A Rigorous View of Mode Confusion

Jan Bredereke and Axel Lankenau

Universität Bremen
FB 3, P.O. box 330 440, D-28334 Bremen, Germany
{brederek,alone}@tzi.de
www.tzi.de/{~brederek,~alone}
Fax: +49-421-218-3054

Abstract. Not only in aviation psychology, mode confusion is recognised as a significant safety concern. The notion is used intuitively in the pertinent literature, but with surprisingly different meanings. We present a rigorous way of modelling the human and the machine in a shared-control system. This enables us to propose a precise definition of "mode" and "mode confusion". In our modelling approach, we extend the commonly used distinction between the machine and the user's mental model of it by explicitly separating these and their safety-relevant abstractions. Furthermore, we show that distinguishing three different interfaces during the design phase reduces the potential for mode confusion. A result is a new classification of mode confusions by cause, leading to a number of design recommendations for shared-control systems which help to avoid mode confusion problems. A further result is a foundation for detecting mode confusion problems by model checking.

1 Introduction and Motivation

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 "receiver 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. So, it's all around — but what exactly is a mode, what defines a mode confusion situation and how can we detect and avoid automation surprises?

As long as we have no rigorous definition, we should regard a *mode confusion* as one kind of an automation surprise. It refers to a situation in which a technical system can behave differently from the user's expectation. Whereas mode confusions in typical human-computer interactions, such as the word processor example mentioned above, are "only" annoying, they become dangerous if we consider safety-critical systems.

S. Anderson et al. (Eds.): SAFECOMP 2002, LNCS 2434, pp. 19–31, 2002.

Today, many safety-critical systems are so-called embedded shared-control systems. These are interdependently controlled by an automation component and a user. Examples are modern aircraft, automobiles, but also intelligent wheelchairs. We focus on such shared-control systems in this paper and call the entirety of technical components *technical system* and the human operator *user*. Note that we have to take a black-box stand, i. e. we can only work with the behaviour of the technical system observable at its interfaces: since we want to solve the user's problems, we have to take his or her point of view, which does not allow access to internal information of the system.

As Rushby points out [1], in cognitive science it is generally agreed upon that humans use so-called *mental models* when they interact with (automated) technical systems. Since there are at least two completely different interpretations of the notion "mental model" in the pertinent literature, it is important to clarify that we refer to the one introduced by Norman [2]: A mental model represents the user's knowledge about a technical system, it consists of a naïve theory of the system's behaviour. According to Rushby [1], an explicit description of a mental model can be derived, e. g., in form of a state machine representation, from training material, from user interviews, or by user observation.

We briefly recapitulate the pertinent state of the art here. It remains surprisingly unclear what a mode as such is. While some relevant publications give no [3, 4] or only an implicit definition [5, 6] of the notions "mode" and "mode confusion", there are others that present an explicit informal definition [7, 8, 9, 10]. Doherty [11] presents a formal framework for interactive systems and also gives an informal definition of "mode error". Wright and colleagues give explicit but example driven definitions of the notions "error of omission" and "error of commission" by using CSP to specify user tasks [12].

Interestingly, the way of modelling often seems to be influenced significantly by the tool that is meant to perform the final analysis. Degani and colleagues use State Charts to model separately the technical system and the user's mental model [13]. Then, they build the composition of both models and search for certain states (so-called "illegal" and "blocking" states) which indicate mode confusion potential. Butler *et al.* use the theorem prover PVS to examine the flight guidance system of a civil aircraft for mode confusion situations [4]. They do not consider the mental model of the pilot as an independent entity in their analysis. Leveson and her group specify the black-box behaviour of the system in the language SpecTRM-RL that is both well readable by humans and processible by computers [10, 14, 15]. In [10], they give a categorisation of different kinds of modes and a classification of mode confusion situations. Rushby and his colleagues employ the Murϕ model-checking tool [16, 5, 3]. Technical system and mental model are coded together as a single set of so-called Murϕ rules. Lüttgen and Carreño examine the three state-exploration tools Murϕ, SMV, and Spin with respect to their suitability in the search for mode confusion potential [17]. Buth [9] and Lankenau [18] clearly separate the technical system and the user's mental model in their CSP specification of the well-known MD-88-"kill-the-capture" scenario and in a service-robotics example, respectively.

