On Preventing Telephony Feature Interactions which are Shared-Control Mode Confusions

Jan Bredereke
Universität Bremen, FB 3 · P.O. box 330 440 · D-28334 Bremen · Germany
brederek@tzi.de · www.tzi.de/˜brederek

Abstract.
We demonstrate that many undesired telephony feature interactions are also shared-control mode confusions. A mode confusion occurs when the observed behaviour of a technical system is out of sync with the behaviour of the user's mental model of it. Several measures for preventing mode confusions are known in the literature on human-computer interaction. We show that these measures can be applied to this kind of feature interaction. We sketch several more measures for the telephony domain.

1 Introduction

Automation surprises are ubiquitous in today's highly engineered world. We are confronted with *mode confusions* in many everyday situations: When our cordless phone rings while it is located in its cradle, we establish the line by just lifting the handset — and inadvertently cut it when we press the hook toggle button as usual with the intention to start speaking. We get annoyed if we once again overwrite some text in the word processor because we had hit the "Ins"-key before (and thereby left the insert mode!) without noticing. The American Federal Aviation Administration (FAA) considers mode confusion to be a significant safety concern in modern aircraft. For instance, consider the crash of an Airbus A320 near Strasbourg, France, in 1992 [1]. Probably due to heavy workload because of a last-minute path correction demanded by the air traffic controller, the pilots confused the "vertical speed" and the "flight path angle" modes of descent. As a result, the Air Inter machine descended far too steeply, crashed, and 87 people were killed.

Many safety-critical systems today are so-called embedded shared-control systems. These are interdependently controlled by an automation component and a user. Examples are modern aircraft and automobiles.

In telephony, call control is shared between users and many modern telephony features. Some examples are call screening, call forwarding, voice mail, and credit card calling. Multiparty features such as three-way calling let all users involved share call control to some extent. In contrast to many shared-control systems, telephones usually are not immediately safety-critical. Nonetheless, users expect a comparably high reliability which must not be impaired by undesired feature interaction.

Feature interaction occurs when one feature modifies or subverts the operation of another one. The above mode confusion example of the auto off-hook feature and the hook toggle button feature also demonstrates an undesired interaction between these two features. The

feature interaction benchmark of Cameron *et.al.* [2] presents examples for many different kinds of feature interactions (see Sect. 3).

We found that many undesired telephony feature interactions are also shared-control mode confusions. Several measures for preventing mode confusions are known in the literature on human-computer interaction. We now show that these measures can be applied to this kind of feature interaction.

This paper is organized as follows: Section 2 introduces to mode confusions. In Section 3, we investigate the well-known feature interaction benchmark [2] and discuss the mode confusions we found there. Section 4 presents a definition of mode confusion adapted to telephony, and in Section 5 we show how the notion of mode confusion can help against undesired feature interactions. Section 6 summarizes our paper and discusses future work.

2 Mode Confusions

2.1 What is a Mode Confusion?

Humans use *mental models* when they interact with technical systems in general, and with automated systems in particular. This is generally agreed upon in cognitive science [3]. Unfortunately there is an additional, completely different meaning for the notion "mental model" in the pertinent literature. We refer to the one introduced by Norman [4]: a mental model represents the user's knowledge about a technical system, it consists of a naïve theory of the system's behaviour.

An explicit description of a mental model can be derived, according to Rushby [3]. It can be represented, e. g., in form of a *state machine* with modes and mode transitions. We can extract it from training material, from user interviews, or by user observation. Cañas *et al.* [5] survey work on this and show in three experiments with 140 participants how exposing users to different knowledge elicitation tasks allows to figure out their mental models. Rushby qualifies his statement by conceding that it is difficult and expensive to extract the mental models of the individual users. Fortunately, this is neither necessary for his nor for our work. We are interested in representative examples of mental models. Our goal is to design the system such that as many users as possible will construct an adequate mental model.

A mental model can sometimes change over time, but this does not conflict with our goal of mode confusion prevention. The user's knowledge changes when he/she learns or forgets. When the user's mental model changes after we have extracted and documented it, we cannot predict the behaviour of the user correctly anymore. But we are not interested in any particular user's behaviour. Again, we are interested in improving the design of the technical system in order to ease its use in general.

The mental model which is the user's long-term knowledge is different from the user's current working abstraction of this mental model. When performing a task, the user concentrates on the part of his/her knowledge which he/she assumes to be relevant [5].

Interestingly, no publication defines the notions of "mode" and "mode confusion" rigorously [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] in a recent survey of ours [17]. We therefore propose such a rigorous definition of "mode" and "mode confusion" for safety-critical systems in that work. We sketch it in the following.

Some researchers in the Human Factors (HF) community will disagree with our rigorous definitions [17] at certain points, but this does not matter for the investigation of mode confusions in telephony feature interactions. Different researchers in the HF community have a

slightly different intuitive understanding of mode and mode confusion (and we are part of that). We proposed our rigorous definitions as a base for discussion. They enable to pinpoint the differences and to resolve them by discussion in the HF community.

We must use a *black-box view* of a running technical system for the definition. This is because the user of such a system has a strict black-box view of it and because we want to solve the user's problems. As a consequence, we can observe (only) the environment of the technical system. When something relevant happens in the environment, we call this an *event*.

The technical system has been constructed according to some *requirements* document REQ. We can describe REQ entirely in terms of observable events, by referring to the *history* of events until the current point of time. For this description, no reference to an internal state is necessary. Usually, several histories of events are equivalent with respect to what should happen in the future. Such equivalences can greatly simplify the description of the behaviour required, since we might need to state only a few things about the history in order to characterise the situation.

During any run of the technical system, it is in one specific state at any point of time. The (possibly infinite) state transition system specified by REQ defines the admissible system runs.

We call the user's mental model of the technical system $\mathrm{REQ}^{\mathrm{M}}$. Ideally, $\mathrm{REQ}^{\mathrm{M}}$ should be the same as REQ. During any run of the technical system, $\mathrm{REQ}^{\mathrm{M}}$ is also in one specific state at any point of time. You may think of the behaviour of $\mathrm{REQ}^{\mathrm{M}}$ as a "parallel universe" in the user's mind. Ideally, it is tightly coupled to reality.

Our approach is based on the motto "*the user must not be surprised*". This is an important design goal for shared-control systems. We must make sure that the reality does not exhibit any behaviour which cannot occur according to the mental model of it. Additionally, the user must not be surprised because something expected does *not* happen. When the mental model prescribes some behaviour as necessary, reality must not refuse to perform it. For example after dialling a number, a phone must either produce an alert tone or a busy tone, and it must never ring itself.

