A Case Study for Using a Cognitive Model of Learned Carelessness in Cognitive Engineering

Andreas Lüdtke

Kuratorium OFFIS e.V.
Escherweg 2, 26121 Oldenburg, Germany
luedtke@offis.de

Claus Möbus

Carl von Ossietzky Universität
26111 Oldenburg
moebus@informatik.uni-oldenburg.de

Abstract

The goal of our work is to develop a methodology for automatically predicting potential pilot errors during the design of mode-based systems for modern cockpits. The main ingredient of this methodology is a cognitive model of pilot behaviour allowing to predict misconceptions in the pilot's mental procedure model based on a cognitive process called “learned carelessness”. In a first stage we use this model inside a simulation platform to automatically degrade a normative procedure model in a psychological plausible way. In a second stage we apply formal verification techniques to exhaustively analyse what human errors can be caused by the degraded procedure. In this text we describe a case study conducted at the Lufthansa Flight Training Centre where we analysed a Piper Cheyenne autopilot.

1 Human Error Analysis in Early Stages of Safety Critical System Design

It is often cited that 60-80% of commercial aircraft accidents are caused by the flight crew. What is the reason of this high percentage? The Federal Aviation Administration (FAA) conducted a study to investigate the interaction between flight crews and modern cockpit systems (like flight management and autopilot (AP) systems). One major concern highlighted in the study is a general lack of mode awareness that exists independent from the manufacturer, operator, or airplane type (Lyall & Wilson, 1997). A mode may be understood as a system state in which it delivers a distinguishable function. Lack of mode awareness may lead to mode error where an action is performed that would be correct in some modes but not in the present one. Mode errors lead to "automation surprises", where an operator can no longer follow what the system is doing. During design, the need for modes has to be balanced against the probability of mode errors.

In industry human errors are often considered very late in the system development process when a prototype is available for flight simulator studies with test and line pilots. At this stage changes to improve the usability are very time-consuming and expensive. In the field of cognitive engineering researches develop new approaches in order to allow human error analysis (HEA) already in early system design when only a formal model of the later system exists. Most state of the art approaches to tackle this challenge are based on Norman’s (1988) assumption that operators use mental models to guide their operation of interactive machines. In *design-centred approaches*, formal verification is used on a combination of mental user models and system models to identify potential human errors. The advantage is that a complete analysis can be performed, but the question is, how to generate psychological plausible mental models. In *user-centred approaches*, human simulation is performed to predict user behaviour. The advantage here is that human cognitive processes can be considered explicitly; a disadvantage is that simulation can never be complete. We suggest the integration of both approaches by (I) generating psychologically plausible mental models through simulation of cognitive learning processes, (II) transforming these models into a design notation in order to (III) perform formal verification to automatically compute potential pilot errors. In this way, simulation and verification complement each other and the mental model is the mediating concept.

2 A new Approach Combining Simulation and Formal Verification

We developed a cognitive architecture focusing on an error producing learning process called *learned carelessness*. This psychological theory (Frey & Schulz-Hardt, 1997) states that humans have a tendency to neglect safety precautions if this has immediate advantages, e.g. it saves time. Careless behaviour emerges if safety precautions

have been followed several times but would not have been necessary, because no hazards occurred. Then, people deliberately omit safety precautions because they are considered a waste of time. The absence of hazardous consequences acts as a negative reinforcer of careless behaviour. Learned carelessness is a process which is characteristic for human nature because we have to simplify in order to be capable to perform efficiently in a complex environment. We implicitly degrade our mental model to optimise it for routine situations. Unfortunately this may be disastrous in slightly deviating scenarios. Thus it is crucial to consider this process in system design. To support the analysis we implemented our cognitive model in PROLOG. It is capable to interact with formal system designs that are developed with a commercial case tool (Statemate). The interaction is performed in a human simulation platform, which integrates the cognitive model with Statemate and also with a flight simulation software. An evaluation of the model based on empirical data gathered in flight simulators at the Lufthansa Flight Training Centre has already shown convincing results (Lüdtke & Möbus, 2004).