The support of this clear separation is one reason why Buth's comparison between the tool Murϕ and the CSP tool FDR favours the latter [9, pages 209-211]. Almost all publications refer to aviation examples when examining a case study: an MD-88 [19, 7, 10, 16, 9], an Airbus A320 [3, 6], or a Boeing 737 [5].

Rushby proposes a procedure to develop automated systems which pays attention to the mode confusion problem [1]. 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 [20, 3].

Hourizi and Johnson [6, 21] generally doubt that avoiding mode confusions alone helps to reduce the number of plane crashes caused by automation surprises. They claim that the underlying problem is not mode confusion but what they call a "knowledge gap", i. e. the user's insufficient perception prevents him or her from tracking the system's mode.

As far as we are aware, there is no publication so far that defines "mode" and "mode confusion" rigorously. Therefore, our paper clarifies these notions. Section 2 introduces to the domain of our case study, which later serves as a running example. Section 3 and 4 present a suitable system modelling approach and clarify different world views, which enables us to present rigorous definitions in Sect. 5. Section 6 works out the value of such definitions, which comprises a foundation for the automated detection of mode confusion problems and a classification of mode confusion problems by cause, which in turn leads to recommendations for avoiding mode confusion problems. A summary and ideas for future work conclude the paper.

2 Case Study Wheelchair

Our case study has a service robotics background: we examine the cooperative obstacle avoidance behaviour of our wheelchair robot. The Bremen Autonomous Wheelchair "Rolland" is a shared-control service robot which realizes intelligent and safe transport for handicapped and elderly people [22, 23]. The vehicle is a commercial off-the-shelf power wheelchair. It has been equipped with a control PC, a ring of sonar sensors, and a laser range finder. Rolland is jointly controlled by its user and by the software. Depending on the active operation mode, either the user or the automation is in charge of driving the wheelchair.

3 Precise Modelling

Before we can discuss mode confusion problems, some remarks on modelling a technical system in general are necessary. The user of a running technical system has a strict *black-box view*. Since we want to solve the user's problems, we must take the same black-box point of view. This statement appears to be obvious, but has far-reaching consequences for the notion of mode. The user has no way of observing the current internal state, or mode, of the technical system.

Nevertheless, it is possible to describe a technical system in an entirely black-box view. Our software engineering approach is inspired by the work of Parnas [24, 25], even though we start out with events instead of variables, as he does.

We can observe (only) the environment of the technical system. When something relevant happens, we call this an *event*. When the technical system is the control unit of an automated wheelchair, then an event may be that the user pushes the joystick forward, that the wheelchair starts to move, or an event may as well be that the distance between the wheelchair and a wall ahead becomes smaller than the threshold of 70 cm.

The technical system has been constructed according to some *requirements* document REQ. It contains the requirements on the technical system, which we call SYSREQ, and those on the system's environment NAT. However, if we deal with an existing system for which no (more) requirements specification is available, it might be necessary to "reverse engineer" it from the implementation.

For the wheelchair, SYSREQ should state that the event of the wheelchair starting to move follows the event that the joystick is pushed forward. SYSREQ should also state what happens after the event of approaching a wall. Of course, the wheelchair should not crash into a wall in front of it, even if the joystick is pushed forward. We can describe this entirely in terms of observable events, by referring to the *history* of events until the current point of time. If the wheelchair has approached a wall, and if it has not yet moved back, it must not move forward further. For this description, no reference to an internal state is necessary.

In order to implement the requirements SYSREQ on a technical system, one usually needs several assumptions about the environment of the technical system to construct. For example, physical laws guarantee that a wheelchair will not crash into a wall ahead unless it has approached it closer than 70 cm and has continued to move for a certain amount of time. We call the documentation of assumptions about the environment NAT. NAT must be true even before the technical system is constructed. It is the implementer's task to ensure that SYSREQ is true provided that NAT holds.

