or poor environmental influences. Anticipation errors occur for the same reasons, but the incorrect actions cannot be executed because a precondition is not satisfied (e.g., a container still has its top attached).

The SRN model occasionally makes anticipation and omission errors, primarily through the capture process described before.

The MFG model explains anticipation and omission errors as follows. Again, after an interruption, the system retrieves an old control code and assumes that it represents the most recently completed step. Most of the time, this assumption will be correct. Sometimes, however, the most recent old code will represent an intended step, not a completed step. As we noted earlier, interruptions typically occur between procedural steps, after one step has been completed and the next not yet initiated. However, some interruptions, as a function of noise in timing parameters, occur after the model completes a step, and also after it generates the control code for the next step, but before it performs any cognitive or behavioral actions to implement the next step. In this case, the control code retrieved after the interruption would correctly be the one to execute, but if the model follows its default processing, it will again infer that the code it retrieves has been executed, and as a result skip a step.

This can be viewed as a failure of reality monitoring, in which the model fails to represent a distinction between a step it has executed and a step it only intended to execute. Indeed, the model has no explicit representation that distinguishes between intended and executed actions; there are only control codes, which, after an interruption, the system assumes represent completed actions. This is likely an oversimplification relative to task environments that involve lookahead search or some other kind of mental simulation, where mental and physical states could more easily become confused. Here, we assume that one step of lookahead occurs on some proportion of interruptions, in effect, but that when this happens it is accidental and the system has no memory of it.

While all three models can account for the majority of error types, neither IAN nor SRN makes strong predictions about which types of errors should be more prevalent in this type of task. MFG, however, makes a strong prediction that perseveration errors should occur more often than any other type of sequence error. Additionally, MFG

makes a nuanced prediction that errors should be graded from the correct step, especially with respect to perseveration errors.

2. Experiment 1

There are very few existing datasets that can be used to constrain or reject these different models (Botvinick & Plaut, 2006). One of the issues is that when a task is routine, people generally make very few errors, making statistical analysis difficult. Thus, different researchers have examined errors in non-routine tasks (Ruh, Cooper, & Mareschal, 2005), made the task difficult to remember (Giovannetti, Schwartz, & Buxbaum, 2007; Ruh, Cooper, & Mareschal, 2008) or interrupted participants during the routine task (Botvinick & Bylsma, 2005). We used an interruption paradigm because interruptions have been shown to increase error rates even on well-learned tasks (Li et al., 2008; Ratwani et al., 2008). In addition, we provided no global place-keeping (Gray, 2000) such that the next step of the task could not be determined from visible cues.

2.1. Method

2.1.1. Participants

Fifteen George Mason University students participated for course credit.

2.1.2. Task and materials

The primary task was a complex production task called the sea vessel task (based on Li et al., 2008; Ratwani et al., 2008). The goal was to fill an order for two different types of sea vessels by entering in order details through various widgets on the interface (Fig. 1). Order information was provided in the middle of the screen on the "Navy Manifest." A correct sequence of actions is required to complete the order: (1) Enter Vessel Information, (2) Material, (3) Paint, (4) Weapons, and (5) Location. Before entering information into each widget, the widget must be "activated" by clicking the corresponding selector button (lower right hand corner of Fig. 1). Completing the information in each widget (i.e., completing one step of the procedure) requires the participant to add the information for both types of sea vessels, and thus involves several perceptual and motor actions.

After completing each widget, the participant must click "ok." This action causes the information that the participant entered into widget to be removed from the display, so that it cannot serve as an explicit cue indicating which steps have been completed.

After the participant has finished entering information into all five widgets, he or she must process the order, by clicking the "Process" button. This action causes a small pop-up window to appear informing the participant of the total number of sea vessels that have been created. This pop-up window served as a false completion signal (Reason, 1990). Participants must click the "ok" button to acknowledge this window. Finally the "Complete Contract" button must be clicked to finish the order. The "Next Order" button is then clicked to bring up a new order. Any deviation from this procedure where a participant attempted to work on an incorrect subgoal was recorded as a sequence error; any time an error at the subgoal level was made, the computer emitted a brief auditory tone to alert the participant that an error was made. When a participant committed an error the participant had to continue with the task until the correct action was made.

The procedure described above was arbitrary, but participants had no trouble learning it because (1) the information that was needed to fill in the widgets was available on the Navy Manifest and (2) the order of the widgets was straightforward to remember due to a simple spatial rule, which we provided to participants.

The interrupting task required participants to answer addition problems with four single digit addends. During the interruption, none of the primary-task display was visible.

