Running Head: SKILL ACQUISITION IN AIR TRAFFIC CONTROL

A Model of Individual Differences in Skill Acquisition in the Kanfer-Ackerman Air Traffic Control Task

Niels A. Taatgen

University of Groningen

Abstract

Individual differences in skill acquisition are influenced by several architectural factors. According to Ackerman’s theory, general intelligence, speed of proceduralization and psychomotor speed influence different stages of skill acquisition. The ACT-R cognitive architecture allows for direct testing of this theory by manipulating parameters that correspond to these factors. The present study discusses an ACT-R model of the Kanfer-Ackerman Air Traffic Control task in which the relevant abilities can be manipulated directly. The model predictions show the same patterns of correlations as the patterns found by Ackerman in the experimental data.

Keywords: Skill acquisition, Air Traffic Control, Individual differences, ACT-R

A Model of Individual Differences in Learning the Kanfer-Ackerman Air Traffic Control Task

Skill acquisition is usually characterized as going through three stages: a cognitive stage, an associative stage and an autonomous stage (Fitts, 1964). The three stages can be characterized by moving from conscious, slow and error-prone to unconscious, fast and error-free. Anderson (1982) explains these three stages in terms of a transition from declarative knowledge to procedural knowledge. In the cognitive stage knowledge is declarative and needs to be interpreted. Interpreting knowledge is slow, and may lead to errors if the relevant knowledge cannot be retrieved at the right time. Procedural knowledge on the other hand is compiled and therefore fast and free of errors, and can be associated with the autonomous stage. The associate stage is an in-between stage, during which part of the knowledge is declarative and another part compiled.

A problem in the study of complex problem solving, especially in a learning context, is the vastness of individual differences. In order to study the acquisition of complex skills, it is a good research strategy to have a theory of individual differences. From the perspective of the cognitive architecture, there are two sources of individual differences: <u>architectural differences</u> and <u>knowledge differences</u> (Taatgen, 1999a). Architectural differences are differences in the cognitive architecture itself. In terms of an architecture like ACT-R, architectural differences can be tied to global parameters. For example, working-memory capacity is tied to the <u>W</u>-parameter in ACT-R, the parameter that controls the amount of spreading activation. Individual differences in working-memory capacity can be explained by estimating a different value of <u>W</u> for each individual (Lovett, Reder & Lebiere, 1997). Differences in knowledge are based on the idea that people have different problem solving strategies. In terms of a cognitive model, this means individualized models have different initial contents of declarative and procedural memory.

In this paper I will focus on architectural differences. Ackerman (1988, 1990) identified three sources: general intelligence, perceptual speed, and psychomotor abilities. According to Ackerman, each of these three abilities correlates with a different stage of skill acquisition. In the cognitive stage, general intelligence is the most important aspect, as an adequate representation of the task needs to be formed. In the associative stage, the knowledge compilation process (which Ackerman associates with perceptual speed) will dominate performance, so individual differences in that aspect will become important. In the final autonomous stage, all knowledge is proceduralized, and differences in psychomotor abilities will be the most important factor. Figure 1 illustrates the general predictions of the theory.

Ackerman (1998; 1990) gathered evidence for this theory by correlating learning behavior on a complex task (the Kanfer-Ackerman Air Traffic Controller task[1], KA-ATC) with performance on simpler tasks that explicitly test the three abilities Ackerman thought to be relevant in the three stages of skill acquisition. It turned out that measures of general intelligence correlate well with the first blocks of ATC performance, measures of perceptual speed with the middle blocks, and measures of psychomotor abilities with the later blocks.

Cognitive modeling offers a different approach to finding support for Ackerman's

1. Kanfer-Ackerman Air Traffic Controller Task© program is copyrighted software by Ruth Kanfer, Philip L. Ackerman, and Kim A. Pearson, University of Minnesota.

theory. Instead of correlating performances on different tasks, a model can be made of the complex task, and architectural parameters can be varied that correspond to the relevant dimensions of individual differences. This is the approach we will examine in this paper.