The rule of non-surprise means that the relationship between the reality and the user's mental model of it must be a *relationship of implementation to specification*. The reality should do exactly what the mental model prescribes, no less and no more. In case that the user does not know what to expect, but knows that he/she does not know, then the reality is free to take any of the choices. A common example is that the user does not know the exact point of time at which the technical system will react to an event, within some limits.

We can describe such an implementation/specification relationship formally by a refinement relation. Many formalisms have been proposed for this. We use CSP [18, 19]. It is one suitable formalism with suitable tool support. In CSP, *failure refinement* is precisely the relation described above.

In CSP, one describes the externally visible behaviour of a system by a so-called process. Processes are defined over events. CSP offers a set of operators. One can use them to specify processes. CSP also provides a number of refinement operators. There is a precise formal semantics for all of this [18].

The user does not always notice when the behaviour of his/her mental model $\mathrm{REQ}^{\mathrm{M}}$ is not the same as that of the reality REQ. This is because the user's mind does not take part in *any* event in the environment. The user perceives the reality through his/her senses only.

The user's senses SENSE translate from the set of events in the environment to a set of events in the user's mind. SENSE is not perfect. Therefore we must distinguish these two

sets. For example, the user might not hear a signal tone in the phone due to loud surrounding noise. Or the user might not listen to all of a lengthy announcement text, or he/she might not understand the language of the announcement. At the very least, there is always a larger-than-zero delay between any environment event and the respective mental event. In all these cases, what happens in reality, as described by REQ, is different from what happens according to the user's perception of it, as described by $\mathrm{SENSE}(\mathrm{REQ})$.[1]

The user is surprised only if the *perceived* reality does not behave the same as his/her expectations. This is why the user does not always notice a difference between the actual reality REQ and the "parallel universe" $\mathrm{REQ}^{\mathrm{M}}$ in his/her mind.

We cannot compare the perceived reality $\mathrm{SENSE}(\mathrm{REQ})$ to the mental model of the reality $\mathrm{REQ}^{\mathrm{M}}$ directly. They are defined over different sets of events (mental/environment). We need a translation.

The user has a mental model of his/her own senses $\mathrm{SENSE}^{\mathrm{M}}$. $\mathrm{SENSE}^{\mathrm{M}}$ translates the behaviour of the mental model of the technical system $\mathrm{REQ}^{\mathrm{M}}$ into events in the user's mind. It does this in the same fashion as SENSE does it for REQ.

The user's knowledge about the restrictions and imprecisions of his/her own senses is also part of $\mathrm{SENSE}^{\mathrm{M}}$. Ideally, the user should know about them precisely, such that $\mathrm{SENSE}^{\mathrm{M}}$ and SENSE are the same. The user is not surprised if the process $\mathrm{SENSE}(\mathrm{REQ})$ is a failure refinement of the process $\mathrm{SENSE}^{\mathrm{M}}(\mathrm{REQ}^{\mathrm{M}})$.[2]

Our definition of mode confusion [17] concentrates on the safety-relevant aspects of the technical system. This is because traditionally the safety-critical systems community has perceived mode confusions as a problem.

When the user concentrates on safety, he/she performs an on-the-fly simplification of his/her mental model $\mathrm{REQ}^{\mathrm{M}}$ towards the safety-relevant part $\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}}$. This helps him/her to analyse the current problem with the limited mental capacity. Psychological studies show that users always adapt their current mental model of the technical system according to the specific task they carry out [5]. The "initialisation" of this adaptation process is the static part of their mental model, the so-called "conceptual model" [20]. This model represents the user's knowledge about the system and is stored in the long term memory.

Analogously to the abstraction performed by the user, we perform a simplification of the requirements document REQ to the safety-relevant part of it $\mathrm{REQ}_{\mathrm{SAFE}}$. $\mathrm{REQ}_{\mathrm{SAFE}}$ can be either an explicit, separate chapter of REQ, or we can express it implicitly by specifying an abstraction function, i. e., by describing which aspects of REQ are safety-relevant. We abstract REQ out of three reasons: $\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}}$ is defined over a set of abstracted events, and it can be compared to another description only if it is defined over the same abstracted set; we would like to establish the correctness of the safety-relevant part without having to investigate the correctness of the entire mental model $\mathrm{REQ}^{\mathrm{M}}$; and our model-checking tool support demands that the descriptions are restricted to certain complexity limits.

We express the abstraction functions mathematically in CSP by functions over processes. Mostly, such an abstraction function maps an entire set of events onto a single abstracted event. Other transformations are hiding (or concealment [19]) and renaming. But the formalism also allows for arbitrary transformations of behaviours; a simple example being a certain

[1] One must also distinguish between what happens in reality and what happens according to the software's perception of it, due to imperfect input/output devices. But these software events are not relevant here since we must view the technical system as a black-box.

[2] In [17], we used the name MMOD for $\mathrm{SENSE}^{\mathrm{M}}(\mathrm{REQ}^{\mathrm{M}})$. We did not define $\mathrm{SENSE}^{\mathrm{M}}$ and $\mathrm{REQ}^{\mathrm{M}}$ separately. We now make a distinction between these two different kinds of mental model for clarity.

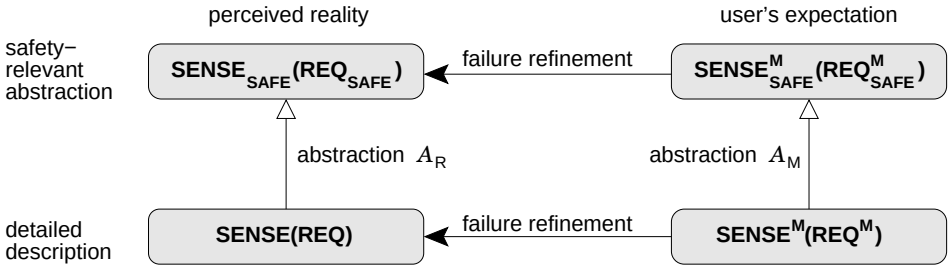

Figure 1: Relationships between the different refinement relations.

event sequence pattern mapped onto a new abstract event. We use the abstraction functions $\mathcal{A}_R$ for REQ and $\mathcal{A}_M$ for $\mathrm{REQ}^{\mathrm{M}}$, respectively.