2.1.3. Design and procedure

Each order on the sea vessel task constituted a single trial; participants performed twelve trials. Control and interruption trials were manipulated in a within-participants design; half of the trials were control with no interruption and half were interruption trials with two interruptions each. The order of trials was randomly generated. There were six predefined interruption points in the sea vessel task. There was a potential interruption point after clicking "ok" in each of the five widgets. The sixth interruption point was after the "Process" button was clicked. During the experiment there were a total of 12 interruptions (6 interruption trials $\times$ 2 interruptions in each trial); each lasting 15 s. Participants were instructed to answer as many addition problems as possible in this time interval. The interruptions were equally distributed among the six interruption locations. When the primary-task display was reinstated after the interruption, there were no visual cues on the task interface indicating where to resume (i.e. no global place-keeping).

Before beginning the experiment, participants were given instructions about the two tasks they were going to have to perform and completed two trials as part of training; one had no interruptions and one had two interruptions. Following these two training trials, participants had to perform two consecutive randomly selected trials on their own without making a sequence error before the participant could begin the experiment; participants typically took two trials to learn the task to criterion. Forcing participants to perform two consecutive error free trials was a method for ensuring that participants were proficient at the task before beginning the actual experiment. The experiment was self-paced. A break was offered after six trials.

2.1.4. Description of errors

Perseveration errors were any actions that repeated an action that had already been accomplished for that trial.

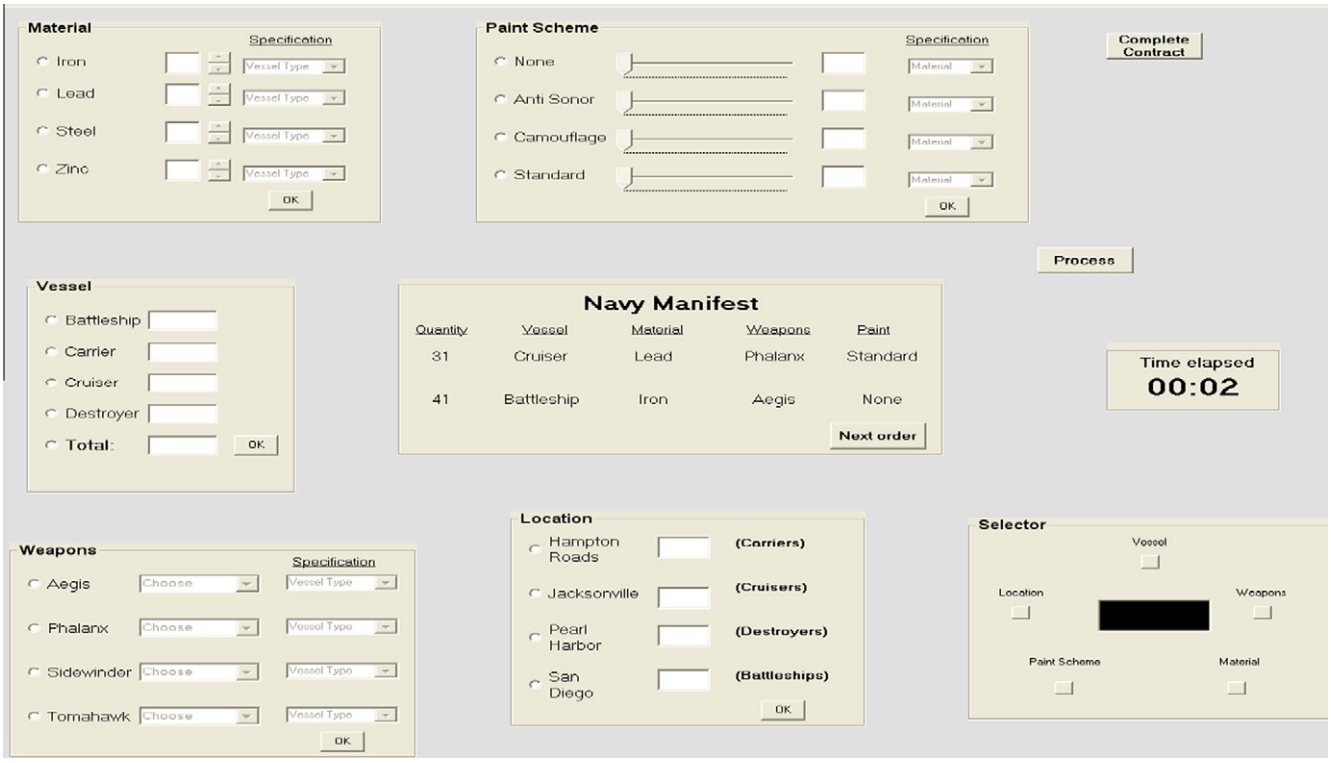

Fig. 1. Screenshot of the ship production task.