4 Clarification of World Views

4.1 Where are the Boundaries?

The control software of a technical system cannot observe physical events directly. Instead, the technical system is designed such that sensor devices generate internal input events for the software, and the software's output events are translated by actuator devices into physical events, again. Neither sensors nor actuators are perfectly precise and fast, therefore we have a distinct set of software events. Accordingly, the requirements on the technical system and the requirements on the software cannot be the same. For example, the wheelchair's ultrasonic distance sensors for the different directions can be activated in turns only, resulting in a noticeable delay for detecting obstacles. We call the software

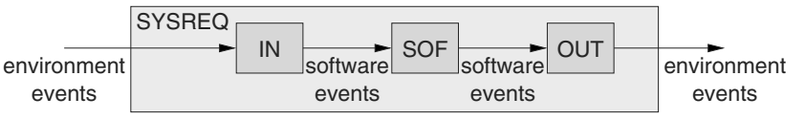

Fig. 1. System requirements SYSREQ vs. software requirements SOF

requirements SOF, the requirements on the input sensors IN and the requirements on the output actuators OUT. Figure 1 shows the relationships among them. An important consequence is that the software SOF must compensate for any imperfectness of the sensors and actuators so that the requirements SYSREQ are satisfied. When defining SOF, we definitely need to take care whether we refer to the boundary of SOF or of SYSREQ.

This becomes even more important when we consider the user who cooperates with the technical system. He or she observes the same variables of the environment as the technical system does. But the user observes them through his/her own set of senses SENS. SENS has its own imperfections. For example, a human cannot see behind his/her back. Our automated wheelchair will perceive a wall behind it when moving backwards, but the user will probably not. Therefore, we need to distinguish what actually happens in reality (specified in REQ, i. e. the composition of SYSREQ and NAT) from the user's mental model MMOD of it. When making a statement about MMOD, we definitely need to take care whether we refer to the boundary of MMOD or of REQ.

When we define the interfaces precisely, it turns out that there is an obvious potential for a de-synchronisation of the software's perception of reality with the user's perception of it. And when we analyse this phenomenon, it is important to distinguish between the three different interfaces: *environment* to machine (or to user), *software* to input/output devices, and *mental* to senses. As a result, we are able to establish a precise relation between reality as it is perceived by the user and his/her mental model of it. This relation will be the base of our definition of mode confusion.

4.2 Brief Introduction to Refinement

As will be explained later, we use a kind of specification/implementation relation in the following sections. Such relations can be modelled rigorously by the concept of *refinement*. There exist a number of formalisms to express refinement relations. We use CSP [26] as specification language and the refinement semantics proposed by Roscoe [27]. One reason is that there is good tool support for performing automated refinement checks of CSP specifications with the tool FDR [27]. This section shall clarify the terminology for readers who are not familiar with the concepts.

In CSP, the behaviour of a process P is described by the set $traces(P)$ of the event sequences it can perform. Since we must pay attention to what can be

done as well as to what can be *not* done, the traces model is not sufficient in our domain. We have to enhance it by so-called *failures*.

Definition 1 (Failure). *A* failure *of a process P is a pair (s, X) of a trace s ($s \in traces(P)$) and a so-called* refusal *set X of events that may be blocked by P after the execution of s.*

If an output event o is in the refusal set X of P, and if there also exists a continuation trace s' which performs o, then process P may decide internally and non-deterministically whether o will be performed or not.

Definition 2 (Failure Refinement). *P refines S in the failures model, written $S \sqsubseteq_F P$, iff $traces(P) \subseteq traces(S)$ and also $failures(P) \subseteq failures(S)$.*

This means that P can neither accept an event nor refuse one unless S does; S can do at least every trace which P can do, and additionally P will refuse not more than S does. Failure refinement allows to distinguish between external and internal choice in processes, i.e. whether there is non-determinism. As this aspect is relevant for our application area, we use failure refinement as the appropriate kind of refinement relation.

4.3 Relation between Reality and the Mental Model

Our approach is based on the motto "*The user must not be surprised*" as an important design goal for shared-control systems. This means that the perceived reality must not exhibit any behaviour which cannot occur according to the mental model. 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. These two aspects are described by the notion of failure refinement, as defined in the previous section.

There cannot be any direct refinement relation between a description of reality and the mental model, since they are defined over different sets of events (i.e., environment/mental). We understand the user's senses SENS as a relation from environment events to mental events. SENS(REQ) is the user's perception of what happens in reality. The user is not surprised if SENS(REQ) is a failure refinement of MMOD. As a consequence, the user's perception of reality must be in an *implementation/specification relationship* to the mental model.