We differentiate between cognitive model/architecture and procedure models. The human cognitive system may be seen as a "machine" that gets visual/acoustic input and "computes" output in form of verbal utterances and physical actions (like pressing buttons). Our cognitive model describes parts of these processes computationally. In line with established approaches from cognitive science, like ACT-R (Anderson & Lebiere, 1998), we assume that cognition can be explained in terms of information processing. Procedure models describe the knowledge that is applied by the computational processes, e.g. knowledge necessary to perform a flight procedure like takeoff or approach. In essence, the cognitive model we are using may be understood as an interpreter of formal procedure models.

In the analysis we investigate if the interaction between procedure model and system design is susceptible to routine learning processes: Is it likely that a pilot, applying the procedure in day-to-day routine flight, mentally develops a simplified version that may lead to pilot errors in non-routine situations? Examples of such failures are pressing AP buttons when they are not allowed, forgetting an action, or expecting incorrect modes.

3 The Case Study

We performed a case study analysing an autopilot system of a Piper Cheyenne PA42 IIIA. The aim of the case study was twofold: (1) We wanted to test our HEA method and (2) to evaluate the plausibility of the cognitive model. For the second purpose we compared the cognitive model behaviour with behavioural data (including pilot errors) of four pilots collected in flight simulators. We tried to reconstruct the data of one subject using a model-tracing technique. First, the reconstruction was performed without and afterwards, with our cognitive mechanism of learned carelessness (Lüdtke & Möbus 2004). The second run produced a significantly better fit to the data, which shows that our formalisation of learned carelessness explains empirically observed pilot errors. Further results gained by predicting the data of the other three pilots will be shown briefly in this section. The case study was performed in six main steps: (1) System modelling, (2) task analysis, (3) scenario definition, (4) human simulation, (5) formal verification and (6) design change.

1. System modelling: The main target of the analysis is the formal system model. This model must encompass the mode logic (set of modes and mode transitions) and the corresponding control laws. We developed a mode logic model for the Piper Cheyenne autopilot and designed the mode functionality using PID (proportional, integral, differential) controllers. Furthermore we formalised necessary requirements for mode logics (e.g. exactly one mode has to be engaged at every point in time) in LTL-syntax to be able to use the model checker (ModelCertifier) plug-in of Statemate to automatically prove that the mode logic model fulfils the constraints.

2. Task analysis: A second input to the analysis is a model of the flight procedure to be investigated. The procedure defines a normative mental model that is uploaded to our cognitive architecture. Normative here means that a user (model) who operates the system relying on this knowledge would commit no errors. For the case study we modelled flight procedures for altitude change and auto flight engagement. The procedure knowledge is modelled in a GOMS (goals, operators, methods, selection rules)-like fashion (Card, Moran & Newell, 1983):

- *Goals* define a state to be achieved by the pilot. Complex tasks are decomposed into sub-goals that have to be achieved in a prescribed order. The dynamic function of a goal is to provide a control point where the operator can evaluate the progress of the task execution. The task of performing an altitude change with the auto pilot can be decomposed in four sub-goals: "enter desired altitude", "select altitude", "adjust vertical speed", "stabilise indicated airspeed (IAS)".

- *Operators* are the functional entities of the procedure model. We distinguish percept, motor and memory operators. Motor operators represent physical pilot actions like pressing a button or dialling a number into a devise (e.g. alerter), percept operators serve to acquire information about the environment via the displays. Memory operators read variables from short-term memory or store variables in short-term memory. Operators for the altitude change procedure are: "dial desired altitude into the ALERTER", "press ALTS-button", "press ETRIM-button" and "press IAS-button".
- *Methods* describe how a goal may be achieved via a sequence of operators and sub-goals.
- *Selection rules* are used to select between different methods for a goal. The rules associate the before mentioned conceptual task entities with each other. Figure 1 shows the general structure in Backus-Naur-Form. The left-hand side defines the conditional part. Each rule is responsible for exactly one goal ('Goal') or percept ('Percept') and can be chosen if the Boolean expression ('Cond'), defined over system variables (e.g. current mode) or environmental variables (e.g. current altitude), is true. The right-hand side specifies a method, a sequence of percept operators ('Percept'), motor operators ('Motor'), and subgoals ('Goal'). The subgoals may be partially ordered using the keyword 'After'.

