Chapter 2
Interactive Behavior and Human Error

In this chapter[1]:

- Usability is about how users use systems, i.e., user *behavior*. How is this characterized? What drives it?
- Major properties of user behavior regarded here are a) the time needed and b) the errors made. What distinguishes erroneous from 'normal' behavior?
- Which types of errors are important in HCI and how can these be explained theoretically?

The basic assumption of this work is that the behavior of a user of a system depends largely on the interface of the system. As John and Kieras have stated:

> Human activity with a computer system can be viewed as executing methods to accomplish goals, and because humans strive to be efficient, these methods are heavily determined by the design of the computer system. This means that the user's activity can be predicted to a great extent from the system design. (John and Kieras 1996a)

In other words: Given a sufficiently detailed definition of the user interface, one should be able to predict user behavior. What exactly follows from John and Kieras' proposition that "humans strive to be efficient" may be arguable, though. There is usually a tradeoff between effort and time. And the 'sweet spot' that gives the best result may differ between people and contexts.[2]

Starting from the premise given above, the factors that shape interactive behavior can be stated more formally. One such formalism is the ETA-triad (Gray 2000) as shown in Fig. 2.1. Understanding interactive behavior depends on understanding in

[1] Parts of Sect. 2.3 have already been published in Halbrügge et al. (2015b).

[2] Example: Keying ahead without visual feedback can save time, but needs more cognitive resources than pure reaction to visual cues on the interface.

M. Halbrügge, *Predicting User Performance and Errors*, T-Labs Series in Telecommunication Services, DOI 10.1007/978-3-319-60369-8_2

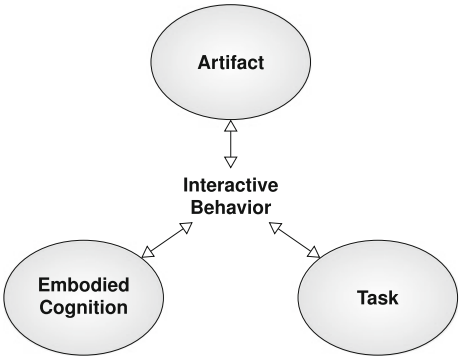

Fig. 2.1 The ETA-triad (Gray 2000)

three areas: the task, the artifact, and the constraints of the perception-cognition-motor system of the embodied[3] user.

In the context of this work, the schema of the ETA-triad is first applied to predict both successful and unsuccessful user behavior. These predictions shall then form the basis of automated usability predictions of the 'artifact' part of the triad.

The current chapter describes the basic assumptions and models used to describe the properties of the embodied user. It starts with a cognitive engineering model of human action control and error, followed by a phenotypical classification system of human error. This is finally contrasted with a theory-driven attempt to explain why errors happen while pursuing goals with a computer system.

2.1 Action Regulation and Human Error

A sufficiently detailed model of human decision-making for cognitive engineering domain has been proposed by Rasmussen (1986). His *step-ladder model* (see Fig. 2.2) consists of a perceptual leg on the left and an action leg on the right. The decision making process is started by activation through some new percept. This may trigger an attentional shift ("Observe") and conscious processing of the percept ("Identify"). Interpretation and evaluation leads to the definition of a new goal ("Define Task") and/or trigger an already existing action sequence ("Stored Procedure") which are finally executed.

Most activities in daily life do not require to go up to the very top of the ladder. Shortcuts between any two elements of the ladder may be acquired through learning. Examples for such shortcuts are given in Fig. 2.2.

[3] In this work, the term "embodiment" is used in a more elaborated sense than "cognition with added perception and motor capabilities". Instead, "embodied cognition" means that the analysis is not targeting the mind of the user, but the user-artifact dyad. In terms of Wilson's six views of embodied cognition, this is mainly related to the aspects "We off-load cognitive work onto the environment" and "The environment is part of the cognitive system" (Wilson 2002).