The relation SENSE must be abstracted in an analogous way to $\mathrm{SENSE}_{\mathrm{SAFE}}$. They are relations from processes over environment events to processes over mental events. It should have become clear by now that $\mathrm{SENSE}_{\mathrm{SAFE}}$ needs to be rather true, i. e., a bijection which does no more than some renaming of events. If $\mathrm{SENSE}_{\mathrm{SAFE}}$ is "lossy", we are already bound to experience mode confusion problems. $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{SAFE}}$ accordingly is the user's mental model of $\mathrm{SENSE}_{\mathrm{SAFE}}$.

Figure 1 shows the relationships among the different descriptions. In order not to surprise the user with respect to safety, there must be a failure refinement relation on the abstract level between $\mathrm{SENSE}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ and $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{SAFE}}(\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}})$.

We defined the notion of mode confusion rigorously [17]. The central definition is:

Definition 1 (Mode confusion). A *mode confusion* between $\mathrm{SENSE}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ and $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{SAFE}}(\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}})$ occurs if and only if $\mathrm{SENSE}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ is not a failure refinement of $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{SAFE}}(\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}})$, i.e., iff $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{SAFE}}(\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}}) \quad \not\sqsubseteq_F \quad \mathrm{SENSE}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$.

We also defined the notion of mode rigorously [17]. A mode is a just state, but we reserve the word for the "states" of abstracted descriptions, i. e., of $\mathrm{SENSE}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ and of $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{SAFE}}(\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}})$. A state (or mode) is a potential future behaviour. We can distinguish two states (or two modes) of a system only if the system may behave differently in the future. This is because of the black-box view. For two different modes, there must be a safety-relevant difference. We omit the further details here.

Every time $\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}}$ changes, one must decide anew whether a mode confusion occurs. Our definition of mode confusion is based on the (rather strong) assumption that $\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}}$ is stable over time. The user generates $\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}}$ on-the-fly from $\mathrm{REQ}^{\mathrm{M}}$ and must re-generate it later when he/she needs it again. This re-generation might lead to a different result. In particular, the re-generation requires the user's recollection of the current mode. A user's lapse [21] here can result in a mode confusion. This happens when the user selects a mode as initial mode which does not match the reality's current mode.

2.2 What Can We Do Against Mode Confusion Problems?

Rushby proposes a procedure to develop automated systems which pays attention to the mode confusion problem [3]. The main part of his method is the integration and iteration of a

model-checking based consistency check and the mental model reduction process introduced by [22, 6].

Vakil and Hansman, Jr. [23] recommend three approaches to reduce mode confusion potential in modern aircraft: pilot training, enhanced feedback via an improved interface, and, most substantial, a new design process (ODP, for operator directed design process) for future aircraft developments. ODP aims at reducing the complexity of the pilot's task, which may involve a reduction of functionality.

Our definitions allow us to classify mode confusion problems, to derive recommendations from the classification for avoiding some of the problems, and we can also use the definitions as a foundation for detecting mode confusions [17].

We *classify* mode confusion problems into four classes:

1. Mode confusion problems which arise from an *incorrect observation* of the technical system or its environment. This may have *physical* or *psychological* reasons.

2. Mode confusion problems which arise from *incorrect knowledge* of the human. The incorrect knowledge may be either about the *technical system* or its environment, or about the human's own *senses*.

3. Mode confusion problems which arise from the *incorrect abstraction* of the user's knowledge to the safety-relevant aspects of it.

4. Mode confusion problems which arise from an *incorrect processing* of the abstracted mental model by the user. This can be a memory lapse or a "rule-based" mistake [21], i. e., a mode transition that is not part of the model.[3]

The above causes of mode confusion problems lead directly to some *recommendations for avoiding* them. In order to avoid an incorrect observation of the technical system and its environment, we must check that the user can physically observe all safety-relevant environment events, and we must check that the user's senses are sufficiently precise to ensure an accurate translation of these environment events to mental events. If this is not the case, then we must change the system requirements. We must add an environment event controlled by the machine and observed by the user which indicates the corresponding software input event. In order to avoid an incorrect observation because of a psychological reason, we must check that observed safety-relevant environment events become conscious reliably.

Establishing a correct knowledge of the user about the technical system and its environment can be achieved by documenting the requirements of them rigorously. This enables us to produce user training material, such as a manual, which is complete with respect to functionality. This training material must not only be complete but also learnable. Complexity is an important learning obstacle. Therefore, the requirements of the technical system should allow as little non-deterministic internal choices as possible, since tracking all alternative outcomes is complex. This generalises and justifies the recommendation by others to eliminate "implicit mode changes" [13, 11]. We can eliminate a non-deterministic internal choice by the same measure as used against an incorrect physical observation: we add an environment event controlled by the machine which indicates the software's choice.

[3] In [17], we viewed this cause as part of the "incorrect knowledge" cause. We now prefer to distinguish the different kinds of human error identified by Reason [21].

Improving the user's knowledge about his/her own senses has little potential for avoiding mode confusion problems. If the user knows that some things may happen, but he/she cannot perceive them, then they are non-deterministic choices to the user's mind. Again, the user will have difficulties with the complexity of tracking alternative outcomes. We can improve the technical system's design if we not only avoid incorrect observations but also such incomplete observations. The concrete measures are similar to those against incorrect observations.

Ensuring a correct mental abstraction process is mainly a psychological question and mostly beyond our scope. Our work leads to the basic recommendation to either write an explicit, rigorous safety-relevance requirements document or to indicate the safety-relevant aspects clearly in the general requirements document. The latter is equivalent to making explicit the safety-relevance abstraction function for the machine $\mathcal{A}_R$. Either measure facilitates to produce training material which helps the user to concentrate on safety-relevant aspects.

Incorrect processing of the abstracted model by the user is a *human error*. Reason [21] distinguishes three basic types of human error: skill-based slips and lapses, rule-based mistakes, and knowledge-based mistakes. Slips appear as incorrect observation for psychological reasons in our classification, and knowledge-based mistakes appear as incorrect knowledge. Lapses and rule-based mistakes cause incorrect processing. The design of a system that avoids them is again a psychological question and mostly beyond our scope. Reason [21] surveys advices for reducing the human error risk.

Our definitions form a *foundation for detecting* mode confusions by model-checking. We model-checked successfully an autonomous wheelchair for mode confusion problems [24].

3 Mode Confusion Problems in the Feature Interaction Benchmark

We found that a considerable number of undesired telephony feature interactions in the feature interaction benchmark of Cameron *et.al.* [2] are also shared-control mode confusions. The benchmark was written by practitioners from the telecom industry. It provides representative examples of a broad range of undesired feature interactions.