Anticipation and omission errors were any actions that skipped one or more steps. Because it was not possible to actually omit a step, all skipped steps were categorized as anticipation errors. Errors that occurred within a widget, such as selecting the wrong material, were not analyzed. Errors where participants failed to activate a particular widget before attempting to work on the widget were not analyzed; these are called device initialization errors (Cox & Young, 2000) and are a different class of errors from sequence errors that are considered to be task specific and driven by the design of the interface.

2.1.5. Measures

Error rates were calculated for control and interruption trials by calculating percentages (actual errors/total error opportunities). Multiple incorrect actions in a sequence were counted as a single error for the purposes of calculating error rates. Error actions that occurred less than 500 ms from the previous action were excluded from all analyses as they were taken to be inadvertent mouse clicks; this accounted for less than one percent of the data.

2.2. Results and discussion

2.2.1. Comparing error rates

Of the fifteen participants, eleven participants made at least one perseveration or anticipation error. To examine the effect of interruptions on error rates, the error rate on the step immediately after an interruption was compared with the error rate on control trials using a repeated measures ANOVA. Participants made more errors immediately following an interruption ($M = 9.3\%$) compared to the control ($M = .9\%$), $F(1, 14) = 5.8$, $\mathrm{MSE} = 91.9$, $p < .05$. Participants rarely made errors in the control trials, suggesting the task was well-learned. The non-zero error rate on control trials also matches studies showing that people do make errors on well-learned tasks (Reason, 1990).

2.2.2. Pattern of error actions

Next, we focused on the pattern of error actions. In order to compare error actions at different points in the task hierarchy, the error actions were coded relative to the correct action at that point in the task hierarchy. Recall that the correct order of actions was Vessel, Material, Paint Scheme, Weapons, Location, Process and Complete Contract. If the next correct action is to work on the "Weapons" subtask and the participant made the error of working on the "Paint" subtask, this error action was coded as a "−1". If instead the participant clicks the "Process" button this was coded as a "2". Based on this coding scheme, a "−1" represents a repeat or perseveration of the just completed action and a "1" represents a skip or an anticipation of the next correct action. All errors were coded using this scheme.

The distribution of error actions is illustrated in Fig. 2. A visual inspection of this graph suggests that both perseveration and anticipation errors occur relatively frequently. Additionally, the number of errors seems to be close to the next correct action in both directions and graded away from the correct step for perseveration errors. To deter-

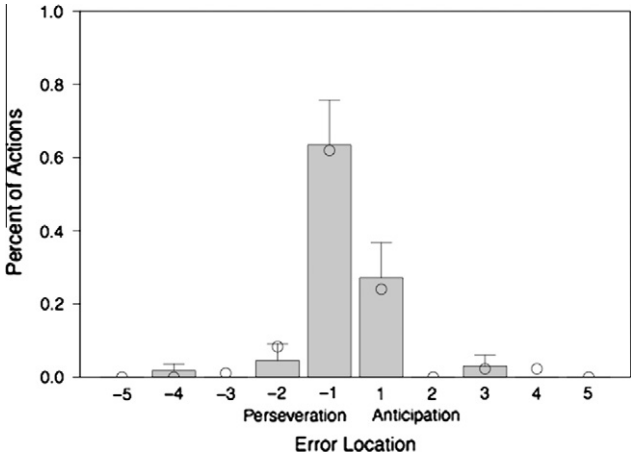

Fig. 2. Distribution of errors during a sequential action task. Bars are empirical data; circles are model fits. Error bars are 95% confidence intervals.

mine whether immediate perseveration errors (repeats) were the most common error action, a repeated measures ANOVA was conducted to compare error actions at this position to all other error actions. There was a significant difference among the different error positions, $F(7,70) = 12.8$, $MSE = 434.2$, $p < .0001$. Tukey HSD post hoc comparisons revealed that participants were significantly more likely to make an immediate perseveration error ($M = 63.5\%$) than to make any other type of error (all p's $< .05$).

2.3. Model description

We developed a cognitive simulation of the MFG model using the ACT-R 6.0 theory and simulation environment (Anderson, 2007). ACT-R is a production system, with each production representing a fine-grained unit of control knowledge. On each system cycle, one production is selected to "fire", meaning do the work it represents, which is often to manipulate the contents of a set of "buffers" that collectively represent a kind of immediate memory or focus of mental attention.