A Model of the ATC Task

The ATC Task

Although the ATC task is a simplified version of real Air Traffic Control, it is still a complicated task. Figure 2 shows the interface of the task. The goal is to score as many points as possible by landing planes and making no errors. The planes that have to be landed are represented at the top-left part of the screen, and are organized in three hold levels (indicated in the POS. column). Planes can be moved between hold levels, and can be landed from hold level 1 (the bottom four slots). There are four runways in the bottom-left of the screen on which planes can be landed. The choice of runway is constrained by a number of rules concerning runway length (long or short), plane type (prop, 727, dc10 or 747), runway direction (north-south or east-west), runway condition (dry, wet or icy), wind direction (north, south, east or west) and wind speed (0-20, 25-35 or 40-50 knots). The main rules of interest in the context of the model are the rules about whether a plane may land on the short runway (planes may always land on the long runway):

747's may never land on the short runway
727's may land on the short runway when the runway is dry or the wind speed is 0-20 knots
DC10's may land on the short runway when the runway is not icy and the wind is not 40-50 knots.
Prop's may always land on the short runway

Once a plane has successfully been assigned to a runway, it occupies the runway for some time. The runway has to be clear again before other planes may be assigned to it. Planes have a limited amount of fuel: the fuel column indicates the number of minutes the plane has left. When a plane runs out of fuel, it crashes. Except for the planes in the three hold levels, there is a queue of waiting planes. A waiting plane can be entered into an empty slot.

The interface is operated by the keyboard, mainly by using the up and down keys to move the arrow in the display up and down, and the return key to select planes and runways. Subjects receive 50 points for successfully landing a plane, 10 penalty points for violating a rule (the interface gives feedback on these violations), and 100 penalty points for each plane that crashes. Trials take 10 minutes each, after which the total amount of points is calculated.

An overview of the ACT-R architecture

The model presented here is based on the ACT-R 4.0 cognitive architecture. The theoretical foundation of the ACT-R architecture is rational analysis of human cognition (Anderson, 1990). According to rational analysis, each component of the cognitive system is optimized with respect to the demands from the environment given its computational limitations. The main components in ACT-R are a declarative (fact) memory and a production (rule) memory. ACT-R is a hybrid architecture in that it has both symbolic and sub-symbolic aspects. I describe these components informally. Further details about the ACT-R architecture can be found in Anderson and Lebiere (1998).

Items in declarative memory, called chunks, have different levels of activation to reflect their use: chunks that have been used recently or chunks that are used very often receive a high activation. This activation decays over time if the chunk is not used. In addition, chunks cannot act by themselves; they need production rules for their application. In order to use a chunk, a production rule has to be invoked to retrieve it from declarative memory and another rule to do something with it. Since ACT-R is a goal-driven theory, chunks are always retrieved to achieve some goal. In the context of the KA-ATC task there are several goals. One of the goals may be to land a plane for which it may be necessary to hand over the control to a lower-level goal, e.g., a goal to move the arrow on the screen to the desired plane.

The behavior of production rules is also governed by the principle of rational analysis. Each production rule has a real-valued quantity associated with its expected outcome. Expected outcome is calculated from estimates of the cost and probability of reaching the goal if that production rule is chosen. The unit of cost in ACT-R is time. ACT-R's learning mechanisms constantly update these estimates based on experience. If multiple production rules are applicable for a certain goal, the production rule is selected with the highest expected outcome.

In both declarative and procedural memory, selections are made on the basis of some evaluation, either activation or expected outcome. This selection process is noisy, so the item with the highest value has the greatest probability of being selected but other items get opportunities as well. This may produce errors or suboptimal behavior but also allows the system to explore knowledge and strategies that are still evolving. In addition to the learning mechanisms that update activation and expected outcome, ACT-R can also learn new chunks and production rules. New chunks are learned automatically: each time a goal is completed it is added to declarative memory. If an identical chunk is already present in memory, both chunks are merged and their activation values are combined. Chunks acquired through perception, information on the screen for example, are also stored. New production rules are learned on the basis of specializing and merging existing production rules. Since this process is quite crucial in our model we will examine it in more detail later on.