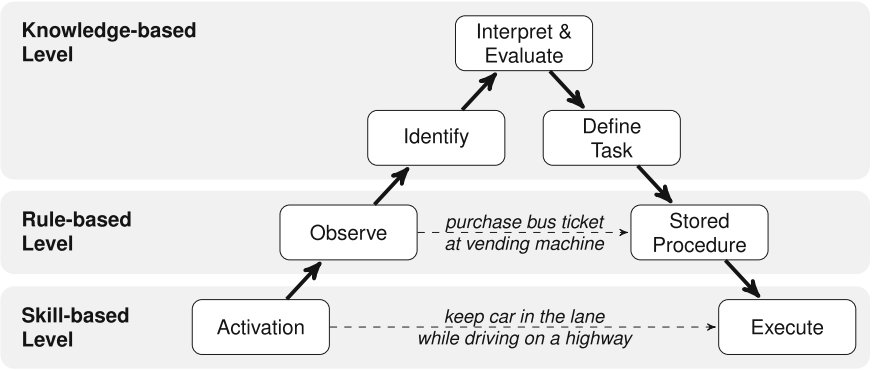

Fig. 2.2 Step-ladder model of decision making. Adapted and simplified from Rasmussen (1986, p. 67), Hollnagel (1998, p. 61), Reason (1990, p. 64), and Rasmussen (1983). *Solid arrows* display the expected sequence of processing stages during decision-making or problem solving. *Dashed arrows* represent examples for shortcuts that have been established mainly through training. Such shortcuts may exist between any two of the boxes

Another view on the step-ladder model is the level of action control that is applied. These levels are represented as gray boxes in the background of the figure. According to Rasmussen (1983), human action control can be described on three levels: skill-, rule-, and knowledge-based behavior. Skill-based behavior on Rasmussen's lowest level is generated from highly automated sensory-motor actions without conscious control. Knowledge-based behavior on the other hand is characterized by explicit planning and problem solving in unknown environments. In between the skill and the knowledge levels is rule-based behavior. While being goal-directed, rule-based actions do not need explicit planning or conscious processing of the current goal. The stream of actions follows stored procedures and rules that have been developed during earlier encounters or through instruction. Interaction with computer systems is mainly located on this intermediate rule-based level of action control.

2.1.1 Human Error in General

Human error is commonly defined as

> those occasions in which a planned sequence of mental or physical activities fail to achieve its intended outcome, [and] when these failures cannot be attributed to the intervention of some chance agency. (Reason 1990, p. 9)

This definition is very broad as it is meant to cover any kind of erroneous action. It is nevertheless instrumental in highlighting a property of human error that makes its research so complicated: Whether something is an error or not depends on its *intended outcome*. This has two consequences. First, it is impossible to determine whether something is an error or not without knowing the (unobservable) intention

behind it. And second, errors usually cannot be observed as they happen, but only when their outcome becomes manifest. In terms of the step-ladder model, this means that only the "Execute" stage on the lower right creates overt behavior. In case of an error, the cause of the error may be located on any other stage or connection between two stages.

2.1.2 Procedural Error, Intrusions and Omissions

Interaction with computer systems is mainly located on the intermediate rule-based level of action control. On this level, behavior is generated using stored rules and procedures that have been formed during training or earlier encounters. Errors on the rule-based level are not very frequent (below 5%), but pervasive and cannot be eliminated through training (Reason 1990). While Norman (1988) subsumes these errors within the 'slips' category, Reason (1990, 2016) refers to them as either 'lapses' in case of forgetting an intended action or 'rule-based mistakes' when the wrong rule (i.e., stored procedure) is applied. Because of this ambiguity, the term *procedural error* is used throughout this work.

Procedural error is defined as the violation of the (optimal) path to the current goal by a non-optimal action (cf., Singleton 1973). This can either be the addition of an unnecessary or even hindering action, which is called an *intrusion*. Or a necessary step can be left out, constituting an *omission*.

How does procedural error manifest itself in daily life?

Example: Postcompletion Error

A very common and also well researched example of procedural error is *postcompletion error* (Byrne and Bovair 1997; Byrne and Davis 2006). This type of error is characterized by the omission of the last step of an action sequence if the overall goal of the user has already been accomplished before. Typical examples of postcompletion errors are forgetting the originals in a copy machine[4] and leaving the bank card in a teller machine after having taken the money.

Similar errors can happen during the first step of an action sequence as well. This type has been coined *initialization error* (Li et al. 2008; Hiltz et al. 2010). An example of this kind of error is forgetting to press a 'mode' key before setting the alarm clock on a digital watch (Ament et al. 2010).

Because of its prototypical nature, postcompletion error has become one of the basic tests for error research and action control theories. The ability of such theories to explain postcompletion error is often used as argument in favor of their validity (e.g., Byrne and Davis 2006; Altmann and Trafton 2002; Butterworth et al. 2000, see Sect. 2.3 below). Before reviewing theories of procedural error, its notion shall first be contrasted with other descriptions of typical errors that occur at home and in the workplace.

[4]Side remark: According to a copy shop clerk, this error has been superseeded in frequency by clients forgetting their data stick after having received their printout.

2.2 Error Classification and Human Reliability

Early error research has focused on classification systems of human error. These contain usually more categories than the differentiation between omissions and intrusions given above. The best known of these classification systems has been created by Norman and will be presented in the following.