The model interacts with a software simulation of the interface shown in Fig. 1. The model has perception and visual attention productions that search for the widget name associated with the current procedural step (e.g., "Vessel"). It also has motor productions that move the mouse cursor to and press the "ok" button associated with a given widget once processing for that step is complete. The model did not perform the post-completion step (Byrne & Bovair, 1997). When the model completed the last procedural step, the model waited the same amount of time that the post-completion step typically took (about 5 s) in order to simulate the time it took to perform that step. Developing a post-completion model is clearly an important component of this task and is currently under progress.

The model has only a lean representation of the processing that occurs during each procedural step. Human participants, during a given step, have to enter various kinds of information into the widget associated with that step, for each of two different vessel types. For example, in the Material step for the scenario in Fig. 1, the participant would click the radio buttons for Lead and Iron, and select Cruiser and Battleship in the corresponding Specification boxes and the quantity needed. We assume that each of these stages within a procedural step may involve retrieving the control code. The model, having attended visually to the widget for the current step, first creates the control code for that step, then always performs exactly one retrieval of the control code in the processing for that step. The control code begins with a slightly higher than normal activation (the strengthening constraint from Altmann & Trafton, 2002) so that it has a high enough activation to be retrieved in the next step. Retrieval during uninterrupted performance is also facilitated by the mental context through activation spreading to the current control code from the focus of attention (the priming constraint from Altmann & Trafton, 2002); we assume that the mental context that provides this priming is displaced by an interruption, so is not available to facilitate retrieval at task resumption. The environmental context could also prime the control code, but that aspect is not implemented in the current model, given that in this task environment there were no relevant environmental cues on the primary-task display after the interruption.

After the control code is retrieved, the final motor action for that step is performed (pressing "ok" in that widget). The model then transitions to the next step by using the control code it retrieved on the previous step to index its knowledge of the procedure. The model's declarative knowledge of the procedure is stored in ACT-R's long-term declarative memory, with each step being stored as a "chunk" that points associatively to the chunk representing the next step. Eventually, this declarative knowledge can turn into procedural knowledge, but this transition is not represented in this model. For the present modeling effort, we simply assume that knowledge of the procedure is well-learned, to the point where each next step is accessible without error, and that the keeping-track mechanism built on control codes is what reproduces the patterns in the empirical data.

When an interruption occurs, the model detects the change on the computer display. The effect of the interruption is to clear all information about the primary task from the buffers representing the model's mental context. The model does not otherwise perform the interrupting task, but simply "spins" until the primary-task display is reinstated, after 15 s. In this context, we assume that the interrupting task itself does not impact the error rate.

To resume the primary task after the interruption, the model tries to retrieve a control code. Sometimes this retrieval will fail, under conditions described below, and the model will guess a step. Most often the retrieval will succeed, and the memory system will return a control code. The model then interprets this code as the one representing

the step it completed before the interruption, and uses it to index its knowledge of the procedure to determine the next step. Usually this interpretation will be correct, because the most recent control code (1) will usually be most active, because it is the least decayed and (2) will usually represent a completed step, because interruptions are triggered when the "ok" button is pressed, an action that signifies step completion. Occasionally one condition or the other will be violated, causing a sequence error. Activation noise may cause an older control code to be transiently more active than the most recent one and thus intrude on the retrieval, which behaviorally will cause a perseveration error. Variability in timing of internal events may mean that the most recent control code does not correspond to a completed step. Sometimes the motor command to press "ok" will have been delayed relative to central processing, such that, in the moments before the interruption is triggered, central processing continues as usual, retrieving the next procedural step and generating the associated control code. After the interruption, the most recent control code will therefore be one that the system in a sense intended to perform but then had to abort to respond to the interruption. Such situations lead to anticipation errors, where the model tries to skip the "intended" step.

Several model parameters interact to affect the behavior described above, all of them affecting activation dynamics. One is activation noise (ACT-R parameter: ans, set to .03), which governs the variance of the zero-mean logistic noise distribution sampled for the activation of each control code on each system cycle. A second parameter is the decay rate (ACT-R parameter: bll, set to the default value 0.5). A third parameter is priming (ACT-R's: mas parameter set to 3) which provides contextual priming to help retrieve the relevant control code.

To reproduce the empirical data, we ran 100 simulated trials. The fits are shown in Fig. 1 (circle markers).

2.3.1. Model fit

As is evident in Fig. 2, the model matches the data quite well; $R^2 = .99$ and RMSD = 1.9. The model captures the pattern in the data that perseveration errors occur more frequently than anticipation errors and that perseveration errors show a graded pattern away from the correct step. Note that the model does not predict anticipation errors beyond the first step (e.g., +1). The model suggests that any anticipation errors beyond the first step (e.g., +2, +3, etc.) are due to the model's failure to be able to retrieve any control code and guessing or retrieving a previous trial's control code. This latter possibility is actually a perseveration error being manifested as an anticipation error.