The Model

The ATC task is a complicated task, modeling all aspects is a major effort. As the model focuses on the learning aspects of the task, other aspects will be ignored or simplified. The model does not model the perceptual-motor parts of the task in detail, but rather uses an ad-hoc lisp-interface to do this. For example, a lisp function perceives all planes in hold level 1 and adds descriptions of them to declarative memory.

Another aspect the model simplifies are the more strategic aspects of the task. The main exploratory learning aspect is learning what planes under what conditions may land on the short runway. Other strategic aspects are not modeled. As a consequence, the model's peak performance (around 2000 points) is not as good as human peak performance (around 3500 points).

The basis for the model is the idea that the instructions are represented in declarative memory, and need to be retrieved and interpreted (Taatgen, 1999b; Anderson, 2000). The production rules that interpret the declarative instructions are not task-specific, and can be used for other tasks as well. The declarative representation that is used is a mixture of ideas expressed by Taatgen (1999b) and by Anderson (2000).

Declarative rules are organized in lists of instructions that are usually executed in order. Each rule has an action that can be supplied with at most two arguments. An argument can be a constant, a variable or a reference. A constant is used as it is. A variable is something that needs a value, for example by retrieving something from declarative memory or by perceiving something in the outside world. Instantiating a variable creates a chunk of type binding, that holds the relation between the variable, its value and the current context. An argument of type reference later retrieves a binding. The creation of these bindings is a way for the model to keep track of aspects of the current task, however, these bindings may be lost due to decay in memory. The following example of a declarative instruction used in the model is part of the instruction to land a plane:

```
land1
        isa instruction
        action perceive-a-plane
        arg1 plane
        type1 variable
        arg2 plane-type
        type2 variable
        prev land
```

```
land2
        isa instruction
        action perceive-weather
        arg1 wind-speed
        type1 variable
        arg2 runway-condition
        type2 variable
        prev land1
```

```
land3
        isa instruction
        action retrieve-experience
        arg1 plane-type
        type1 reference
        arg2 wind-speed
        type2 reference
        prev land2
```

```
land4
        isa instruction
        action decide-no
        arg1 take-long-runway
        type1 constant
        prev land3
```

The first instruction is to perceive an arbitrary plane in hold level 1, and to store it and its type in two variables (which are added to declarative memory as binding-chunks). The second step is to check the weather, and to store the wind-speed and runway-condition. The third step tries to retrieve a past experience concerning the plane-type and the wind-speed. If this past experience is unfavorable, the fourth step decides to take the long runway.

The interpretation process of an instruction involves at least two steps (=production rule firings): the instruction has to be retrieved from memory, and the instruction has to be carried out. Additional steps are necessary if variables and references have to be instantiated, or if the instruction is complicated.

The current model is provided with a declarative instruction to do the ATC task. This instruction is not a literal interpretation of the instructions given to the participants, but reasonable first approximation of a strategy. Another assumption in this strategy is that the model has not memorized all the rules about when a certain plane may land on the short runway, but instead relies on trial-and-error to rediscover these rules. The instructions can be summarized as follows:

Main goal

1. If there are any planes in hold level 1, land one of them
2. Else, move an arbitrary plane from hold level 2 or 3 to hold level 2 or 1.
3. If there are no planes anymore, get between 1 and 6 new planes from the queue.

Landing a plane

1. Select an arbitrary plane in hold level one.
2. Look at the current weather conditions
3. Try to retrieve a past experience with the current plane type and the current wind-speed
4. If the past experience is unfavorable, select the long runway and move the plane there.

5. Try to retrieve a past experience with the current plane and the current runway condition
6. If the past experience is unfavorable, select the long runway and move the plane there
7. If both experiences were favorable, or not present, select the short runway and move the plane there

To move something from A to B

1. Press up or down keys until the arrow is at A
2. Press enter
3. Press up or down keys until the arrow is at B
4. Press enter

Learning in the Model

Four learning mechanisms play a role in the behavior of the model: declarative symbolic, declarative subsymbolic and procedural learning (symbolic and subsymbolic).