Please note that an equality relation always implies a failure refinement relation, while the converse is not the case. If the user does not know how the system will behave with regard to some aspect, but knows that he/she does not know, then he/she will experience no surprise nevertheless. Such indifference can be expressed mathematically by a non-deterministic internal choice in the mental model.

4.4 Abstractions

When the user concentrates on safety, he/she performs an on-the-fly simplification of his/her mental model MMOD towards the safety-relevant part

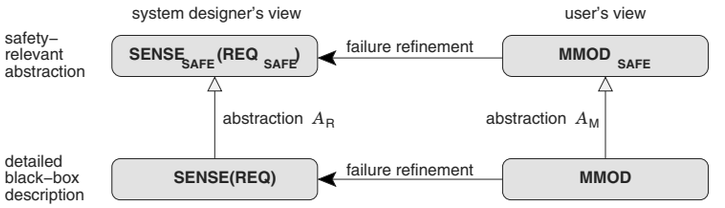

Fig. 2. Relationships between the different refinement relations

$\mathrm{MMOD}_{\mathrm{SAFE}}$. This helps him/her to analyse the current problem with the limited mental capacity. Analogously, 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: $\mathrm{MMOD}_{\mathrm{SAFE}}$ is defined over a set of abstracted mental 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 everything; 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. For example, it is irrelevant whether the wheelchair's speed is 81.5 or 82 cm/s when approaching an obstacle – all such events with a speed parameter greater than 80 cm/s will be abstracted to a single event with the speed parameter fast. Other transformations are hiding (or concealment [26]) and renaming. But the formalism also allows for arbitrary transformations of behaviours; a simple example being a certain event sequence pattern mapped onto a new abstract event. We use the abstraction functions $\mathcal{A}_R$ for REQ and $\mathcal{A}_M$ for MMOD, respectively.

The relation SENS from the environment events to the mental events must be abstracted in an analogous way. It should have become clear by now that SENS needs to be rather true, i. e., a bijection which does no more than some renaming of events. If SENS is "lossy", we are already bound to experience mode confusion problems. For our practical work, we therefore first make sure that SENS is such a bijection, and then merge it into REQ, even before we perform the actual abstraction step which enables the use of the model-checking tool.

Figure 2 shows the relationships among the different descriptions. In order that the user is not surprised with respect to safety, there must be a failure refinement relation on the abstract level between $\mathrm{SENS}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ and $\mathrm{MMOD}_{\mathrm{SAFE}}$, too.

5 A Rigorous View of Mode and of Mode Confusion

We will now present our rigorous definitions of *mode* and *mode confusion*. We will then motivate and discuss our choices.

In the following, let $\mathrm{REQ}_{\mathrm{SAFE}}$ be a safety-relevant black-box requirements specification, let $\mathrm{SENS}_{\mathrm{SAFE}}$ be a relation between environment events and mental events representing the user's senses, and let $\mathrm{MMOD}_{\mathrm{SAFE}}$ be a safety-relevant mental model of the behaviour of $\mathrm{REQ}_{\mathrm{SAFE}}$.

Definition 3 (Potential future behaviour). *A* potential future behaviour *is a set of failures.*

Definition 4 (Mode). *A* mode *of* $\mathrm{SENS}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ *is a potential future behaviour. And, a* mode *of* $\mathrm{MMOD}_{\mathrm{SAFE}}$ *is a potential future behaviour.*

Definition 5 (Mode confusion). *A* mode confusion *between* $\mathrm{SENS}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ *and* $\mathrm{MMOD}_{\mathrm{SAFE}}$ *occurs if and only if* $\mathrm{SENS}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ *is not a failure refinement of* $\mathrm{MMOD}_{\mathrm{SAFE}}$ *i.e., iff* $\mathrm{MMOD}_{\mathrm{SAFE}} \not\sqsubseteq_F \mathrm{SENS}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$.

After the technical system T has moved through a history of events, it is in some "state". Since we have to take a black-box view, we can distinguish two "states" only if T may behave differently in the future. We describe the *potential future behaviour* by a set of failures, such that we state both what T can do and what T can refuse to do. This definition of "state" is rather different from the intuition in a white-box view, but necessarily so.