The remainder of this section contains a description and a discussion of those feature interaction examples that are also mode confusion problems. All descriptions of feature interactions are quoted from Cameron *et.al.* [2] (except one variant in Section 3.3 and one extra example of ours in Section 3.11). For completeness, we also briefly sketch the wide range of causes from the other feature interaction examples.

We use an informal notion of mode confusion in our discussion here that does not abstract the system to its safety-relevant aspects. This would not make sense for telephony. We discuss this aspect in Section 4 below.

Table 1 on the next page summarizes the mode confusions we found in the feature interaction benchmark. There are 12 mode confusion problems in 8 of the 22 examples of the feature interaction benchmark. This shows that mode confusions are definitely a relevant cause for feature interaction problems. (Obviously, there are other causes, too. We discuss them briefly in the end of this section.)

3.1 Example 2 – Call Waiting and Three-Way Calling (CW & TWC)

Description [2]. "The signalling capability of customer premises equipment (CPE) is limited. As a result, the same signal can mean different things depending on which feature is

Table 1: a summary of the mode confusions found in the feature interaction benchmark of Cameron *et.al.*.

benchmark example no.	benchmark example ID	number of mode confusions	benchmark example no.	benchmark example ID	number of mode confusions
1	CW&AC	–	12	OCS&CF/2	–
2	CW&TWC	2	13	CW&ACB	–
3	911&TWC	1	14	CW&CW	2
4	TCS&ARC	–	15	CW&TWC/2	1
5	OCS&ANC	–	16	CND&UN	–
6	Operator&OCS	–	17	CF&CF	–
7	CCC&VM	2	18	ACB&ARC	–
8	MBS-ED&CENTREX	–	19	LDC&MRC	1
9	CF&OCS	–	20	Hotel	2
10	CW&PCS	1	21	Billing	–
11	OCS&MDNL-DR	–	22	AIN&POTS	–

anticipated. For example, a flash-hook signal (generated by hanging up briefly or depressing a 'tap' button) issued by a busy party could mean to start adding a third party to an established call (Three-Way Calling) or to accept a connection attempt from a new caller while putting the current conversation on hold (Call Waiting). Suppose that during a phone conversation between **A** and **B**, an incoming call from **C** has arrived at the switching element for **A**'s line and triggered the Call Waiting feature that **A** subscribes to. However, before being alerted by the call-waiting tone, **A** has flashed the hook, intending to initiate a three-way call. Should the flash-hook be considered the response for Call-Waiting, or an initiation signal for Three-Way Calling?"

Discussion. This feature interaction can be resolved by a precedence rule. An activated Call-Waiting feature should have a higher precedence than the Three-Way Calling feature, with respect to the interpretation of the flash-hook. This allows both features to work most of the time without introducing a new user signal.

The mode confusion problem in the above particular situation remains, though. The mode of the switching element has changed, but the user issuing the flash-hook signal has not yet noticed it. He/she therefore will be surprised by an unexpected reaction of the system.

The cause of the mode confusion is a race condition between the notification tone for the mode change and the user signal. (A mode change not apparent to the user is called an *implicit mode change* in the literature on human-computer interaction [13, 11].) When **A** plans how to perform the three-way call, **A** will use his/her knowledge about the behaviour of the system, but he/she will concentrate on the "relevant" part for efficiency. This will probably make **A** exclude the Call Waiting feature, even when **A** usually is aware of it. The result is the mode confusion in which **A** expects a dial tone but is connected to the new party instead.

A careful user **A** can avoid the mode confusion, but still remains in an uncomfortable situation. **A** must expect a non-deterministic reaction of the system to the flash-hook signal. **A** can find out the actual current mode only by waiting a short amount of time whether a dial tone becomes audible. Only after this re-synchronization, **A** can either proceed with his/her

plan, or adjust it to the new situation of an incoming call.

3.2 Example 2b – Plain Old Telephone Service and Plain Old Telephone Service (POTS & POTS)

There is another mode confusion problem in the same example of the benchmark. Cameron *et.al.* do not consider it as a feature interaction for the technical reason that no *incremental* features to the Plain Old Telephone Service (POTS) are involved.

Description, continued [2]. "A similar situation occurs when lifting a handset is interpreted as accepting the incoming call, even though the user's intention is to initiate a call – remember the cases when one picks up the phone in the absence of ringing and somebody is already at the other end of the line. The call processing is behaving just as it was designed to, but some users may be momentarily puzzled."

Discussion. This mode confusion is very similar to the previous one. The cause for the mode confusion is the same.

3.3 Example 3 – 911 and Three-Way Calling (911 & TWC)

The following example has been presented by several other authors in a more dangerous and also more concise variant. We also include this variant, using our own words.

Description [2]. "A Three-Way Calling subscriber must put the second party on hold before bringing a third party into the conversation. However, the 911 feature[4] prevents anyone from putting a 911 operator on hold. Suppose that **A** wishes to aid a distressed friend **B** by connecting **B** to a 911 operator using the Three-Way Calling service. If **A** calls **B** first and then calls 911, **A** can establish the three-way call, since **A** still has control of putting **B** on hold before calling 911. However, if **A** calls 911 first, then **A** cannot put the 911 operator on hold to call **B**; therefore **A** cannot make the three-way call. [. . .]"

Description of a more dangerous variant (911 and Consultation Call). "**A** calls **B**; **B** suddenly gets distressed, and **A** does a successful Consultation Call to the 911 operator. Then **A** wants to switch back to **B** for further information, but **A** can't do this because **A** first must put the 911 operator on hold, which is prevented by the 911 feature."

Discussion. The second scenario in the upper description where **A** cannot make the desired Three-Way Call is a mode confusion. **A** would succeed in a normal call, but does not in a 911 call. It is likely that **A** is not aware of the mode change concerning call control in an emergency situation. Again, when **A** plans the three-way call, **A** will concentrate on the "relevant" part of the behaviour of the features for efficiency. **A** does not intend to play tricks on the 911 operator and therefore excludes call control aspects from his abstracted mental model of the system.

[4] In North America, dialling 911 connects to an emergency service.

Discussion of the variant. The system and the abstracted mental model of it of **A** are clearly in different modes here. The cause is the same as for the mode confusion above. A correct plan for **A** would have been to initiate a Three-Way Calling call instead of a Consultation Call.