Declarative Symbolic Learning

ACT-R keeps past experiences in declarative memory. The current model uses these experiences to decide on whether to land a plane on the short or the long runway. The representation used for examples is restricted to two arguments, the plane type and either the runway condition and the wind speed. As a consequence, the model has no problems learning that 747's can never be landed on the short runway, and prop's always, but it has trouble with the DC10's and 727's, as these planes have complicated rules.

Declarative Subsymbolic Learning

Due to practise, the activation of the instruction chunks and the past experiences chunks steadily increases. As a consequence, retrieval times of these chunks decreases.

Procedural Learning

New productions are learned using a combination of specialization and compilation. Specialization involves substituting variables by constants, more in particular variables that occur in the retrieved chunk. As a consequence, retrieving the chunk is on longer necessary. Compilation involves making one rule out of two rules. In order to make sure the new rule has at most one retrieval, the first rule is specialized first.

This mechanism is an additional module for ACT-R 4.0 (Taatgen, 2000), but incorporated in the new ACT-R 5.0. The main function in the model is that it compiles declarative instructions into production rules. Recall that interpreting instructions takes two steps: retrieving the instruction and carrying out the instruction. Production compilation specializes the retrieval of the instruction, and concatenates the result with the rule that carries out the instruction. The following rules gives an example of pushing enter (rules have been abbreviated for clarity):

```
(p retrieve-instruction
  =goal>
    isa gen-goal
    current =prev
    action nil
  =instr>
    isa instruction
    prev =prev
    action =action
==>
  =goal>
    current =instr
    action =action)
```

```
(p press-enter
  =goal>
    isa gen-goal
    action press-enter
==>
  =goal>
    action nil
!eval! (press-enter))
```

These rules can interpret instructions like:

```
mvhold3 isa instruction action press-enter prev mvhold2
```

Proceduralization produces the following rule given these ingredients:

```
(p compiled-rule
=goal>
   isa gen-goal
```

```
      current mvhold2
      action nil
   ==>
   =goal>
      current mvhold3
      action nil
   !eval! (press-enter))
```

In order to promote a gradual introduction of new rules, their parameters are set to the parameter values derived from the parent rules, plus a penalty on the cost (b) parameter. So a new rule starts out at a slight disadvantage, and is slowly integrated into the system as parameters learning establishes the true values of the production parameters.

Modeling Individual Differences

The three abilities identified by Ackerman are modeled by varying three parameters. General ability is modeled by varying the W-parameter. The W-parameter controls the amount of spreading activation, and is associated with working-memory capacity (Lovett, Reder & Lebiere, 1997). Working-memory capacity itself is strongly correlated with general ability (Kyllonen & Christal, 1990). The simulation uses values 0.8, 1.0 and 1.4 as W-values. Speed of knowledge compilation, measured by Ackerman through perceptual speed, is modeled by varying a parameter that controls proceduralization speed. The parameter determines the probability that, given an opportunity to learn a new rule, the rule is actually learned. Values used are: 0.1%, 0.2%, 0.5%, 5%

Psychomotor speed is modeled by varying the time needed for a key-press. Values used for this parameter are: 150 ms, 200 ms and 250 ms

Results of the Model

In order to assess results of the model, I will compare the model outcomes to the data from

Ackerman (1990). A single run of the model consists of going through 24 trials of 10 minutes each. For each combination of individual difference parameters the model was run twice, producing 4x3x3x2 = 72 runs.

The model's performance in terms of the number of points scored is shown in Figure 3. As the model is only outfitted with a very basic strategy, and no means to improve it, it is no surprise the subjects outperform the model. The shapes of the curves are however similar.

Figure 4 shows correlations between abilities and performance on the ATC-task found by Ackerman, and the correlations between parameter settings and performance of the model. According to Ackerman's theory, these outcomes should resemble the graphs in Figure 1.