2.4. Summary

The MFG model showed an excellent fit to the experimental data. However, as with all model fitting paradigms, it is possible that the model will not generalize to other situations because the cognitive processes, parameters, participants, or experimental task may be idiosyncratic. To explore this possibility, a second task was developed. If the different participants and new task qualitatively replicates the sequence error findings, we assume data is robust and replicable. Additionally, if the existing model can quantitatively fit the new task data, we assume that our model accurately describes the cognitive processes involved in sequence errors.

3. Experiment 2

This experiment used a different and much better interface from Experiment 1.[1]

3.1. Method

3.1.1. Participants

Thirty-six George Mason University undergraduate students, participated for course credit.

3.1.2. Task and materials

Participants performed a computer-based procedural task as the primary task and, similar to Experiment 1, participants were periodically interrupted by a secondary task. The primary task was a complex financial management task. The goal of the task was to successfully fill client's orders for different types of stocks. The orders were to either buy or sell and were presented four at a time at the top of the screen (see Fig. 3). The current prices of the stocks associated with the orders were presented in the center of the screen in the Stock Information column. The actual stock prices fluctuated every 45 s (see Fig. 3).

To complete an order, participants first had to determine which of the client orders could currently be executed by comparing the client's requested price to the actual market price of the stock from the Stock Information column. If the client's order is to buy a stock, the current stock price must be equal to or less than the requested buy price for the order to be executable. If the client's order is to sell a stock, the current stock price must be equal to or greater than the requested sell price for the order to be executable. Once an order was determined to be executable, the participant clicked the *Start Order* button for the respective order. To actually fill the order, the participant had to enter details from the order itself and the Stock Information column into eight different widgets on the screen. Participants had to follow a specific procedure to complete the order; the specific sequence was: Quantity, Cost, Order Info, Margin, Stock Exchanges, Transaction, Stock Info, and Review. Overall, the financial management task has a considerably better interface compared to the sea vessel task. In particular, the spatial layout of the interface (working from top to bottom down the left column and then the

[1] This experiment was also reported in Ratwani and Trafton (in press). The focus of that report was on post-completion errors and there is no analysis overlap.