3.4 Example 7 – Credit-Card Calling and Voice-Mail service (CCC & VM)

Description [2]. "Instead of hanging up and then dialling the long distance access code again, many credit-card calling services instruct callers to press [#] for placing another credit-card call. On the other hand, to access voice mail messages from phones other than his/her own, a subscriber of some Voice-Mail service such as *Aspen* can (1) dial the Aspen service number, (2) listen to the introductory prompt (instruction), (3) press [#] followed by the mailbox number and passcode to indicate that the caller is the subscriber, and then (4) proceed to check messages. However, when a customer places a credit-card call to Aspen, the customer does not know exactly when the Credit-Card Calling feature starts passing signals to Aspen instead of interpreting them itself. Suppose that **A** has frequently called Aspen and knows how to interact with Aspen. When **A** places a credit-card call to Aspen, **A** may hit [#] immediately without waiting for the Aspen's introductory prompt. However, the [#] signal could be intercepted by the credit-card call feature; hence it is interpreted as an attempt to make a second call."

Discussion. The abstracted mental model of the system already has switched to the voice mail mode, while the system still is in credit-card mode. The user wanted to plan a shortcut in the voice mail service. In order to do this, he constructed an abstraction of the relevant parts of the telephone system. But in the abstraction step he made the mistake of dropping the entire Credit-Card Calling feature, even though it was still latently active.

Even with a correct abstraction, a potential for mode confusion remains. The user cannot observe when the system switches from credit-card mode to voice-mail mode. This happens somewhere between dialling the last digit of the Aspen access number and the end of Aspen's introductory prompt. We have an implicit mode change, again. The user can re-synchronize only by waiting, as above.

3.5 Example 10 – Call Waiting and Personal Communication Services (CW & PCS)

Description [2]. "Call Waiting is a feature assigned to a *directory number*. However, Call Waiting uses the status of the *line* with which the number is associated to determine whether the feature should be activated: at present in a public switched telephone network, if a non-ISDN line is in use, then it is busy; a second call to the same line will trigger the switching element to send out a call-waiting tone. PCS[5] customers may not all be subscribers of Call Waiting. Suppose that **X** and **Y** are both PCS customers currently registered with the same CPE[6]; **X** has Call Waiting but **Y** does not. We further assume that **Y** is on the phone when somebody calls **X**. Since **X** has Call Waiting and the line is busy, the new call triggers the Call Waiting feature of **X**. But is it legitimate to send the call-waiting alert through the line to interrupt **Y**'s call? If not, then **X**'s Call Waiting feature is ignored."

[5] personal communication services
[6] customer premises equipment

Discussion. If **Y** is alerted, this will cause a mode confusion for **Y** due to incorrect knowledge about the system. **Y** has no Call Waiting and does not know about the additional alerting mode the system can get into. When **Y** is alerted, **Y** probably does not know how to leave the mode that causes this annoying signal.

Personal Communication Services (PCS) will show mode confusions when combined with several other features, too. When a user shares a line with other users through PCS, his/her mental model of the system must also comprise a significant part of the features of the other users. Since this often will not be the case, the system can easily get into modes he/she does not know of, similar as in the above example.

3.6 Example 14 – Call Waiting and Call Waiting (CW & CW)

We have two mode confusion problems here.

Description, first part [2]. "Call Waiting allows a subscriber to put the other party on hold. However, it does not protect the subscriber from being put on hold. Confusion can arise when two parties exercise this type of control concurrently. Suppose that both **A** and **B** have Call Waiting, and **A** has put **B** on hold to talk to **C**. While on hold, **B** decides to flash the hook to answer an incoming call from **D**, which puts **A** on hold as well. If **A** then flashes the hook expecting to get back to the conversation with **B**, **A** will be on hold instead, unless either **B** also flashes the hook to return to a conversation with **A** or **D** hangs up automatically returning **B** to a conversation with **A**."

Discussion. **A**'s mental model of the system does not include the possibility of being put on hold while exercising the Call Waiting feature. This incorrect knowledge about the system causes the mode confusion.

Description, second part [2]. "An ambiguous situation arises, when **B** hangs up on the conversation with **D** while **A** is still talking to **C**; there are two separate contexts in which to interpret **B**'s action. Assume that CW1 refers to the Call Waiting call among **C-A-B** and CW2 refers to the one among **A-B-D**. According to the specification of Call Waiting, in the context of CW2 **B** will be rung back (because **A** is still on hold) and, upon answering, become the held party in the CW1 context and hear nothing. But in the context of CW1 the termination **B** will be interpreted as simply a disconnection, thus **A** and **C** are placed in a normal two-way conversation, and **B** is idled. The question is: Should **B** be rung back or should **B** be idled?"

Discussion. **B**'s incorrect knowledge about the implemented system can cause a mode confusion. The incomplete requirements specification for this combinations of features must be disambiguated by the implementers, in one way or the other. The choice is not obvious. User **B** must also disambiguate the situation, but may well take the opposite choice, even without noticing that there are others. In case that **B** expects to be idled after hanging up, but actually is rung back, this results in a mode confusion for **B**. **B** interprets the ringing as a new incoming call but then hears nothing when answering. In case that **B** is idled but expects to be rung back, this results in a period of mode confusion while **B** waits to be called back in vain.

3.7 Example 15 – Call Waiting and Three-Way Calling (revisited) (CW & TWC / 2)

Description [2]. "Consider how Call Waiting and Three-Way Calling might interact in the situations where a user can exercise both features simultaneously on the same line. The call control relationship can now become quite complicated. Suppose that **A** has both Call Waiting and Three-Way Calling, and **A** is talking to **B**. Now **C** calls **A**, so **A** uses Call Waiting to put **B** on hold and talks to **C**. **A** may decide to have **B** join his conversation with **C**, so he puts **C** on hold, makes a second call to **B**, and after **B** answers the call with Call Waiting, **A** brings **C** back into the conversation to establish a three-way call. There are three contexts in this establishment: a Call Waiting call and a Three-Way Calling call, both established by **A** among **B-A-C**, and a Call Waiting call established by **B** as **A-B-A**. Now, if **B** hangs up, then according to the contexts established by **A**, the session becomes a two-way call between **A** and **C**; according to the contexts established by **B** though, **B** should get a ring-back because **B** still has **A** on hold."

Discussion. **B**'s incorrect knowledge about the implemented system can cause a mode confusion exactly as in the previous example.