Figure 4a and d show the impact of general intelligence. Ackerman measured intelligence by administering a battery of tests for general intelligence (Letter sets, Raven progressive matrices, figure classification and analogies). The model simulates this ability by varying W. A higher value of *W* facilitates the retrieval process by increasing spreading activation. Initially this factor is very important, as both instructions and task information are represented declaratively. As more and more instructions are proceduralized, the stress on declarative memory lessens, so the impact of W on performance decreases.

Figure 4b and e show the impact of speed of proceduralization. Ackerman assessed this ability and psychomotor speed by administering a set of choice-reaction tests (9CRT, 4CRT, 2CRT and a simple reaction test) in 12 blocks. These tests span the range of perceptual speed ability (more choices and less practice) to psychomotor speed (less choices and more practice). Figure 4b uses the results of the first block of the 9CRT, the test at the perceptual-speed extreme of the range, while Figure 4c uses simple-reaction time results in block 12 at psychomotor-speed extreme of the spectrum.

In the model the speed of proceduralization has its main effect in the middle blocks of trials. As proceduralization prerequires some experience with the knowledge it uses to construct new rules, it plays only a small role in the first few trials. Although proceduralization remains an important factor until the end of the experience, its impact trails off slightly, as productions that have the largest impact on performance are learned relatively early.

Figure 4c and f depict the impact of psychomotor speed. In the model this factor becomes more important as experience grows. Although the influence of the effort parameter that models psychomotor speed remains the same, the variance due to other factors decreases, increasing the impact of this psychomotor speed.

Note that for all three abilities, the correlations for the model are larger than the correlations for the data. This should be no surprise, as the parameter manipulations in the model have a direct impact on performance, while assessing these abilities through tests, as is done in the data, is only indirect. Another reason why the correlations in the model are higher is that the model ignores knowledge differences, thereby amplifying the architectural differences.

Discussion

Despite the limitations of the model, it succeeds in going through the three stages of skill acquisition, as demonstrated by the correlations with abilities that characterize these stages. As such it supports the ideas about skill acquisition put forth here and in earlier models based on the same principles (e.g., Taatgen 1999b).

The model also exhibits an example of an ACT-R model where all learning mechanisms are used, instead of a subset of mechanisms for a small task. As such it supports the notion of ACT-R as an architecture of cognition.

One might ask what the added value of a cognitive model is to Ackerman’s theory. It can be observed that the outcomes of the model are much closer to the data than Ackerman’s more qualitative predictions in Figure 1. The model allows the study of what the exact impact of an individual difference related parameter is, and may also help understand other experiments where Ackerman’s theory does not seem to hold. Some issues need further exploration: for example, it is not clear what the exact relationship between perceptual speed and speed of proceduralization is. Ackerman doesn’t have a clear explanation for this. A model of the 9CRT using proceduralization might clarify this issue.

The strategy of the model is still its main limitation: it cannot improve the simple initial strategy very much. The retrieval of examples to guide behavior is something that can be extended, and general strategies to improve on plans can be added. The declarative representation is very flexible, so allows easy modification (as opposed to productions). Work by Lee, Anderson and Matessa (1995) and John and Lallement (1997) may be useful for this purpose.

Finally, the perceptual-motor aspects of the model can be extended to improve is credibility and scope of modeling learning, possibly based on Lee and Anderson (2001).

References

Ackerman, P. L. (1988). Determinants of individual differences during skill acquisition: Cognitive abilities and information processing. Journal of experimental psychology: General, 117(3), 288-318.

Ackerman, P. L. (1990). A correlational analysis of skill specificity: learning, abilities, and individual differences. Journal of experimental psychologie: learning, memory, and cognition, 16(5), 883-901.

Anderson, J. R. (1982). Acquisition of cognitive skill. Psychological Review, 89, 369-406.

Anderson, J. R. (1990). The adaptive character of thought. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (2000). Learning from instructions. Proceedings of the seventh annual ACT-R workshop.

Anderson, J. R. & Lebiere, C. (1998). The atomic components of thought. Mahwah, NJ: Erlbaum.

Fitts, P. M. (1964). Perceptual-motor skill learning. In A. W. Melton (Eds.), Categories of human learning. New York: Academic Press.