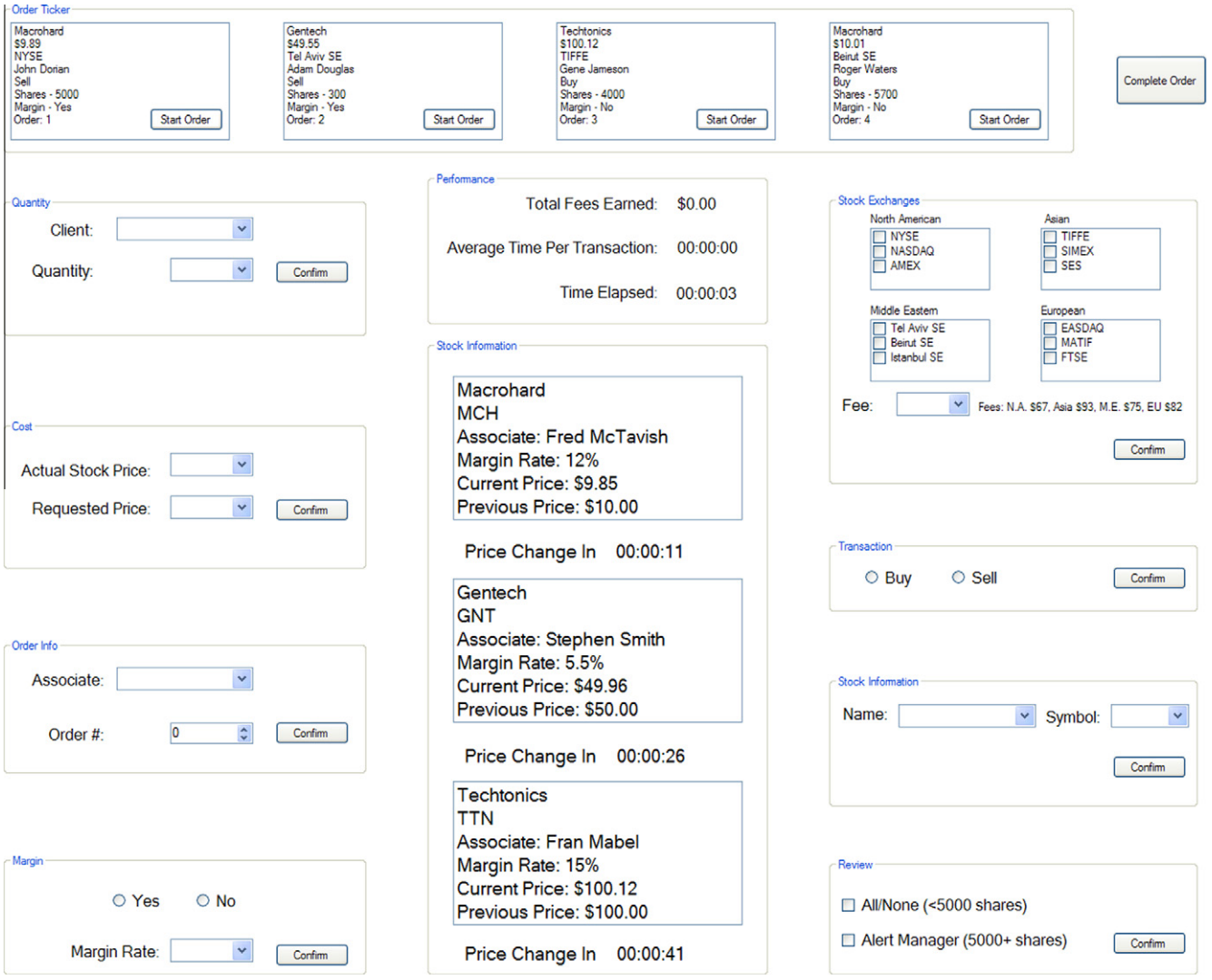

Fig. 3. Screenshot of the financial management task.

right column of Fig. 3) and the operations required to perform the task are quite intuitive. After entering information in each widget, the participant clicked the *Confirm* button and could then move on to the next widget. After clicking confirm on the final widget (the Review widget), a pop-up window appeared confirming the details of the order. The participant then had to acknowledge the window by clicking *Ok*. Finally, to complete the order the participant clicked the *Complete Order* button (upper right corner).

Similar to Experiment 1, if a participant deviated from the strict procedure, the computer emitted a beep signifying that an error had been made and the participant had to continue working until the correct action was completed. No information remained on the interface after entering information in the widgets (i.e. no global place-keeping, Gray, 2000).

The interrupting task was the same as Experiment 1 with the exception that participants were presented with five possible solutions (4 incorrect, 1 correct) and had to click on the button associated with the correct solution.

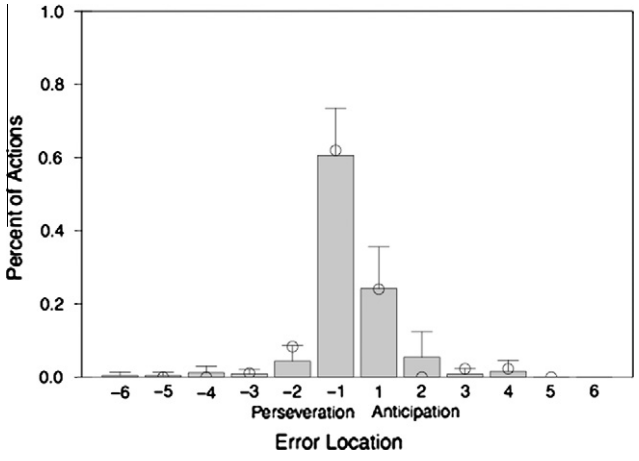

Fig. 4. Distribution of errors during the financial planning task. Bars are empirical data; circles are model fits. Error bars are 95% confidence intervals.

3.1.3. Design and procedure

The completion of one order on the financial management task constituted a trial. The design of control and interruption trials was the same as Experiment 1, except that there were eight possible interruption points in the financial management task. These points occurred after clicking the *Confirm* button following the first seven widgets and after acknowledging the false completion signal, just prior to the post-completion action. The location of the interruptions on a trial by trial basis was randomized with the constraint that exactly two interruptions occurred just prior to the post-completion step and at least one interruption occurred at each of the other seven possible locations. The interruption itself lasted for fifteen seconds.

The procedure was the same as Experiment 1.

3.1.4. Measures

Data collection, error coding, and the calculation of error rates were the same as Experiment 1.

3.2. Results and discussion

3.2.1. Comparing error rates

Of the 36 participants, 32 participants made at least one perseveration or anticipation error. Error rates were compared between the control trials and actions immediately after the interruption using a repeated measures ANOVA. Participants made more errors following an interruption ($M = 6.7\%$) compared to the control ($M = .3\%$), $F(1, 35) = 45.8$, $\mathrm{MSE} = 16.2$, $p < .05$. Participants rarely made errors in the control trials, suggesting the task was well-learned.

3.2.2. Pattern of error actions

Next, we focused on the pattern of error actions. As in Experiment 1, the error actions were coded relative to the correct action at that point in the task hierarchy.

