Cognitive Modelling Approach to Diagnose Over-Simplification in Simulation-Based Training

Andreas Lüdtke, Claus Möbus, and Heinz-Jürgen Thole

University of Oldenburg, Department of Computer Science, Germany,
{luedtke, moebus, thole}@informatik.uni-oldenburg.de,
WWW home page: http://lls.informatik.uni-oldenburg.de

Abstract. Simulation-based training has become a standard in operational knowledge training for supervisory control in safety-critical environments. But traditional simulators do not support mental model formation of automated systems though these systems are a dominant part in modern control systems. Recently various "intelligent components" for this support have been suggested. But these approaches neglect the dynamic character of mental models. They focus on building a normative model at the beginning of the training but do not consider how it evolves due to knowledge acquisition processes. In this paper we present a model-based approach to diagnose success-driven learning in simulator training and to predict dangerous over-simplifications. Our research focuses on pilot training for automated cockpits.

1 Introduction

New developments in computer technology made high fidelity simulators with a very close resemblance to reality possible. Simulator training allows students to develop operational skills without harming their own life, that of others or real world equipment. But simulation-based learning has to be accompanied by good support and feedback on the learners' behaviour [7]. Recently several researchers emphasized the need for intelligent technologies that diagnose the student's performance and are able to assist the learner [1, 8] and to adapt simulated scenarios accordingly [18]. As Connolly et al. [1] state it, "the focus of development efforts should be on incorporating intelligent technologies into these simulators rather than emphasizing the relative fidelity of the systems" (p. 535). In modern highly automated cockpits the need for "intelligent components" that support the construction of adequate mental models of the automatic systems has been pointed out [10, 15]. Most tasks have been automated and the pilot's role can be understood as supervisory control [17]. He has to program the automatic systems, to observe their behaviour and to intervene when an error occurs. This role imposes increased demand on the mental capabilities. A crucial question is whether pilots can cope with them and are able to build an adequate mental model of the systems behaviour at the beginning of the training and will maintain it when gaining experience. Humans develop routine, which makes their performance faster. But on the other hand routine may also lead to over-simplifications (OS) [4] that appear to work well in normal situations but lead to error in exceptional cases. This may result in catastrophic human errors as a number of accidents show. Thus it is not enough to support mental model formation at the beginning of the training. Moreover we have to take into account knowledge acquisition processes that take part when the pilot applies his normative model. In this paper an approach based on a two-layered pilot model is presented. The first layer is a normative mental model of auto pilot (AP) behaviour and the second a success-driven knowledge acquisition process with access to the first layer. We intent to connect it to a flight simulator in order to diagnose OS based on the performance trace of the pilot student. In Ohlsson's categorisation [14] our pilot model is a simulation model, because it is able to construct the student's problem solving path step by step. Other approaches to pilot modeling (e.g. MIDAS [2], Javaux's and Oliver's approach [6]) differ from our model because they do not consider knowledge acquisition processes.

The cognitive approach relies on the empirically proofed ISP-DL (Impasse-Success-Problem Solving-Driven Learning) theory of knowledge acquisition [11, 13]. In our working group this psychological theory guided the development of IPSEs (Intelligent Problem Solving Environments) in various domains, for instance functional programming in the ABSYNT project. We have adapted the framework according to pilot domain requirements and used it to formalise normative operational rules for an AP system. The contents of the rules have been extracted from type rating documents and interviews. According to the ISP-DL theory experienced pilots will need less planning than novices, because they act according to stored and well-tried but still dynamically changing schemata [12]. Schemata are constructed through success-driven learning, optimised for routine situations but lacking specific features for exceptional situations. Thus they may be used to model OS.

After an introduction we present a brief explanation of a typical pilot error. The main part of this paper contains the detailed description of the modeling framework followed by an empirical case study.