Our next step to the notion of "mode" then is more conventional. We use the notion of "state", if at all, in the context of the non-abstracted descriptions. Two states of a wheelchair are different, for example, if the steerable wheels will be commanded to a steering angle of 30 degrees or 35 degrees, respectively, within the next second. These states are equivalent with regard to the fact of obstacle avoidance. Therefore, both states are mapped to the same abstracted behaviour by the safety-relevance abstraction function. We call such a distinct safety-relevant potential future behaviour a *mode*. Usually, many states of the non-abstracted description are mapped together to such a mode. On a formal level, both a state and a mode are a potential future behaviour. The difference between both is that there is some important safety-relevant distinction between any two modes, which need not be the case for two states.

We now can go on to *mode confusions*. The perceived reality and the user's mental model of it are in different modes at a certain point of time if and only if the perceived reality and the mental model might behave differently in the future, with respect to some safety-relevant aspect. Only if no such situation can arise in any possible execution trace, then there is no mode confusion. This means that the user's safety-relevant mental model must be a specification of the perceived reality. Expressed the other way around, the perceived reality must be an implementation of the user's safety-relevant mental model. This specification/implementation relationship can be described rigorously by failure refinement. If we have precise descriptions of both safety-relevant behaviours, we can

rigorously check whether a mode confusion occurs. Since model-checking tool support exists, this check can even be automated. Please note that we referred to the reality, as described by REQ, which not only includes the system's requirements SYSREQ but also the environment requirements NAT. This restricts the behaviour of SYSREQ by NAT: behaviour forbidden by physical laws is not relevant for mode confusions.

Our mathematical description allows for some interesting analysis of consequences. It is known in the literature that implicit mode changes may be a cause of mode confusion. In our description, an implicit mode change appears as an "internal choice" of the system, also known as a (spontaneous) "τ transition". The refinement relation dictates that any such internal choice must appear in the specification, too, which is the user's mental model in our case. This is possible: if the user expects that the system chooses internally between different behaviours, he/she will not be surprised, at least in principle. The problem is that the user must keep in mind all potential behaviours resulting from such a choice. If there is no clarifying event for a long time, the space of potential behaviours may grow very large and impractical to handle in practice.

6 Results

Our definitions form a *foundation for detecting* mode confusions by model-checking. It has opened new possibilities for a comprehensive analysis of mode confusion problems, which we currently explore in practice.

Our clarification of world views in Sect. 4 enables us to *classify* mode confusion problems into three classes:

1. *Mode confusion problems which arise from an* incorrect observation *of the technical system or its environment.*

   Formally, this is the case when SENS(REQ) is not a failure refinement of MMOD, but where SENS(REQ) would be a failure refinement of MMOD, provided the user's senses SENS would be a perfect mapping from environment events to mental events.
   The imperfections of SENS may have *physical* or *psychological* reasons: either the sense organs are not perfect; for example eyes which cannot see behind the back. Or an event is sensed, but is not recognised consciously; for example because the user is distracted, or because the user currently is flooded with too many events. ("Heard, but not listened to.")
   Please note that our notion of mode confusion problem also comprises the "knowledge gap" discussed in the research critique by Hourizi and Johnson [6, 21] (see Sect. 1). In our work, it appears as a mode confusion problem arising from an incorrect observation due to psychological reasons.

2. *Mode confusion problems which arise from* incorrect knowledge *of the human about the technical system or its environment.*

   Formally, this is the case when SENS(REQ) is not a failure refinement of MMOD, and when a perfect SENS would make no difference.

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

   Formally, this means that SENS(REQ) is a failure refinement of MMOD, but $\mathrm{SENS}_{\mathrm{SAFE}}(\mathrm{REQ}_{\mathrm{SAFE}})$ is not a failure refinement of $\mathrm{MMOD}_{\mathrm{SAFE}}$.
   Since the safety-relevant requirements abstraction function $\mathcal{A}_R$ is correct by definition, the user's mental safety-relevance abstraction function $\mathcal{A}_M$ must be wrong in this case (compare Figure 2 above).