John, B. E. & Lallement, Y. (1997). Strategy use while learning to perform the Kanfer-Ackerman air traffic controller task. In M. G. Shafto & P. Langley (Eds.), Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society (pp. 337-342). Mahwah, NJ: Erlbaum.

Kyllonen, P. C. & Christal, R. E. (1990). Reasoning ability is (little more than) working-memory capacity. Intelligence, 14(4), 389-433.

Lee, F. J., Anderson, J. R., & Matessa, M. P. (1995). Components of dynamic skill acquisition. In Proceedings of the 17th annual conference of the cognitive science society (pp. 506-511). Hillsdale, NJ: Erlbaum.

Lee, F.J. & Anderson J.R. (2001). Does learning a complex task have to be complex? A study in learning decomposition. Cognitive Psychology, 42, 267-316.

Lovett, M. C., Reder, L. M., & Lebiere, C. (1997). Modeling individual differences in a digit working memory task. In M. G. Shafto & P. Langley (Eds.), Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society (pp. 460-465). Mahwah, NJ: Erlbaum.

Taatgen, N. A. (1999a). Cognitief Modelleren: Een nieuwe kijk op individuele verschillen. Nederlands tijdschrift voor de psychologie, 54(4), 167-176.

Taatgen, N. A. (1999b). A model of learning task-specific knowledge for a new task. In M. Hahn & S. C. Stoness (Eds.), Proceedings of the 21th annual conference of the cognitive science society (pp. 730-735). Mahwah, NJ: Erlbaum.

Taatgen, N. A. (2000). Learning new production rules in ACT-R. Online document: http://www.ai.rug.nl/~niels/proceduralization.html. University of Groningen, Netherlands.

Author Note

Niels A. Taatgen, Department of Artificial Intelligence.

I would like to thank Frank Lee and John Anderson for their help, and providing me with a Lisp implementation of the ATC task.

Correspondence concerning this article should be addressed to Niels A. Taatgen, Department of Artificial Intelligence, University of Groningen, Grote Kruisstraat 2/1, 9712 TS Groningen, Netherlands. Electronic mail may be sent via Internet to niels@ai.rug.nl

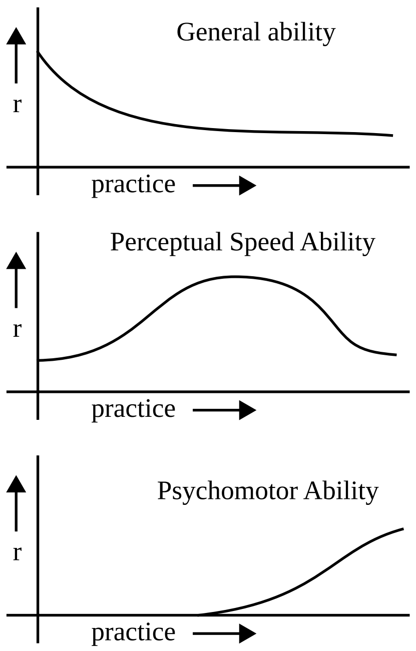
Figure Captions

Figure 1. Predicted ability-performance correlations according to Ackerman. (adapted from Ackerman, 1988).