2 Explanation of an Exemplary Pilot Error

An empirical study in a full-motion Piper Cheyenne flight simulator was conducted at Lufthansa Flight Training in order to identify and explain systematic pilot errors concerning the AP operation. The following error of subject A (fig. 1) happened in a step-climb manoeuvre and according to flight instructors could happen to almost every pilot:

After start the aircraft was cleared from air traffic control for 4000 ft. The pilot dialled this altitude in the 'Alerter' and selected it by pressing the ALTS-Button, which causes the AP to go into 'Altitude Select' mode. Then he pressed the ETRIM-Button to increase vertical speed (VS). Suggested VS for a climb is 2000 ft/min but it has to be adjusted to keep the indicated airspeed (IAS) above the 160 knts limit - increased VS causes IAS to decrease. The adjustment should be delegated to the AP by activating the IAS mode via the IAS-Button. Subject A waited for IAS to reach 160 knts in order to activate IAS mode in the correct moment. Approximately 300 ft before the selected altitude (Lead Point) the AP automatically transitions to 'Capture' mode and decreases VS so that the aircraft smoothly levels off. When finally 160 knts were reached and subject A pressed the IAS-Button the automatic transition to 'Capture' mode had already occured. Surprisingly for the pilot the aircraft did not level off at 4000 ft but continued to climb. Stabilising IAS is not allowed in 'Capture' mode. The problem was that transition to 'Capture' mode occurred exceptionally early and was not noticed, because the pilot had not expected it.

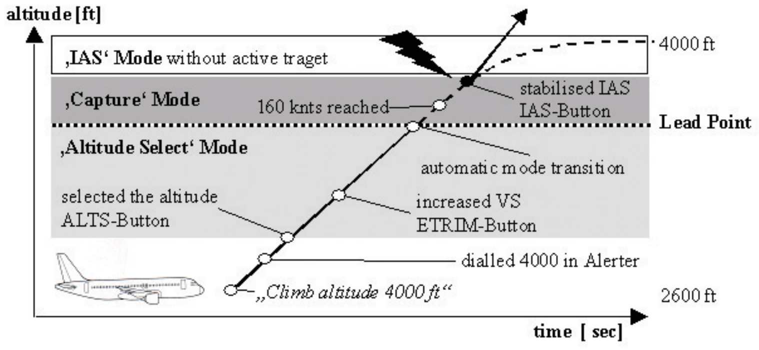

Fig. 1. Trajectory of the exemplary pilot error

To explain this error we constructed and analysed a formal model of the AP mode logic considering the relevant modes. Our analysis was guided by the work of Degani [3] and Leveson et al. [9]. Both suggested structural design features that may be a source for human error. We found a number of instantiations of these in the AP mode logic. For instance "inconsistent behaviour" in connection with the IAS-Button (fig. 2): When it is pressed in 'Altitude Select' mode, IAS mode becomes active, and the selected altitude is kept as an active target. But when pressed in 'Capture' mode though again IAS mode becomes active the selected altitude is no longer an active target - the manoeuvre is cancelled.

From the psychological perspective the "branching error" concept of Reason [16] applies to our scenario, because the initial sequence for both cases (160 knts reached in 'Altitude Select' or in 'Capture' mode) is the same (fig. 2). Then the pilot gets to a "critical decision point" before pressing the IAS-Button. He has to look carefully for the automatic mode transition. Formal interviews revealed that principally pilots know about that decision point, which leads to the assumption that initially both paths are represented in the mental model. But during training pressing the IAS-Button becomes a habit (bold arrows in fig. 2) because most times the transition occurs late enough. In accordance to the "branching error" concept we fade out alternative actions in our present pilot model based merely on the frequency of their successful application in the past.

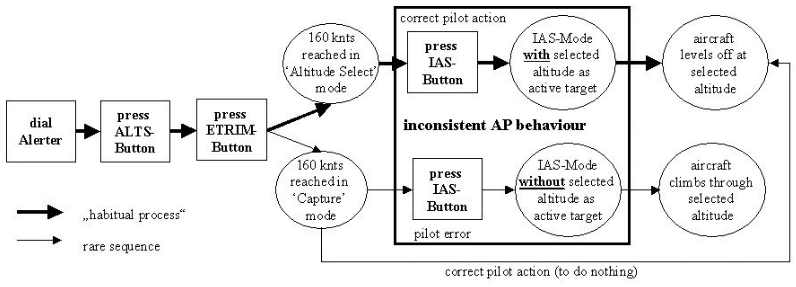
dial Alerter
press ALTS-Button
press ETRIM-Button
160 knts reached in 'Altitude Select' mode
160 knts reached in 'Capture' mode
correct pilot action
press IAS-Button
IAS-Mode with selected altitude as active target
inconsistent AP behaviour
press IAS-Button
IAS-Mode without selected altitude as active target
pilot error
aircraft levels off at selected altitude
aircraft climbs through selected altitude
„habitual process"
rare sequence
correct pilot action (to do nothing)