2.2.1 Slips and Mistakes—The Work of Donald A. Norman

Norman (1981, 1988) distinguished between slips and mistakes. The basic difference between those is that mistakes happen when an incorrect intention is acted out correctly. Slips on the other hand mark situations when a correct intention is acted out incorrectly. Referring to the step-ladder model above (Fig. 2.2), mistakes belong to knowledge-based behavior in the upper part of the ladder and slips belong to either rule-based or skill-based behavior. Norman (1988) does not provide further sub-categories for mistakes, but distinguishes between several types of action slips. These are given with examples in Table 2.1.

Norman's classification scheme has drawn criticism for several reasons. First, according to Hollnagel (1998), it mingles genotypes (e.g., 'associative activation error') and phenotypes (e.g., 'mode error') which leads to inconsistencies. And second, it is disputable whether mode errors are actual action slips in Norman's terms as they are not characterized by faulty action. A 'mistaken system state' should rather be considered an incorrect intention, which puts mode errors into the 'mistake' category.

In the context of this work, of highest importance is whether such a classification suits automatable usability predictions. How does Norman's system relate to these?

2.2.2 Human Reliability Analysis

Classification schemes like Norman's have been combined to models of *human reliability* that can be used to predict overall error rates for safety-related tasks like controlling a nuclear power plant (Kirwan 1997a, b). Unfortunately, the validity of these approaches does not live up to the expectations (Wickens et al. 2015, Chap. 9). This is most probably due to the fact that classification schemes only *describe* human error, but do not *explain* how correct and erroneous behavior is produced. The remainder of this chapter presents models of human behavior and action control that try to provide such explanations.

Table 2.1 Types of action slips and examples as reported in Norman (1988, p. 107f). The examples have been slightly shortened by the author

Type of slip	Example
Capture error (habit take-over)	"I was using a copying machine, and I was counting the pages. I found myself counting '1, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King.' I have been playing cards recently."
Description error (incomplete specification)	"A former student reported that one day he came home from jogging, took off his sweaty shirt, and rolled it up in a ball, intending to throw it in the laundry basket. Instead he threw it in the toilet. (It wasn't poor aim: the laundry basket and the toilet were in different rooms.)"
Data driven error (dominant stimulus driven)	"I was assigning a visitor a room to use. I decided to call the department secretary to tell her the room number. I used the telephone with the room number in sight. Instead of dialing the secretary's phone number I dialed the room number."
Associative activation error (Freudian slip)	"My office phone rang. I picked up the receiver and bellowed 'Come in' at it."
Loss-of-action error (forget intention)	"I have to go to the bedroom. As I am walking, I have no idea why I'm going there. I keep going hoping that something in the bedroom will remind me. Finally I realize that my glasses are dirty. I get my handkerchief from the bedroom, and wipe my glasses clean."
Mode error (mistaken system state)	"I had just completed a long run in what I was convinced would be record time. It was dark, so I could not read the time on my stopwatch. I remembered that my watch had a built-in light. I depressed the button, only to read a time of zero seconds. I had forgotten that in stopwatch mode, the same button cleared the time and reset the stopwatch."

2.3 Theoretical Explanations of Human Error

2.3.1 Contention Scheduling and the Supervisory System

Norman and Shallice (1986) proposed a model of action selection called 'contention scheduling' which depends on activation through either sensory (horizontal) 'triggers' or internal (vertical) 'source' schemas which represent volitional control by a so-called 'supervisory attentional system'. The 'contention' in this model arises from reciprocal inhibition of the schemas[5] that belong to individual actions. These rather simple assumptions can already explain some types of errors, e.g., capture errors as activation from a sensory trigger that overrides the action sequence that had been followed before. From the description of the model, it should already be clear that it does not cover errors on the knowledge-based or skill-based levels, but aims at routine activities like making coffee.

[5]Note: These are 'action' schemas, not to be confused with the 'source' schemas.

The contention scheduling model has been implemented by Cooper and Shallice (2000) with a subsequent validation using data from patient groups with impaired action control. Interestingly, they do not stick to the error categories that had be put forward by Norman (1988), but use a coding system based on the *disorganization* of actions—as opposed to errors—within a sequence instead (Schwartz et al. 1998). They write about Norman's classification system:

> These categories are neither disjoint nor definitive, and there can be difficulties in using them to classify certain action sequences (Cooper and Shallice 2000, p. 300)

Later research aimed at confirming the existence of a supervisory system based on action latencies while learning new routines (Ruh et al. 2010).