3.8 Example 19 – Long distance calls and Message Rate Charge services (LDC & MRC)

Description [2]. "Each long distance call consists of at least three segments – two local accesses at each end and one provided by an interexchange carrier in between. Should a customer be charged for the segments that have been successfully completed even if the call did not reach its final destination? Would it be counted as one unit toward the total local units allowed per month for a Message Rate Charge service?"

Discussion. First of all, this is a problem of ambiguous requirements, but there can also be a mode confusion because of incorrect knowledge about the behaviour of the system. Because of the difficult billing questions, the user can easily have false expectations on the behaviour. A call segment may be in a charged connection mode earlier than expected. The user will notice this confusion only much later, when he receives the bill. For example, he might be charged for long distance call attempts that he knows were never completed. An overrun of the allowed Message Rate Charge units could in principle also be a surprise, but it is much less likely that the user really counts all his local calls.

3.9 Example 20 – Calling from hotel rooms (Hotel)

Description [2]. "Many hotels contract with independent vendors to collect access charges for calls originated from phones in their premises. Without being able to access to the status of call connections, some billing applications developed by these vendors use a fixed amount of time to determine if a call is complete or not – thus one can be billed for incomplete calls that rang a long time, or not billed for very short duration calls (even long distance)."

Discussion. This is another mode confusion with a particularly long delay. The user will detect it several days later when paying the hotel bill. The cause is incorrect knowledge about

the behaviour of the system. The user is usually not informed about the unusual way to determine the start of billing by a timer and incorrectly assumes that completing a connection starts billing. In case that the user knows the behaviour of the system correctly, a mode confusion due to the implicit state change can still occur, that is, due to incorrect observation. This happens when the system and the user perceive a call duration just below / just above the threshold due to imprecise time measurement.

3.10 Benchmark Feature Interactions Which are No Mode Confusions

The remaining examples present undesired feature interactions with a wide range of causes. Often, there are ambiguous, incomplete, or conflicting requirements. In some examples, the restrictions of the current implementation cause a problem, or the implementation is just deficient. In all of these examples, there are either no surprising modes, or the user is not actively involved.

3.11 A Non-Benchmark Example – Key Lock and Volume Adjust (Lock & Vol)

This example is not from the benchmark, but its causes are particularly interesting with respect to our classification. Our colleague Axel Lankenau experienced this problem, we report it here.

Description (by ourselves). "Our colleague's mobile phone has the feature to lock its keys. This prevents unintended commands while carrying it in the pocket. The lock mode is indicated permanently by a small key symbol on the display. The lock can be released only by pressing the pound key for a long time. If any key is pressed, the lock mode is shown clearly on the display ("press '#' to unlock"), such that the user knows that he must unlock the phone before any further usage. There is one exception to the lock: when the phone rings, the user can press the hook button to accept the call. The phone remains locked otherwise. The phone also has two buttons at the side of its case which allow to adjust the volume of the speaker. It happened that our colleague carried his phone in the pocket in locked mode, and the phone rang. He took out the phone, pressed the hook button to accept the call, and held the phone to his ear. He then noticed that the volume level was not right and tried to adjust it. This did not work, and it surprised and annoyed him."

Discussion. This problem has three causes: incorrect processing by the user, incorrect observation for psychological reasons, and incorrect observation for physical reasons. First, there was a slip of memory when our colleague did not remember that his phone was in locked mode after some time of non-use. This was incorrect processing. Second, he committed a lapse as he did not look at the display while accepting the call. He can do it without looking and therefore missed to check the mode. This was an incorrect observation for psychological reasons. Afterwards, he could not see the display of the phone while it was close to his ear. This was an incorrect observation for physical reasons.

A solution to the problem could be: when the user presses any key in the locked mid-call mode, the phone not only shows a textual warning message, but also generates an unambiguous warning beep tone. Another solution could be to redesign (and weaken) the lock feature such that it releases the lock entirely when the user accepts a call.

Table 2: the causes of the mode confusions in the benchmark and of our one extra example.

cause \ ID / no.	CW&TWC 2	POTS&POTS 2b	911&TWC 3	CCC&VM 7	CW&PCS 10	CW&CW 14	CW&TWC/2 15	LDC&MRC 19	Hotel 20	Lock&Vol —
incorrect observation									•	••
incorrect knowledge				•	•	••	•	•	•	
incorrect abstraction	•	•	•	•						
incorrect processing										•

4 Does the Safety-Critical System's Notion of Mode Confusion Work in Telephony?

4.1 Is the Classification of Causes Useful For Telephony?

Our classification of causes for mode confusions in safety-critical systems shows a distribution of causes in telephony which is still reasonable. Table 2 classifies all mode confusions in the feature interaction benchmark of Cameron *et.al.* [2] and also our one extra example according to their causes. The dominant cause is incorrect knowledge of the user about the system (7 cases). Also important is an incorrect abstraction of the user's knowledge to the relevant parts of it (4 cases). Rare is an incorrect observation by the user (1 case). One cause does not appear in the benchmark, but in our extra example: incorrect processing by the user (1 case). This extra example also shows more incorrect observations by the user (2 cases). One cause does not appear at all: incorrect knowledge of the user about his/her own senses.

Two of the four classes appear to be less important for telephony: incorrect observation by the user and incorrect processing by the user. One can suspect that this is only because the authors of the benchmark concentrated on "technical" problems and were not interested in human factors problems. But we would need more empirical data to support this.

If incorrect observations should not be relevant, this is not a problem for our approach. The observation relation SENSE just becomes a one-to-one mapping. It must remain nevertheless in the rigorous definition of mode confusion. It ensures "type correctness" for the events.

If incorrect processing by the user should occur only rarely, this is even an advantage for our approach. The user then sticks more closely to the mental model on which our mode confusion analysis is based.

4.2 What Does Not Fit?

Our definition of mode confusion for safety-critical systems does not fit nicely for telephone switching systems with one respect: the user does not abstract to safety-relevant aspects of the system, but to the set of telephony features which are relevant currently.

4.3 How Can We Adapt the Definition to Telephone Switching Systems?

We must use a different kind of abstraction. The telephone user does not abstract to the safety-relevant behaviour $\mathrm{REQ}^{\mathrm{M}}_{\mathrm{SAFE}}$. Instead, the user abstracts his/her knowledge about the

behaviour of the telephone switching system $\mathrm{REQ}^{\mathrm{M}}$ to the behaviour $\mathrm{REQ}^{\mathrm{M}}_{\mathrm{R}}$ of those features which are "relevant".