Fig. 2. Inconsistent AP behaviour after IAS-Button

3 Modelling Framework

To build the pilot model we take advantage of the goals-means-relation (GMR) modeling framework that was originally used in the ABSYNT project [11, 12] to simulate knowledge acquisition processes. ABSYNT is a functional visual programming language, indeed a tree representation of pure LISP. Using a problem-solving monitor (PSM) a learner acquires basic functional programming concepts while working on programming tasks. The learner starts with basic pieces of knowledge acquired in theoretical lessons. If he succeeds in solving a familiar task, the knowledge he used is optimised by eliminating planning steps. Next time when working on a similar problem he will perform faster because of a shorter problem-solving path. In ABSYNT this process of success-driven learning is modelled by composition of simple GMR rules, which yields schemata and moreover whole solution cases. In the pilot context GMRs are used to represent the pilot's knowledge about how to operate the AP - the operational rules. Rule composition serves to model the development of "habitual processes" and OS in this context. In order to use the ABSYNT modeling framework in this way the two domains have been compared carefully in detail. The text that follows provides a summary of this comparison and a brief description of the knowledge representation for the pilot domain.

The main difference is that the pilot acts in a dynamic environment whereas the ABSYNT domain is static. The state of the aircraft constantly changes caused not only by the pilot but also by external influences. The modelling framework has to take this into account. First the various problem-solving activities have been compared. The ISP-DL theory states that the problem-solving process can be decomposed into phases very alike to the Rubicon Theory [5]:

- In the deliberation phase the problem solver decides to strive for the goal to solve a chosen or given task. Task goals in ABSYNT are specifyed by a predicative requirement that has to be fulfilled by a program yet to be created. In the pilot domain task goals are predicatively specifyed by flight parameters (e.g. altitude) to be achieved by a manoeuvre yet to be performed.
- In the planning phase the problem solver elaborates goals into implementable sub goals. In ABSYNT the result is a plan containing a set of language primitives the learner intends to arrange in the PSM. These actions only depend on goals and previously executed actions. Because of the dynamic environment the pilot must constantly check the current situation. He derives a plan containing a set of intended checks on the displays and movements of control instruments(CI).
- In the execution phase plans are executed. ABSYNT actions refer to mouse movements to arrange graphical icons in the PSM. In the pilot domain there are move-actions refering to movements of CI (e.g. pressing buttons) and check-actions refering to checks on displays to get information about the AP and the aircraft.
- In the evaluation phase the result of the actions is evaluated according to the task goal. In ABSYNT only the final program is evaluated, not those actions that have been "undone". In the pilot domain every single action is interpreted instantly by the AP and influences the goal achievement. Most times incorrect pilot actions can be alleviated by counteractions, but this increases pilots' workload and extends the manoeuvre time. So, the whole action trace has to be evaluated and not only the final result.

Next we present the comparison with regard to knowledge representation:

Concept	**General Description**	**Characteristics in ABSYNT**	**Characteristics in the pilot domain**
Goals	are derived and detailed during planning	represented by predicates describing a goal with associated sub goals	the predicates additionally describe instruments to be checked
Means	set of operator and leaf nodes	operator nodes represent programming language primitives; leaf nodes stand for parameters and constants	operator nodes represent move- and check-actions; leaf nodes represent state expressions that should be true after execution
GMR	associates a goal with a means	associates programming goals with LISP primitives	associates manoeuvre goals with pilot actions (checks and moves)
GMR rules	GMR is implemented in a set of rules in a Horn clause format. The GMR on the left hand side (Head) of a rule holds, if all GMRs on the right (Body) are true. There are four types of rules: task rules, goal elaboration rules, operator rules, and leaf rules		

Task rules	convert a task goal into a planning goal and impose constraints for the means	the task is to construct a program that fulfills the task goal	the task is to combine actions in an adequate chronological order and dependency that fulfills the task goal
Goal elaboration rules	differentiate a goal but the implementation is postponed	reification of goals made goal elaboration "executable" in the PSM	reification of goals is not desirable because of workload considerations
Operator rules	describe how goals can be achieved by operator nodes	describe the implementation of programming goals	describe the realisation of manoeuvre goals and constrain the execution of move-actions on the result of check-actions
Leaf rules	special type of primitive operator rules	describe the implementation of parameters and constants by leaf nodes	describe the result of pilot actions by pilot leaf nodes