In contrast to previous classifications of mode confusion problems, this classification is *by cause* and not phenomenological, as, e.g., the one by Leveson [10].

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 whether the user can physically observe all safety-relevant environment events, and we must check whether 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. This measure has been recommended by others too, of course, but our rigorous view now indicates more clearly when it must be applied.

Avoiding an incorrect observation also comprises that we check whether psychology ensures that observed safety-relevant environment events become conscious. Our approach points out clearly the necessity of this check. The check itself and any measures belong to the field of psychology, in which we are not expert.

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 conceive 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" [10, 8]. Internal non-determinism may arise not only from the software, but also from the machine's sensor devices. If they are imprecise, the user cannot predict the software input events. We can eliminate both kinds of 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 or the input device's choice, respectively.

Ensuring a correct mental abstraction process is mainly a psychological question and mostly beyond the scope of this paper. 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 conceive training material which helps the user to concentrate on safety-relevant aspects.

7 Summary and Future Work

We present a rigorous way of modelling the user and the machine in a shared-control system. This enables us to propose precise definitions of "mode" and "mode confusion". In our modelling approach, we extend the commonly used distinction between the machine and the user's mental model of it by explicitly separating these and their safety-relevant abstractions. Furthermore, we show that distinguishing three different interfaces during the design phase reduces the potential for mode confusion. Our proposition that the user must not be surprised leads directly to the conclusion that the relationship between the mental model and the machine must be one of specification to implementation, in the mathematical sense of refinement. Mode confusions can occur if and only if this relation is not satisfied. A result of this insight is a new classification of mode confusions by cause, leading to a number of design recommendations for shared-control systems which help to avoid mode confusion problems.

Since tools to model-check refinement relations exist, our approach supports the automated detection of remaining mode confusion problems. For illustration, we presented a case study on a wheelchair robot as a running example. A detailed version of the case study is discussed in [28].

Our work lends itself to extension into several directions. We currently work in our case study on exploiting the new potential for detecting mode confusion problems by model-checking. Furthermore, the recommendations for avoiding mode confusion problems can be tried out. Experts in psychology will be able to implement the non-technical ones of our rules by concrete measures. Finally, we see still more application domains beyond aviation and robotics.

References

[1] Rushby, J.: Modeling the human in human factors. In: Proc. of SAFECOMP 2001. Volume 2187 of LNCS., Springer (2001) 86–91 20, 21

[2] Norman, D.: Some observations on mental models. In Gentner, D., Stevens, A., eds.: Mental Models. Lawrence Erlbaum Associates Inc., Hillsdale, NJ, USA (1983) 20

[3] Crow, J., Javaux, D., Rushby, J.: Models and mechanized methods that integrate human factors into automation design. In Abbott, K., Speyer, J. J., Boy, G., eds.: Proc. of the Int'l Conf. on Human-Computer Interaction in Aeronautics: HCI-Aero 2000, Toulouse, France (2000) 20, 21

[4] Butler, R., Miller, S., Pott, J., Carreño, V.: A formal methods approach to the analysis of mode confusion. In: Proc. of the 17th Digital Avionics Systems Conf., Bellevue, Washington, USA (1998) 20

[5] Rushby, J.: Analyzing cockpit interfaces using formal methods. In Bowman, H., ed.: Proc. of FM-Elsewhere. Volume 43 of Electronic Notes in Theoretical Computer Science., Pisa, Italy, Elsevier (2000) 20, 21