2.3.2 Modeling Human Error with ACT-R

A more rigorous attempt to explain human error based on psychological theory of action control has been presented by Gray (2000). He observed users while programming a videocassette recorder (VCR) to record television shows and modeled their behavior using the now outdated version 2 of the cognitive architecture ACT-R (Anderson and Lebiere 1998, see Sect. 4.2). According to Gray, using a cognitive model leads not only to better understanding of human error, it also creates better vocabulary for the description of errors than simple category systems like the one of Norman (1981, see Table 2.1).

Gray assumes that goals and subgoals that control behavior are represented in a hierarchical tree-like structure. The goal stack of ACT-R 2 is used to traverse this tree in a depth-first manner to produce actual behavior. In order to complete a goal, its first subgoal is pushed to the stack. After completion of the subgoal, it is popped from the stack and the next subgoal is pushed. Based on this process, errors can be divided into push errors (attaining a subgoal at an unpredicted point in time) and pop errors (releasing a subgoal too early or too late).

Push errors observed in Gray's VCR paradigm were for example setting 'rec-mode' before start and end time had been set (rule hierarchy failure) or trying to access something that is currently visible, but unchangeable (display-induced error). Push errors tend to decrease with practice (through learning of the goal hierarchy).

Pop errors can be further decomposed into premature pops (goal suspensions) and postponed pops. Premature pops manifest themselves by a subgoal being interrupted before it is completed (intrusion). Interrupting goals are often close to the interrupted ones. Interestingly, premature pops *increase* with routine. Gray attributes this to competition with leftovers from previous trials. Postponed pops on the other hand were mainly physical slips, e.g., too many repetitions while setting the clock to the start or end time of the show to be recorded.

Gray's cognitive model proved correct in the sense that it a) works, i.e., can solve the task, b) matches human behavior on correct trials, and c) makes similar errors that humans make. At the same time, the vision based strategy applied in the model

serves as error detection and error recovery strategy as well. This is also in line with the error recovery behavior observed in the VCR paradigm. Of 28 detected and corrected errors, only four were not visible to the user. Gray concludes that error detection is local. Errors are detected and corrected either right after they have been made, or not at all.

Postcompletion errors (see Sect. 2.1.2) could be classified as premature pops in Gray's nomenclature, but Gray's model has problems explaining these. As ACT-R 2's goal stack has perfect memory, the model does not exhibit premature pops if there is no other goal that can take over control. At the end of an action sequence, no such intruder is available.

The approach taken by Gray provides important insights about how errors can be explained and showcases the usefulness of cognitive modeling as a research method in this field. The assumption of a goal hierarchy that is processed recursively using a stack has been questioned, though. An alternative, more parsimonious theory of action control is the Memory for Goals model (Altmann and Trafton 2002).

2.3.3 Memory for Goals Model of Sequential Action

The Memory for Goals model (MFG; Altmann and Trafton 2002) postulates that goals and subgoals are not managed using a dedicated goal stack, but reside in generic declarative memory. This implies that goals not 'special', but are memory traces among many others. As such, they are subject to the general characteristics of human associative memory (Anderson and Bower 2014), in particular time-dependent and noisy *activation*, *interference*, and *priming*. With respect to action control and human error, lack of activation of a subgoal can cause omissions, while interference with other subgoals can result in intrusions.

Based on these assumptions, postcompletion error (see Sect. 2.1.2) is mainly explained by lack of activation through priming. In the MFG, a sequence of actions arises from consecutive retrievals of subgoals from declarative memory. These retrievals are facilitated by priming from internal and external cues. As the subgoals that correspond to typical postcompletion errors (e.g., taking the originals from a copy machine) are only weakly connected to the overall goal of the action sequence (e.g., making copies), they receive less priming and are therefore harder to retrieve.

While the MFG theory initially has been validated on the basis of Tower-of-Hanoi experiments, i.e., rather artificial problem-solving tasks in the laboratory, it has been shown to generalize well to procedural error during software use and has been extensively used in the human-computer interaction domain (e.g., Li et al. 2008; Trafton et al. 2011).

2.4 Conclusion

The current chapter introduced the basic concepts related to human behavior with regards to cognitive engineering of interactive software systems. User behavior can be decomposed based on how control is applied. This may happen on the skill-based, rule-based, or knowledge-based level (Rasmussen 1983). Errors can happen on any of these levels, but it can be hard to distinguish them based on the overt behavior alone. Understanding how users manage their goals and how they come into action is key to understanding user error.

In the following chapter, the focus shifts from the Embodied cognition part of the ETA-triad to the Artifact part.

Springer

http://www.springer.com/978-3-319-60368-1

Predicting User Performance and Errors
Automated Usability Evaluation Through Computational Introspection of Model-Based User Interfaces

Halbrügge, M.

2018, XVI, 149 p. 44 illus., 17 illus. in color., Hardcover
ISBN: 978-3-319-60368-1