The main differences in knowledge representation are the list of instruments I contained in the goal representation and the semantics of operators. First, we have chosen to extend the representation of goals, because the instruments pilots have to look for always depends on the present goal respectively manoeuvre. For example in a 'free climb' (without selected altitude) there is no 'Capture' mode, thus there is no need to look for it before pressing the IAS-button. By comparing I with the instruments considered in the associated check-actions sources for OS can be identified. Second, the semantics of operators differs in the control flow. In the pilot domain move-actions are only executed if preceding check-actions deliver that necessary preconditions are fulfilled. If not the move-actions are canceled. In ABSYNT there is no such dependency.

Finally we adapt the learning component. In ABSYNT rule composition was used to model changes in the learner's knowledge after he has successfully solved similar tasks (success-driven learning). Rules are composed if they have been used in succession few times and if certain constraints are fulfilled (see performance hypotheses below). Technically this is done according to the cut rule:

$$\begin{array}{ll} \underline{\text{F} \longleftarrow \text{P \& C}, \qquad \text{P'} \longleftarrow \text{A}} & \text{P, P', F must be atoms (here: predicates)} \\ \quad (\text{F} \longleftarrow \text{A \& C})\ \sigma & \text{A and C can be conjunctions of atoms} \end{array}$$

The two clauses above the line resolve to the clause below the line by merging them and thereby eliminating P and P'. P and P' must have at least one identical instantiation. σ stands for the most general unifier and is applied to the whole resolvent. As an example we consider the following two GMR rules written here as Horn clauses:

```
gmr(check_act_ias(instr(speed-ind,mode-annunciation,ias-button)),Act)) ⟵
   gmr(check1_ias(instr(speed-ind)), Check1)) &
   gmr(check2_ias(instr(mode-annunciation)), Check2)) &
   gmr(act_ias(instr(ias-button)),Act)).

 gmr(check2_ias(instr(mode-annunciation)), unequal(value(MA),capture)) ⟵
   value(MA), MA ≠ capture.
```

The second GMR in the body of the first rule and the head of the second one can be unified if σ =[Check2 / unequal(value(MA),capture)]. Applying the cut rule yields the composite:

```
 gmr(check_act_ias(instr(speed-ind,mode-annunciation,ias-button)),Act)) ⟵
   gmr(check1_ias(instr(speed-ind)), Check)) &
   (value(MA), MA ≠ capture) & gmr(act_ias(instr(ias-button)),Act)).
```

By composing rules "habitual processes" are modelled, but there is a major difference in rule composition between ABSYNT and the pilot domain. In ABSYNT composition always generates correct rules. Whereas in the pilot domain we constructed the rules in way so that composition can lead to the elimination of check-actions. In the example the check for the capture phase is replaced by the assumption that this phase is not active. This technique serves to model OS. In ABSYNT four hypotheses about performance differences between novices and experts were derived, where gaining expertise means continuously building more condensed schemata until whole solution cases emerge. At present we apply two of these hypotheses to show empirical indications for rule composition:

Time hypothesis. The selection and execution of each rule takes time, so a pilot using composites should be faster than someone using simple rules.

No-interleaving hypothesis. Actions contained in the same means are processed without interruption by other means. Thus actions in different means should not interleave. This enables to predict a partial order on action steps in different GMR rules. For actions in composites this is not possible.

In the pilot domain these hypotheses serve to diagnose based on the performance trace what rules have been composed and over-simplified.

4 Applicability of Rule Composition - an Empirical Case Study

During the case study 4 flight students have been taped on video. Subject A was filmed in 7 missions (7 * 1.5h). The results of a protocol analysis for subject A in step-climb and step-descent manoeuvres are described. Each manoeuvre protocol resulted from a transcription of a video sequence that was stopped and transcribed every 2 seconds. Manoeuvre time varied from 1 to 4 min.

First we investigated the time hypothesis. We analysed the time the subject needed for the first three manoeuvre actions: Dialling in the altitude, selecting it, and pressing the ETRIM-Button the first time (fig. 3 up to manoeuvre 10). The ETRIM-Button has to be pressed a few times until the desired VS is reached, but this time span heavily depends on environmental factors like wind and air pressure. Thus we considered only the initial stroke. When the times for manoeuvre 3, 4, 8, and 10 are considered as outliers we can draw a line connecting the remaining values. We interpret that the required time gets constantly shorter. The outliers can be explained by tasks that had to be performed in parallel to the climb or descent and by the initial VS before the manoeuvre. E.g. before the fourth manoeuvre VS was already 1500 ft/min, so there was no need to adjust it immediately and he could activate climb power (a parallel task) first.