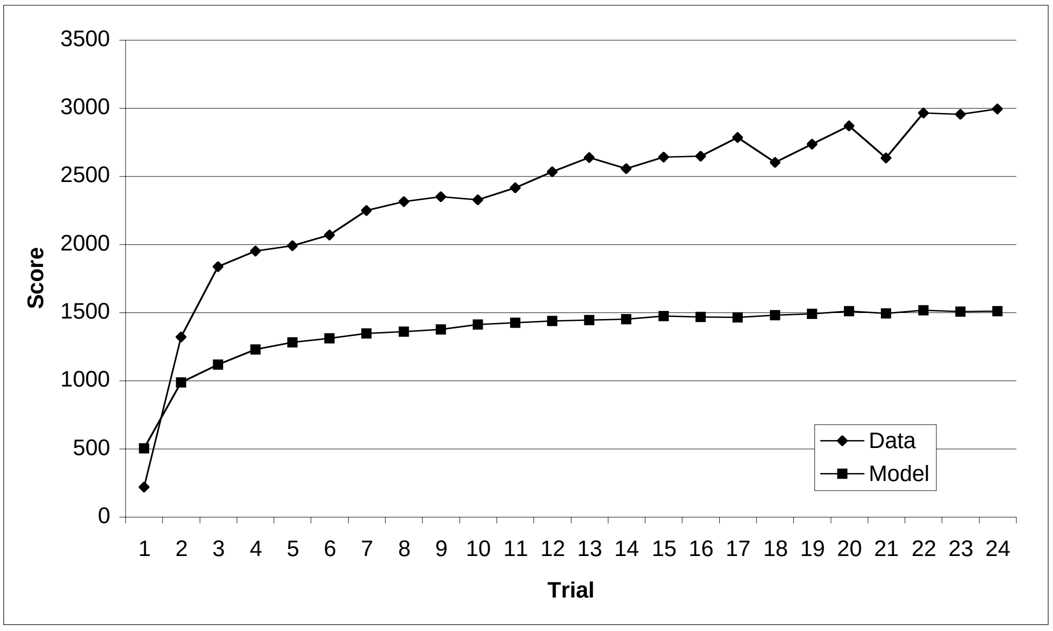
Figure 2. The KA-Air Traffic Controller task

Figure 3. Points scored by the model and subjects in Ackerman (1990).

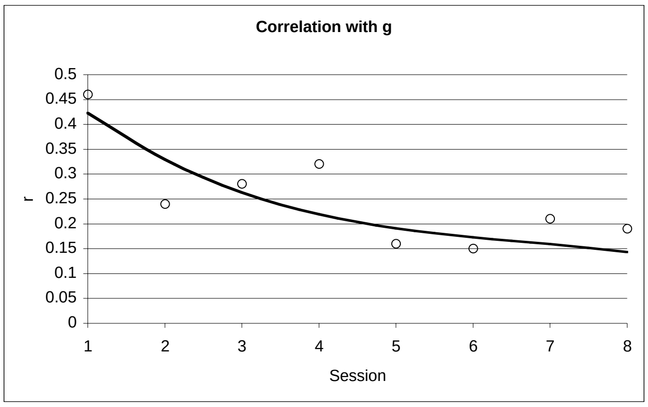
Figure 4. Correlations between Ability scores and performance on the ATC task. (a)-(c) Data from Ackerman (1990) (d)-(f) Outcomes of the model. Solid lines indicate the regression of the ability on practice (cubic polynomial). Each session in the data (a)-(c) consists of three trials.


General ability
r
practice
Perceptual Speed Ability
r
practice
Psychomotor Ability
r
practice

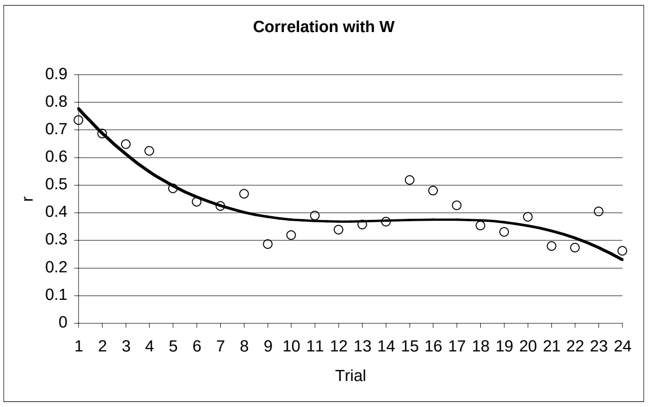
```
  FLT#      TYPE     FUEL      POS.
  ----      ----     ----      ----   Score  :   380
   342      DC10        5       3 n   Landing Pts:  400  Penalty Pts:  -20
   148       727        6       3 s   Runways : DRY
                                3 e   Wind    : 0 - 20 knots from SOUTH
-> 692       747        4       3 w
                                2 n       Flts in Queue: ......
                                2 s            <F1> to accept
   428      prop      * 3       2 e
                                2 w
   259       727        4       1 n
                                1 s
   840      prop        4       1 e
   190      DC10        5       1 w

n ========================= s  #1

n ==================        s  #2

w |||||||||||||||||||||||||  e  #3

w ||||||||||||||||||        e  #4
```