'Rule' ⟨ NUMBER ⟩

'Goal' ⟨ GOAL_NAME ⟩ | **'Percept'** '(' ⟨INSTRUMENT_NAME⟩, ⟨ VARIABLE_NAME⟩ ') '
{ **'Memory'** '(' ⟨INSTRUMENT_NAME⟩, ⟨ VARIABLE_NAME⟩ ') ' }
[**'Cond'** ⟨ BOOLEAN_EXPRESSION ⟩]
'⇒' { **'Memory'** '(' ⟨INSTRUMENT_NAME⟩, ⟨ VARIABLE_NAME⟩ ') ' '(' ⟨ LABEL ⟩ ')' }
{ [**'After'** '(' { ⟨ LABEL ⟩ } ')'] **'Percept'** '(' ⟨INSTRUMENT_NAME⟩, ⟨ VARIABLE_NAME⟩ ') ' '(' ⟨ LABEL ⟩ ')' }
{ [**'After'** '(' { ⟨ LABEL ⟩ } ')'] **'Motor'** '(' ⟨ INSTRUMENT_NAME ⟩, ⟨ VARIABLE_NAME ⟩ ')' '(' ⟨ LABEL ⟩ ')' }
{ [**'After'** '(' { ⟨ LABEL ⟩ } ')'] **'Goal'** ⟨ GOAL_NAME ⟩ '(' ⟨ LABEL ⟩ ')' }

Figure 1: Rule format for task analysis

Internally the cognitive architecture uses a goal-state-means format of rules (Lüdtke & Möbus, 2004) to which the selection rules can be translated easily.

3. Scenario definition: As a third input a sequence of scenarios has to be defined. In general, a scenario is defined by an initial state and a list of events (e.g. clearances) and should describe day-to-day normal revenue flights. During human simulation the pilot model flies the procedures under investigation in each scenario and adapts the normative procedure model. Scenarios consist of variables for flight plan data (e.g. frequencies of VORs, initial approach altitude), initial flight conditions (e.g. current altitude, heading, vertical mode) and flight events (e.g. clearances). For the case study we formalised the scenarios that have been flown by the four subjects of the empirical study.

4. Human simulation: For this phase, we implemented a platform that integrates the cognitive architecture and system model with a flight simulator software. The platform realises a closed loop simulation of the pilot-system interaction. Before the first simulation run is started, the procedure model to be investigated (modelled in phase 2.) has to be uploaded to the cognitive architecture and the first scenario (modelled in phase 3.) has to be chosen and uploaded to the simulation platform. After the first run is completed, the learning component adapts all rules applied in this run. Afterwards, the next scenario is uploaded to the platform and the next simulation run is started . In this run the cognitive architecture either applies the original or the adapted rules of the procedure model dependent on rule strength parameters. This iteration of simulation runs and succeeding learning is repeated for each scenario defined in step 3.

The learning mechanism shall be explained in more detail. The learning component models learned carelessness by symbolic learning that is only performed after completion of a successful task, and sub-symbolic learning after each task. The symbolic learning mechanism melts two rules into one rule by means of rule composition (Möbus, Schröder & Thole, 1995). A precondition for composing rules is that firing of the first rule has evoked the second rule, or more exact, the first rule derives a subgoal or percept that is contained in the left-hand side of the second rule. Melting the rules means building a composite rule by combing the left-hand sides of both rules and also combing both right-hand sides. The crucial point is that in this process elements that are contained on the right-hand side of the first and also on the left-hand side of the second rule are eliminated. This process cuts off intermediate knowledge processing steps. Figure 2 shows the composition of Rule 1 and 2 yielding Rule 1&2. The percept operator on the mode annunciation has been eliminated. Instead the composite Rule 1&2 always stores the value

"ALTS" (the value that has been shown during the preceding task performance; Altitude Select mode) directly in memory without looking at the cockpit display. This results in careless behaviour, because the rule assumes that the mode will always be ALTS, which may not be true. Rule 1&2 and Rule 1 are competitive rules, because they refer to the same goal. In order to choose between competitive rules, each rule has a strength parameter that is updated by a subsymbolic learning mechanism after each task performance. Rule 1&2 is applied if its rule strength is higher than that of Rule 1. If this is the case and Rule 1&2 has been applied successfully, it may be further composed with other rules, so that new composite are formed by melting old composites.