The relevant features are those that are currently active or can become active. They must be active for the user considered and in the scope of time considered. A feature is active if it can contribute to the visible behaviour of the system. A feature can be activated either by the user considered or by another user in the telephone network. The relevant scope of time ends when all features involved become inactive. Often this is the end of the current call. Sometimes, the scope of time is not limited at all, if the effects of a feature's behaviour are permanent. An example is the billing of calls from hotel rooms. The money spent will never return.

We can abstract the actual behaviour of the entire telephone switching system REQ to the behaviour of the relevant features $\mathrm{REQ}_{\mathrm{R}}$ in the same way as the user abstracts $\mathrm{REQ}^{\mathrm{M}}$ to $\mathrm{REQ}^{\mathrm{M}}_{\mathrm{R}}$. The same holds for $\mathrm{SENSE}/\mathrm{SENSE}_{\mathrm{R}}$ and $\mathrm{SENSE}^{\mathrm{M}}/\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{R}}$. We need all these for the adapted definition of mode confusion. The same criterion for the relevance of a feature applies. The adapted definition is:

Definition 2 (Mode confusion in telephony). A *mode confusion in telephony* between $\mathrm{SENSE}_{\mathrm{R}}(\mathrm{REQ}_{\mathrm{R}})$ and $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{R}}(\mathrm{REQ}^{\mathrm{M}}_{\mathrm{R}})$ occurs if and only if $\mathrm{SENSE}_{\mathrm{R}}(\mathrm{REQ}_{\mathrm{R}})$ is not a failure refinement of $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{R}}(\mathrm{REQ}^{\mathrm{M}}_{\mathrm{R}})$, i.e., iff $\mathrm{SENSE}^{\mathrm{M}}_{\mathrm{R}}(\mathrm{REQ}^{\mathrm{M}}_{\mathrm{R}}) \quad \not\sqsubseteq_F \quad \mathrm{SENSE}_{\mathrm{R}}(\mathrm{REQ}_{\mathrm{R}})$.

5 How Can the Notion of Mode Confusion Help Against Feature Interactions?

Attention to mode confusions helps to design features and sets of features with less undesired surprises.

The design should help the user to abstract his/her mental model. The correct abstraction to the relevant features is difficult for a user. Table 2 shows this. In particular, it is difficult to determine which features are active currently.

Enhanced feedback helps. The system must notify the user when a feature becomes active or inactive. For example remember the interaction between credit-card calling and voice mail in benchmark example 7. The announcement of the voice mail service must make clear the point of time from which on voice mail commands may be entered. The credit card feature must announce from which point of time on its command processing is suspended. Unfortunately, we cannot improve the feedback by a richer hardware interface, for example with many indicator lights. The hardware costs prevent us from installing better customer premises equipment everywhere. Improved switch-side feedback is possible, though. An example are announcements.

The designer of a new feature must always check if it is obvious to the user whether the feature is active or not. The designer must also check whether the user perceives the feature as a single entity. If necessary, the design must be changed.

Correct abstraction is easier if the active features are simple and if only a few are active.

The design should limit the complexity of the abstracted mental model. Two factors in particular increase this complexity: a long duration of feature activation and non-determinism.

The longer a feature is active, the higher is the chance that its period of activity overlaps the period of activity of other features. This increases the number of features that the user

must include into his/her abstracted mental model. A feature should terminate its activity after the end of a call, if possible. The purpose of some features does not allow this. In this case the user should get appropriate feedback on the set of active features, at least.

It is difficult for a user to interpret an observed sequence of events correctly on a non-deterministic model. This requires to follow simultaneously several alternative paths in the model. It can exhaust a user's mental memory capacity soon. The system therefore must give the user an immediate feedback signal about any internal choice that changes its behaviour.

A distributed system such as the telephone network inherently exhibits a lot of non-determinism to the individual user. The user cannot perceive what other users do. The user's senses SENSE mask out all telephone usage events in the world except of the local events. Again, feedback for relevant events is necessary. For example, remember the interaction of the call waiting feature and the three-way calling feature in the benchmark example 2. It persists even after a precedence rule has been added. We can avoid the problematic race condition if the newly activated call waiting feature proceeds in two steps: first, it informs the user about its activation. Second, only after it has completed this, it accepts hook flashes. For this, the feature could either use two signals with, e. g., a delay of one second, or it could just make the system ignore hook flashes for one second, starting with the signal tone, until the user must have noticed the mode change.

The design should help the user to know the system correctly. A prospective user must be able to learn the behaviour of a feature easily. Either it must be intuitive to use, or the user must be trained suitably. Our telephone provider offers us a few use-cases only as the description of a newly provisioned feature. These leave many questions about the behaviour open. Research on better teaching material is required here.

A feature must be redesigned if the user cannot learn its behaviour. An example is the activation of Call Waiting (CW) and Personal Communication Services (PCS) on the line of another PCS user not subscribed to CW (example 10 in the benchmark). CW-PCS must not "hi-jack" the line without telling its current user what is going on.

An improved development process. The operator directed design process ODP for avionics by Vakil and Hansman, Jr. [23] produces the user training material even before the software specification. If the system appears to be difficult to handle for its users, it is redesigned immediately. The same process can be applied to telephone switching.

Rushby [3] proposes an iterated development process for shared-control systems which model-checks an abstracted system against an abstracted mental model. The mental model is derived from the user training material such that it matches the mental model of an average user. The goal is to detect potential for mode confusions early. Our rigorous definition of mode confusion is a suitable foundation for such a tool supported development process, both for shared-control systems and in telephony.

A potential obstacle for this model-checking approach is the complexity of a complete telephone switching system. The analysis might be infeasible because of the state space explosion problem. A potential solution is to analyse small sets of features at a time. We need statistical information for this about which features are used together most often. These combinations are most likely to annoy customers if they are prone to mode confusions.

Online mode confusion detection and resolution. Shared-control systems can be designed with an "intelligent" interface component. This component monitors the behaviour of both the rest of the system and of the user. If it detects potential for mode confusion in the current situation, it becomes active and resolves the problem. For example, it may give the user additional information about the mode in which the system currently is. When there is no problem ahead, it is silent and does not distract the user. It detects mode confusions online, at run-time. In each situation, the component model-checks the currently active features only. We do not need to model-check the entire system. This reduces the complexity of the analysis. A potential difficulty are the increased computational costs for the switch. A research project on this kind of intelligent interface started at the University of Bremen in the end of 2002. The application domain of this project is spacial cognition and robotics. But we expect that we can transfer its results to telephony directly.