[6] Hourizi, R., Johnson, P.: Beyond mode error: Supporting strategic knowledge structures to enhance cockpit safety. In: Proc. of IHM-HCI 2001, Lille, France, Springer (2001) 20, 21, 27
[7] Sarter, N., Woods, D.: How in the world did we ever get into that mode? Mode error and awareness in supervisory control. Human Factors **37** (1995) 5–19 20, 21
[8] Degani, A., Shafto, M., Kirlik, A.: Modes in human-machine systems: Constructs, representation and classification. Int'l Journal of Aviation Psychology **9** (1999) 125–138 20, 28
[9] Buth, B.: Formal and Semi-Formal Methods for the Analysis of Industrial Control Systems – Habilitation Thesis. Univ. Bremen (2001) 20, 21
[10] Leveson, N., Pinnel, L., Sandys, S., Koga, S., Reese, J.: Analyzing software specifications for mode confusion potential. In: Workshop on Human Error and System Development, Glasgow, UK (1997) 20, 21, 28
[11] Doherty, G.: A Pragmatic Approach to the Formal Specification of Interactive Systems. PhD thesis, University of York, Dept. of Computer Science (1998) 20
[12] Wright, P., Fields, B., Harrison, M.: Deriving human-error tolerance requirements from tasks. In: Proc. of the 1st Int'l Conf. on Requirements Engineering, Colorado, USA, IEEE (1994) 135–142 20
[13] Degani, A., Heymann, M.: Pilot-autopilot interaction: A formal perspective. In Abbott, K., Speyer, J. J., Boy, G., eds.: Proc. of the Int'l Conf. on Human-Computer Interaction in Aeronautics: HCI–Aero 2000, Toulouse, France (2000) 157–168 20
[14] Rodriguez, M., Zimmermann, M., Katahira, M., de Villepin, M., Ingram, B., Leveson, N.: Identifying mode confusion potential in software design. In: Proc. of the Int'l Conf. on Digital Aviation Systems, Philadelphia, PA, USA (2000) 20
[15] Zimmermann, M., Rodriguez, M., Ingram, B., Katahira, M., de Villepin, M., Leveson, N.: Making formal methods practical. In: Proc. of the Int'l Conf. on Digital Aviation Systems, Philadelphia, PA, USA (2000) 20
[16] Rushby, J., Crow, J., Palmer, E.: An automated method to detect potential mode confusions. In: Proc. of the 18th AIAA/IEEE Digital Avionics Systems Conf., St. Louis, Montana, USA (1999) 20, 21
[17] Lüttgen, G., Carreño, V.: Analyzing mode confusion via model checking. In Dams, D., Gerth, R., Leue, S., Massink, M., eds.: SPIN' 99. Volume 1680 of LNCS., Berlin Heidelberg, Springer (1999) 120–135 20
[18] Lankenau, A.: Avoiding mode confusion in service-robots. In Mokhtari, M., ed.: Integration of Assistive Technology in the Information Age, Proc. of the 7th Int'l Conf. on Rehabilitation Robotics, Evry, France, IOS Press (2001) 162–167 20
[19] Palmer, E.: "Oops, it didn't arm." – A case study of two automation surprises. In: Proc. of the 8th Int'l Symp. on Aviation Psychology. (1995) 21
[20] Javaux, D.: Explaining Sarter & Woods' classical results. The cognitive complexity of pilot-autopilot interaction on the Boeing 737-EFIS. In: Proc. of HESSD '98. (1998) 62–77 21
[21] Hourizi, R., Johnson, P.: Unmasking mode errors: A new application of task knowledge principles to the knowledge gaps in cockpit design. In: Proc. of INTERACT 2001 – The 8th IFIP Conf. on Human Computer Interaction, Tokyo, Japan (2001) 21, 27
[22] Röfer, T., Lankenau, A.: Architecture and applications of the Bremen Autonomous Wheelchair. Information Sciences **126** (2000) 1–20 21

[23] Lankenau, A., Röfer, T.: The Bremen Autonomous Wheelchair – a versatile and safe mobility assistant. IEEE Robotics and Automation Magazine, "Reinventing the Wheelchair" **7** (2001) 29–37 21

[24] Parnas, D. L., Madey, J.: Functional documents for computer systems. Science of Computer Programming **25** (1995) 41–61 22

[25] van Schouwen, A. J., Parnas, D. L., Madey, J.: Documentation of requirements for computer systems. In: IEEE Int'l. Symp. on Requirements Engineering – RE'93, San Diego, California, USA, IEEE Comp. Soc. Press (1993) 198–207 22

[26] Hoare, C.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs, New Jersey. USA (1985) 23, 25

[27] Roscoe, A. W.: The Theory and Practice of Concurrency. Prentice-Hall (1997) 23

[28] Lankenau, A.: Bremen Autonomous Wheelchair "Rolland": Self-Localization and Shared-Control – Challenges in Mobile Service Robotics. PhD thesis, Universität Bremen, Dept. of Mathematics and Computer Science (2002) To appear 29