3500
3000
2500
2000
1500
1000
500
0
Score
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Trial
Data
Model

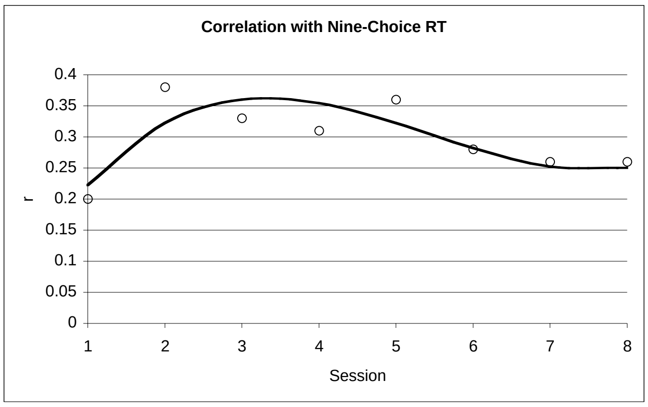
Data **Model**



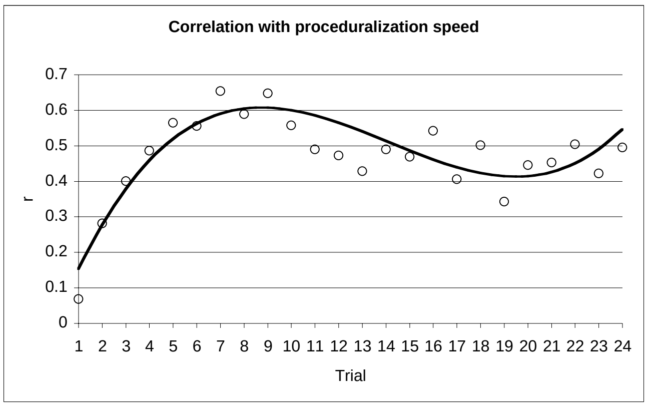
(a)



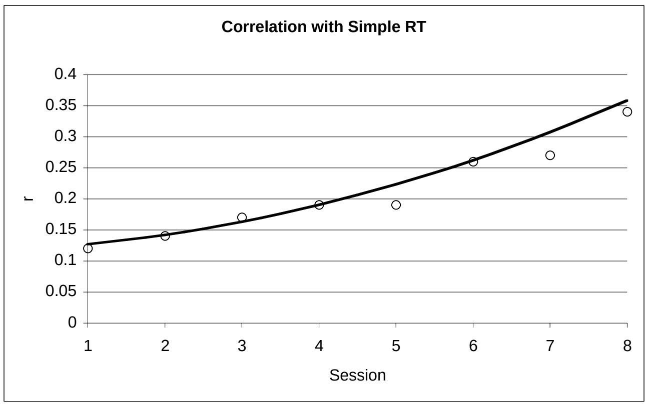
(d)



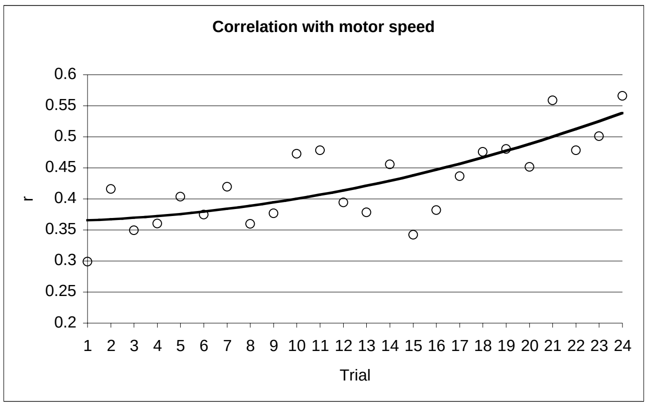
(b)



(e)



(c)



(f)