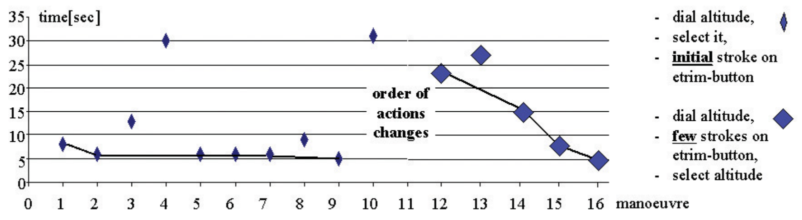

Fig. 3. Performance times for manoeuvres 1 to 11 and 12 to 16

One weak explanation for the faster performance is that the subject gets used to the location of the buttons. But according to our explanation the subject eliminates sub goals and thus needs less planning time. We model this speeding up by rule composition, which is plausible because the manoeuvres could be executed using the same set of rules.

Next we investigated the no-interleaving hypothesis. We assume that at the beginning of the training pilots have one simple rule for every single action, because there are individual constraints for each action. Thus according to the no-interleaving hypothesis for example altitude selection and decreasing VS should not interleave. But in the twelfth manoeuvre subject A starts to interleave exactly these two actions: He hits the ETRIM-Button a few times, then selects the altitude and continues with the ETRIM-Button until the prescribed VS is reached. This interleaving could be observed in all following manoeuvres. According to our model this interleaving is only possible if the subject has built a composite containing the actions for altitude selection and VS adjustment in the same means. He is no longer using the simple rules.

With this new rule set we investigated the time hypothesis again starting from the twelfth manoeuvre. We considered the same three actions as before, but this time in another order. We had to incorporate not only the initial stroke on the ETRIM-Button but all strokes he made before selecting altitude. This time span still leaves out the fine tuning of VS which heavily depends on environmental factors. As can be seen in fig. 3 (from manoeuvre 12 to 16) the time decreases with only one outlier. Because again all manoeuvres could be executed using the same set of rules (including the composite) and according to the time hypothesis our explanation is that further planning steps have been eliminated. In the 16th manoeuvre the subject commits the error described in section 2. This is the first time that the transition to 'Capture'-mode occurs before the 160 knts limit (the desired value for IAS stabilisation) is reached. The active mode is not noticed by the pilot and he stabilises IAS, which causes the plane to overshoot the cleared altitude. Thus we suppose that the necessary check has been eliminated during the rule composition and the resulting composite is over-simplified and applied under inappropriate conditions.

5 Conclusion and Further Work

We described a typical pilot error that was observed regularly in an empirical study we conducted at Lufthansa Flight Training during simulation-based pilot training. Based on theoretical work on system structure and human factors we found a hypothetical explanation of this error. The GMR modelling framework was adapted to the pilot context and serves the purpose of modelling a success-driven knowledge acquisition process by rule composition. This process leads to optimised knowledge ignoring necessary checks on display instruments, which can be understood as OS. Based on empirical data we have shown first indications for the applicability of our cognitive modeling approach. The model predictions will be used to improve learning environments for pilots. We envision integrating it with a flight simulator in order to diagnose OS of pilot students based on their performance trace. Identified OS shall be indicated to the flight instructor, how has to decide what to do. In further work the pilots' attentiveness to crucial automatic mode transitions of the AP has to be analysed thoroughly. Because in the empirical case study we observed, that the described error does not occur in situations with a high level of attentiveness. After the model is fully implemented and attentiveness is integrated it shall be evaluated as a whole.

6 Acknowledgements

We would like to thank the flight instructors and students at the Lufthansa Flight Training, especially the flight instructors Walter Dodel, Manfred Klapper and Christoph Ferenz. Many thanks also to Chief Flight Instructor and Head of Training Harald Neumann who made the empirical study possible.

References

1. Conolly, C. A, 1Lt Johnson, J., MSgt Lexa, C.: AVATAR: An Intelligent Air Traffic Control Simulator and Trainer. In In B.P. Goettl et al. (eds.), Proceedings of the Fourth International Conference, ITS '98. Berlin [u.a.] : Springer (1998).