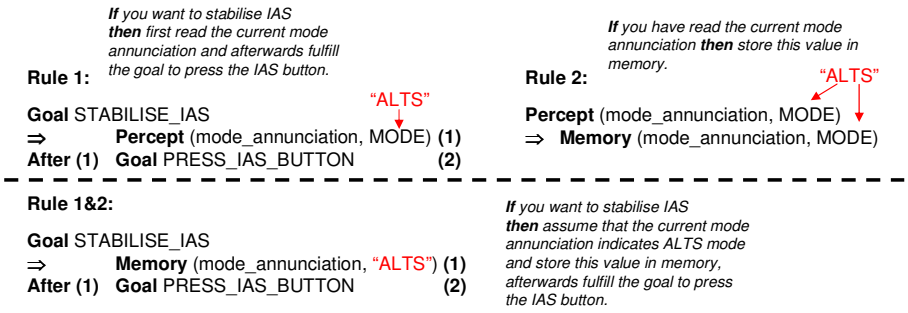

Figure 2: Example for rule composition

The human simulation in the case study was performed for two different procedures (altitude change and auto flight engagement). These trials demonstrated that the implemented functionality of the simulation platform and the pilot model are sufficient enough to simulate the interaction between pilot and system model in varying flight scenarios. The comparison of the model behaviour and real pilot behaviour shows that our model commits errors that comply with errors observed in the empirical study: Seven routine errors of subject A can be successfully reconstructed (Lüdtke & Möbus, 2004) and two routine errors of subject B and D can be correctly predicted. Furthermore, Lüdtke, Möbus & Thole (2002) have shown that the real pilot behaviour exhibits empirical phenomena (speed-up and interleaving) that indicate rule composition. Comparable effects can be evidenced in the model behaviour.

5. Formal verification: After all scenarios have been simulated a snapshot of the now potentially degraded mental model is analysed in order to identify potential human errors due to learned carelessness. The analysis shall answer the question if there is any scenario in which the degraded pilot model commits an error that leads to violation of the overall task goal. Before the analysis can start, the pilot model has to be translated to the design notation (state and activity charts) and tightly integrated with the system model. This is the precondition to be able to apply formal verification techniques. In the case study this translation step was done manually. We are currently working on an automatic translation algorithm.

For formal verification the ModelCertifier verification tool of Statemate is applied. The requirement to be checked was derived from the overall task goal by formalising it in LTL-syntax. The goal of the altitude change task is to reach and maintain a cleared altitude: G(CLEARANCE_DESCEND $\Rightarrow$ FG(CURRENT_ALTITUDE = CLEARED_ALTITUDE)), where G stands for *globally* and FG for *finally globally*. The model checker generates the result *false* and a simulation file showing a counter example. The counter example may be visualised as a waveform using the Statemate Simulation Environment (Figure 3). The waveform shows the evolution of the model variables over time until finally the requirement is violated. In step 5 the event CLEARANCE_DESCEND is generated and CLEARED_ALTITUDE is set to 3500. CURRENT_ALTITUDE is 4000. In step 237 the altitude gets below the cleared altitude and goes on decreasing. The analyst has to interpret the waveform and identify the error that led to this result, taking into account that the normative procedure model does not violate the task goal requirement. In step 101 the integrated model is in a state that is not possible in the normative model: The pilot model presses the IAS button (IAS_BUTTON) though the ALTS mode is active (ALTS_MODE.ACTIVE). Assuming the correctness of the normative model the violation may be attributed to the differences between the normative and degraded model: The degraded model contains a rule where the IAS button is pressed without taking the current flight mode into account. Formal verification has shown that there is at least one scenario where this carelessness leads to violation of the task goal.