6 What Remains to Be Done?

The aim of this paper is to present the new way in which one can view and tackle feature interactions. We demonstrate that many undesired telephony feature interactions are also shared-control mode confusions. Several measures for preventing mode confusions are known in the literature on human-computer interaction. We show that these measures can be applied to this kind of feature interaction. We sketch several more measures for the telephony domain. The next step should be to apply the existing ideas in the literature to feature interactions practically, and to work out the ideas sketched in the previous section.

The relation between ease of abstraction and the structure of the features deserves more research. Are there any additional feature design rules that help the user to abstract to the currently active set of feature behaviour correctly? Research in shared-control systems is interested in the minimal safe mental model [22, 25]. This model is the "smallest" abstraction that is failure equivalent to the safety-relevant part of the behaviour of the technical system [24]. We should find out how the user can have smaller abstracted mental models of the telephone switching system without experiencing mode confusions.

Acknowledgements

We thank Dr. Axel Lankenau for sharing his invaluable expertise on mode confusions with us.

References

[1] CHARLES E. BILLINGS. "Aviation automation: the search for a human-centered approach". Human factors in transportation. Lawrence Erlbaum Associates Publishers, Mahwah, N.J. (1997).

[2] E. JANE CAMERON, NANCY D. GRIFFETH, YOW-JIAN LIN, ET AL.. A feature interaction benchmark in IN and beyond. In L. G. BOUMA AND HUGO VELTHUIJSEN, editors, "Feature Interactions in Telecommunications Systems", pages 1–23, Amsterdam (1994). IOS Press.

[3] JOHN RUSHBY. Modeling the human in human factors. In UDO VOGES, editor, "Computer Safety, Reliability and Security – 20th Int'l Conf., SafeComp 2001, Proc.", volume 2187 of "LNCS", pages 86–91, Budapest, Hungary (September 2001). Springer.

[4] D. A. NORMAN. Some observations on mental models. In D. GENTNER AND A. L. STEVENS, editors, "Mental Models". Lawrence Erlbaum Associates Inc., Hillsdale, NJ, USA (1983).

[5] J. J. CAÑAS, A. ANTOLÍ, AND J. F. QUESADA. The role of working memory on measuring mental models of physical systems. *Psicologíca* **22**, 25–42 (2001).

[6] J. CROW, D. JAVAUX, AND J. RUSHBY. Models and mechanized methods that integrate human factors into automation design. In K. ABBOTT, J.-J. SPEYER, AND G. BOY, editors, "Proc. of the Int'l Conf. on Human-Computer Interaction in Aeronautics: HCI-Aero 2000", Toulouse, France (September 2000).

[7] R. W. BUTLER, S. P. MILLER, J. N. POTT, AND V. A. CARREÑO. A formal methods approach to the analysis of mode confusion. In "Proc. of the 17th Digital Avionics Systems Conf.", Bellevue, Washington, USA (1998).

[8] JOHN RUSHBY. Analyzing cockpit interfaces using formal methods. In H. BOWMAN, editor, "Proc. of FM-Elsewhere", volume 43 of "Electronic Notes in Theoretical Computer Science", Pisa, Italy (October 2000). Elsevier.

[9] RACHID HOURIZI AND PETER JOHNSON. Beyond mode error: Supporting strategic knowledge structures to enhance cockpit safety. In "HCI 2001, People and Computers XIV", Lille, France (10–14 September 2001). Springer.

[10] N. SARTER AND D. WOODS. How in the world did we ever get into that mode? Mode error and awareness in supervisory control. *Human Factors* **37**(1), 5–19 (1995).

[11] A. DEGANI, M. SHAFTO, AND A. KIRLIK. Modes in human-machine systems: Constructs, representation and classification. *Int'l Journal of Aviation Psychology* **9**(2), 125–138 (1999).

[12] BETTINA BUTH. "Formal and Semi-Formal Methods for the Analysis of Industrial Control Systems". Habilitation thesis, University of Bremen, Germany (2001).

[13] N. G. LEVESON, L. D. PINNEL, S. D. SANDYS, S. KOGA, AND J. D. REESE. Analyzing software specifications for mode confusion potential. In "Workshop on Human Error and System Development", Glasgow, UK (1997).

[14] GAVIN DOHERTY. "A Pragmatic Approach to the Formal Specification of Interactive Systems". PhD thesis, University of of York, Dept. of Computer Science (October 1998).

[15] HAROLD THIMBLEBY. "User Interface Design". ACM Press, New York, USA (1990).

[16] PETER WRIGHT, BOB FIELDS, AND MICHAEL HARRISON. Deriving human-error tolerance requirements from tasks. In "Proc. of ICRE'94 – IEEE Int'l. Conf. on Requirements Engineering", Colorado, USA (1994).

[17] JAN BREDEREKE AND AXEL LANKENAU. A rigorous view of mode confusion. In STUART ANDERSON, SANDRO BOLOGNA, AND MASSIMO FELICI, editors, "Computer Safety, Reliability and Security – 21st Int'l Conf., SafeComp 2002, Proc.", volume 2434 of "LNCS", pages 19–31, Catania, Italy (September 2002). Springer.

[18] A. W. ROSCOE. "The Theory and Practice of Concurrency". Prentice-Hall (1997).

[19] C. A. R. HOARE. "Communicating Sequential Processes". Prentice-Hall (1985).

[20] M. A. SASSE. "Eliciting and Describing Users' Models of Computer Systems". PhD thesis, School of Computer Science, The University of Birmingham (April 1997). Available online at www.cs.ucl.ac.uk/staff/A.Sasse/thesis/LINK_ME.html.

[21] JAMES REASON. "Human error". Cambridge University Press (1990).

[22] D. JAVAUX. Explaining Sarter & Woods' classical results. The cognitive complexity of pilot-autopilot interaction on the Boeing 737-EFIS. In "Proc. of HESSD '98", pages 62–77 (1998).

[23] S. S. VAKIL AND R. J. HANSMAN, JR. Approaches to mitigating complexity-driven issues in commercial autoflight systems. *Reliability Engineering & System Safety* **72**(2), 133–145 (February 2002).

[24] AXEL LANKENAU. "The Bremen Autonomous Wheelchair 'Rolland': Self-Localization and Shared Control". PhD thesis, University of Bremen, Germany (August 2002).

[25] DENIS JAVAUX. A method for predicting errors when interacting with finite state systems. How implicit learning shapes the user's knowledge of a system. *Reliability Engineering & System Safety* **72**(2), 147–165 (February 2002).