2. Corker, K.M.: Cognitive Models and Control: Human and System Dynamics in Advanced Airspace Operations. In N.B. Sarter, R. Amalberti (eds.), Cognitive Engineering in the Aviation Domain. Mahwah, NJ : Lawrence Erlbaum Ass. (2000).
3. Degani, A.: Modeling Human-Machine Systems: On Modes, Error, and Patterns of Interaction. Ph.D. Thesis. Georgia Institute of Technology (1996).
4. Feltovich, P.J., Spiro, R.J., Coulson, R.: The nature of conceptual understanding in biomedicine: The deep structure of complex ideas and the development of misconceptions. In D. Evans, V. Patel (eds.), Cognitive science in medicine: Biomedical modelling. Cambridge, MA : MIT Press (1989).
5. Gollwitzer, P.M.: Action phases and mind sets. In E.T. Higgins, R.M. Sorrentino (eds.), Handbook of motivation and cognition: Foundations of social behaviour, Vol 2. New York : Guilford Press (1990) 53-92.
6. Javuax, D., Olivier, E.: Assessing and understanding pilots' knowledge of mode transitions on the A340-200/300. In Proceedings of the International Conference on Human-Computer Interaction in Aeronautics, HCI-Aero'00 (2000).
7. de Jong, T.: Learning and Instruction with Computer Simulations. In Education & Computing, 6, (1991) 217-229.
8. Khan, T.M., Paul, S.J., Brown, K.E., Leitch, R.R.: Model-Based Explanations in Simulation-Based Training. In B.P. Goettl et al. (eds.), Proceedings of the Fourth International Conference, ITS '98. Berlin [u.a.] : Springer (1998).
9. Leveson, N.G., Pinnell, L.D., Sandys, S.D., Koga, S., Reese, J.D.: Analysing Software Specifications for Mode Confusion Potential. In C.W. Johnson (eds.), Proceedings of the Workshop on Human Error and System Development, Technical Report GAAG-TR-97-2. Glasgow Accident Analysis Group (1997).
10. Mitchell, C.M.: Horizons in Pilot Training: Desktop Tutoring Systems. In N.B. Sarter and R. Amalberti (Eds.), Cognitive Engineering in the Aviation Domain. Mahwah, NJ: Lawrence Erlbaum Associates (2000).
11. Möbus,C., Schröder, O., Thole,H.-J.: Diagnosing and Evaluating the Acquisition Process of Programming Schemata. In J.E. GREER, G. McCALLA (eds.), Student Modelling: The Key to Individualized Instruction. Berlin: Springer (1994).
12. Möbus,C., Schröder, O., Thole,H.-J.: Online Modelling the Novice-Expert Shift in Programming Skills on a Rule-Schema-Case Partial Order, in F. Schmalhofer, K.F. Wender, H. Bcker (eds.), Cognition and Computer Programming, Ablex Series in Computational Sciences, Norwood, N.J.: Ablex (1995).
13. Möbus,C., Thole,H.-J., Schröder, O.: Interactive Support of Planning in a Functional, Visual Programming Language. In P. Brna et al. (eds.), Proceedings AI-ED '93, World Conference on Artificial Intelligence and Education (1993) 362-369.
14. Ohlsson, S.: Some principles of intelligent tutoring. In Instructional Science 14. Amsterdam: Elsevier Science Publishers B.V (1986) 293-326.
15. Paries, J., Amalberti, R.: Aviation Safety Paradigms and Training Implications. In N.B. Sarter, R. Amalberti (eds.), Cognitive Engineering in the Aviation Domain. Mahwah, NJ: Lawrence Erlbaum Associates (2000).
16. Reason, J.: Action not as planned. In G. Underwood, R. Stevens (eds.), Aspects of Consciousness. London: Academic Press (1979).
17. Sheridan, T.B.: Supervisory Control. In Handbook of Human Factors and Ergonomics. New York [u.a.] : Wiley (1997).
18. Zhang, D. M., Alem, L.: Using Case-Based Reasoning for Exercise Design in Simulation-Based Training. In C. Frasson, G. Gauthier, A. Lesgold (eds.), Proceedings of the Third International Conference, ITS'96. Berlin [u.a.] : Springer (1996).