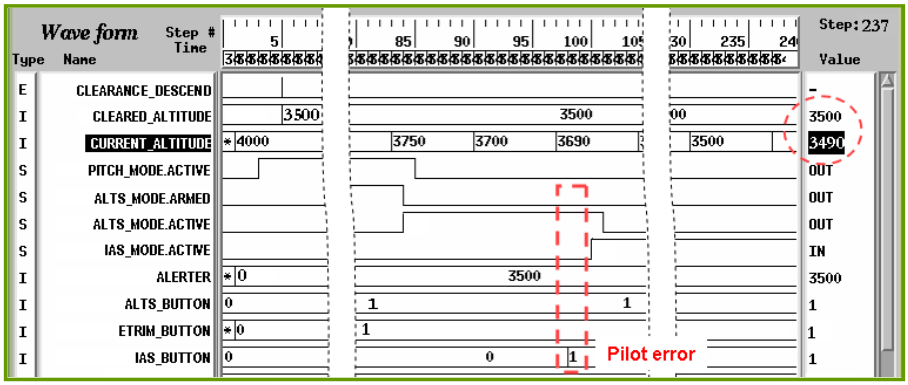

Figure 3: Waveform showing a potential pilot error due to learned carelessness

6. Design changes: Finally the system design should be changed to prevent the identified human errors. In this way the design is adapted to the needs of the human operator. Thus we consider the presented method a step forward in realising a human centred design methodology. The Piper Cheyenne AP mode logic was modified so that pressing the IAS button in ALTS mode has no effect. Afterwards another model checking run was performed to investigate if the changes are effective. This proved that the identified error is no longer possible in the model. But another pilot error was identified. In order to prevent all human errors that are possible due to the degraded pilot model, model checking has to be repeated iteratively until no more errors can be found.

4 Way forward

The work presented in this text is currently continued in the context of the European project ISAAC (Improvement of Safety Activities on Aeronautical Complex Systems) in cooperation with Airbus France, ALENIA Aeronautica and SAAB. ISAAC is financed in the 6th EU framework programme in the thematic area Aeronautics and Space (for further information see http://www.cert.fr/isaac/).

5 References

Anderson, J. R. & Lebiere, C. (1998). *The Atomic Components of Thought*. Hillsdale, NJ: Erlbaum.

Card, S. K., Moran, T. P. & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.

Frey, D. & Schulz-Hardt, S. (1997). Eine Theorie der gelernten Sorglosigkeit. In H. Mandl (Ed.), *Bericht über den 40. Kongress der Deutschen Gesellschaft für Psychologie* (pp. 604-611). Göttingen, Bern, Toronto, Seattle: Hogrefe Verlag für Psychologie.

Lüdtke, A. & Möbus, C. (2004). A Cognitive Pilot Model to Predict Learned Carelessness for System Design. In A. Pritchett and A. Jackson (Eds), Proceedings of the *International Conference on Human-Computer Interaction in Aeronautics* (HCI-Aero), 29.09.-01.10.2004, Toulouse, France.

Lüdtke, A. & Möbus, C. & Thole, H.J. (2002). Cognitive Modelling Approach to Diagnose Over-Simplification in Simulation-Based Training. In St. A. Cerri, G. Gouarderes & F. Paraguacu (Eds), *Intelligent Tutoring Systems, Proceedings of the 6th International Conference* (pp. 496-506). Berlin: Springer.

Lyall, B. & Wilson, J. (1997). *Flight Deck Automation Issues*. Oregon State University & Research Integrations, Incorporated. Retrieved May 5, 2004, from http://www.flightdeckautomation.com/fdai.aspx.

Möbus, C., Schröder, O. & Thole, H.J. (1995). Online Modeling the Novice-Expert Sift in Programming Skills on a Rule-Schema-Case Partial Order. In K.F. Wender, F. Schmalhofer, H.-D. Böcker (Eds), *Cognition and Computer Programming*. Norwood, N.J.: Ablex

Norman, D. A. (1988). *The psychology of everyday things*. New York: Basic Books.