The distribution of error actions is illustrated in Fig. 4. The pattern of results on the financial task is remarkably similar to the results of the Sea Vessel task. As in Experiment 1, perseveration errors were the most common error action, $F(11, 341) = 35.4$, $\mathrm{MSE} = 286$, $p < .0001$. Tukey HSD post hoc comparisons revealed that participants were significantly more likely to make an immediate perseveration error ($M = 60.6\%$) than to make any other type of error (all p's $< .05$).

3.3. Model results

The model was not re-designed for the financial management task or re-run for the new dataset. Rather, the model results from Experiment 1 were simply copied over to the new dataset. As Fig. 4 suggests, the model fits at both a qualitative level ($R^2 = .98$) and a quantitative level ($\mathrm{RMSD} = 2.3$). The qualitative result strongly suggests that the sequence error findings are robust and replicable. The strong quantitative fit suggests that MFG model accurately describes the cognitive processes involved in sequence errors at the subtask level.

3.4. General discussion

The current paper presents two experiments and a novel model of sequential actions. Both experiments used an interruption paradigm, increasing the rate of errors enough to see emergent patterns from the data. The model used a memory for goals model that describes the process people go through both during error-free behavior and when they make errors. In general, errors occurred because the wrong episodic memory was retrieved. Perseveration errors occurred because a recent episodic memory had a high enough activation that, with noise, was retrieved instead of the correct memory. Anticipation errors occurred because the communication between the preparation and execution of an action gets disrupted for some reason.

The MFG model presented here connects with previous MFG models, showing an overall coherence with different tasks and measures. The original MFG model focused on the strengthening constraint, the priming constraint, and the interference level. The current MFG model uses these three constructs as well. The strengthening constraint is necessary because if an episodic trace does not start with a slightly higher activation, it can be very difficult to retrieve even a short period of time later. The priming constraint is used in the current model to help the retrieval of the episodic code given the current mental context. The interference level is critical to the model because it is the main determinant of perseveration errors, since the probability of making an error increases when the noise is high, which increases the chance that another (incorrect) episodic code will be retrieved.

The current MFG model connects less well with our previous model of recovery from task interruptions, which focused on the costs of recovery measured in terms of time rather than errors (Altmann & Trafton, 2007). In that study, we found that times between task-related actions returned asymptotically to baseline after an interruption, rather than abruptly. In our model there, we assumed that task resumption after interruption involved reconstructing an episodic representation of the current state of the interrupted task, and that this reconstruction process was affected by a kind of feedforward priming as each additional state element was retrieved. There is some possibility that the mechanisms underlying the temporal gradient in that data set and the sequence error gradients presented here (Fig. 2, especially the anticipation errors) are related. More generally, however, we would like to understand how the state-reconstruction process we proposed there might relate to the fine-grained control codes we propose here, as we work toward a more complete understanding of how the system recovers after performance of a relatively complex task is interrupted.

The MFG model shares both similarities and differences to the other two models of sequential routine action, IAN

and SRN. MFG focuses on perceptual and memorial processes rather than schemas (IAN) or distributed representations (SRN). However, it is interesting that all three models use noise as one of the primary explanatory constructs for why errors are made.

The current MFG model does have several limitations. First, it only accounts for sequence errors; it does not account for intrusions, capture errors, etc. Second, while both IAN and SRN attempt to model both normal and patient populations, the MFG model only addresses normally functioning individuals. Third, the model-task is quite simple, and a more complete task description is needed to expand the coverage of this model. Finally, the MFG model does not model the learning of the task itself.

The experiments reported here and the MFG model itself do, however, have several strengths. First, the experimental paradigm used here allows errors to be studied in the lab with normal populations. This data and other like it should be able to constrain current models of sequential actions, as Botvinick and Plaut (2006) suggest. Second, the MFG makes both qualitative and quantitative predictions about the error pattern for this task. Both the IAN and SRN models have been critiqued for the way they make perseveration errors. Finally, the model makes episodic memory an aspect of its normal processing, so errors arise out of normal processing of routine behavior.

Acknowledgments

This work was supported by ONR under funding document N0001410WX20426 and N0001410WX30037 to JGT and N000140310063 to EMA. The views and conclusions contained in this document should not be interpreted as necessarily representing the official policies of the US Navy.

References

Altmann, E. M., & Gray, W. D. (2008). An integrated model of cognitive control in task switching. *Psychological Review, 115*(3), 602–639.

Altmann, E. M., & John, B. E. (1999). Episodic indexing: A model of memory for attention events. *Cognitive Science, 23*(2), 117–156.

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive Science, 26*(1), 39–83.

Altmann, E. M., & Trafton, J. G. (2007). Time course of recovery from task interruption: Data and a model. *Psychonomics Bulletin and Review, 14*(6), 1079–1084.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* USA: Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglas, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review, 111*(4), 1036–1060.

Baars, B. J. (1992). *Experimental slips and human error: Exploring the architecture of volition*. New York: Plenum Press.

Botvinick, M. M., & Bylsma, L. M. (2005). Distraction and action slips in an everyday task: Evidence for a dynamic representation of task context. *Psychonomic Bulletin and Review, 12*(6), 1011–1017.

Botvinick, M. M., & Plaut, D. C. (2004). Doing without schema hierarchies: A recurrent connectionist approach to routine sequential action. *Psychological Review, 111*(2), 395–429.

Botvinick, M. M., & Plaut, D. C. (2006). Such stuff as habits are made on: A reply to Cooper and Shallice (2006). *Psychological Review, 113*, 917–928.

Byrne, M. D., & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive Science, 21*, 31–61.

Cooper, R. P. (2007). Tool use and related errors in ideational apraxia: The quantitative simulation of patient error profiles. *Cortex, 43*(3), 319–337.

Cooper, R. P., & Shallice, T. (2000). Contention scheduling and the control of routine activities. *Cognitive Neuropsychology, 17*(4), 297–338.

Cooper, R. P., & Shallice, T. (2006). Hierarchical schemas and goals in the control of sequential behavior. *Psychological Review, 113*(4), 887–916.

Cox, A. L., & Young, R. M. (2000). Device-oriented and task-oriented exploratory learning of interactive devices. In *Proceedings from the proceedings of the international conference of cognitive modeling*, 2000.

Giovannetti, T., Schwartz, M., & Buxbaum, L. (2007). The coffee challenge: A new method for the study of everyday action errors. *Journal of Clinical and Experimental Neuropsychology, 29*(7).

Gray, W. D. (2000). The nature and processing of errors in interactive behavior. *Cognitive Science, 24*(2), 205–248.

Hodgetts, H. M., & Jones, D. M. (2006). Interruption of the tower of london task: Support for a goal activation approach. *Journal of Experimental Psychology: General, 135*, 103–115.

Hommel, B., Müsseler, J., Aschersleben, G., & Prinz, W. (2001). The theory of event coding (TEC): A framework for perception and action planning. *Behavioral and Brain Sciences, 24*, 849–878.

Li, S. Y. W., Blandford, A., Cairns, P., & Young, R. M. (2008). The effect of interruptions on postcompletion and other procedural errors: An account based on the activation-based goal memory model. *Journal of Experimental Psychology: Applied, 14*(4), 314–328.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review, 88*, 1–15.

Norman, D. A., & Shallice, T. (1986). Attention to action: Willed and automatic control of behaviour. In *Proceedings from consciousness and self-regulation: Advances in research and theory*, New York.

Ratwani, R. M., & Trafton, J. G. (in press). A real time eye-tracking system for predicting and preventing post completion errors. *Human Computer Interaction*.

Ratwani, R. M., McCurry, J. M., & Trafton, J. G. (2008). Predicting postcompletion errors using eye movements. In *Proceedings of the conference on human factors in computing systems (SIGCHI 2008)* (pp. 539–542).

Reason, J. T. (1984). Lapses of attention in everyday life. In R. Parasuraman & D. R. Davies (Eds.), *Varieties of attention* (pp. 515–549). Orlando, FL: Academic Press.

Reason, J. T. (1990). *Human error*. Cambridge: Cambridge University Press.

Ruh, N., Cooper, R. P., & Mareschal, D. (2005). The time course of routine action. In B. G. Bara, L. Barsalou, & M. Bucciarelli (Eds.), *The Proceedings of the 27th annual meeting of the cognitive science society* (pp. 1889–1894).

Ruh, N., Cooper, R. P., & Mareschal, D. (2008). The hierarchies and systems that underlie routine behavior: Evidence from an experiment in virtual gardening. In V. Sloutsky, B. Love, & K. McRae (Eds.), *The proceedings of the 30th annual meeting of the cognitive science society* (pp. 339–344).

Sandson, J., & Albert, M. L. (1984). Varieties of perseveration. *Neuropsychologia, 22*, 715–732.

Schwartz, M. F., Montgomery, M. W., Buxbaum, L. J., & Lee, S. S. T. G. (1998). Naturalistic action impairment in closed head injury. *Neuropsychology, 12*(1), 13–28.

Trafton, J. G., Altmann, E. M., Brock, D. P., & Mintz, F. E. (2003). Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human Computer Studies, 58*(5), 583–603.