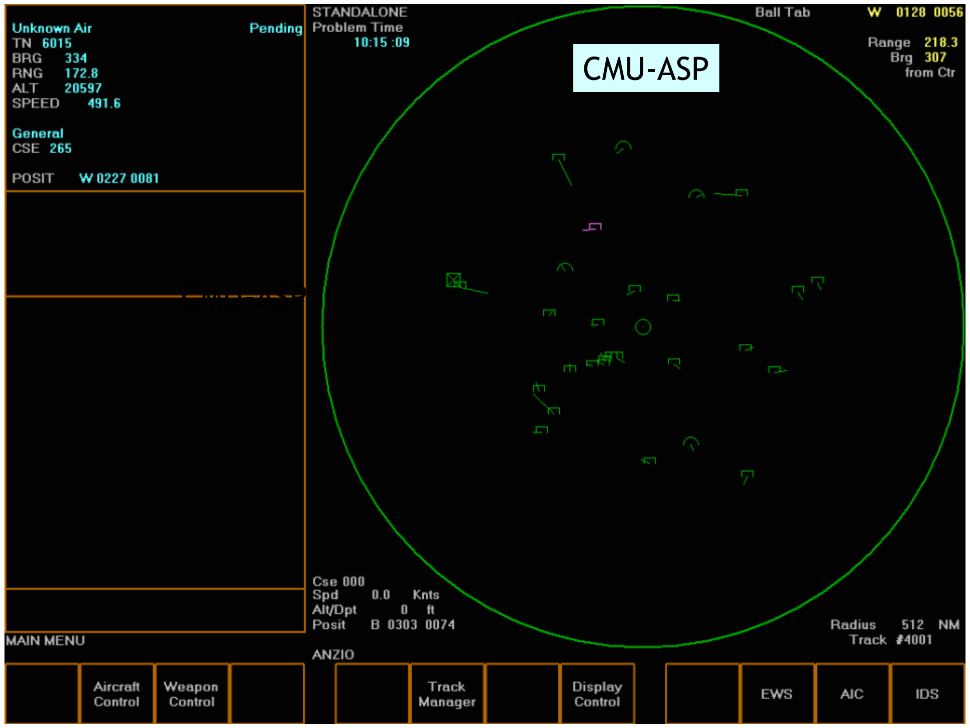
Learning a Complex, Dynamic Skill

John R Anderson

Dan J. Bothell

Scott A. Douglass

Craig Haimson



```
Unknown Air                     Pending   STANDALONE
TN  6015                                  Problem Time
BRG      334                                   10:15 :09
RNG      172.8
ALT      20597
SPEED        491.6

General
CSE  265

POSIT      W 0227 0081
```

```
Ball Tab        W   0128  0056

              Range   218.3
                Brg   307
                  from Ctr
```

```
Cse 000
Spd        0.0     Knts
Alt/Dpt        0     ft
Posit      B  0303  0074

ANZIO
```

```
Radius     512   NM
  Track   #4001
```

Production Compilation(Taatgen & Anderson): The Basic Idea

IF reading the word for a paired-associate test
  and a word is being attended
THEN retrieve the associate of the word
    and note trying to recall

Recall Vanilla-7

IF recalling for a paired-associate test
  and an associated has been retrieved with response N
THEN type N
    and note the answer is being typed

Results in:
IF reading the word for a paired-associate test
  and "vanilla" is being attended
THEN type "7"
    and note the answer is being typed

Instructions for CMU-ASP

1. The task is to identify unidentified tracks. Unidentified tracks are half squares with vectors emanating from them. One should hook (click on) such tracks and then go through the sequence of identifying them. (To identify-tracks first look-for a track that is "half-square" then hook the track then idsequence the track and then repeat)
2. One way to identify a track is to confirm that it is flying at a commercial altitude and speed and then record it as friendly primary id and non-military air id. (To idsequence a track first altitude-test then speed-test and then record it as "friend" "non- military"
3. The other way to identify a track is to request its EWS identity, and then classify the track according to that identity. (To idsequence a track first ews the track for a ews-signal and then classify the track according to the ews-signal)
4. To confirm that a plane is flying at the commercial altitude, look in the upper left, search down for "alt", read the value to the right, and confirm that it is more than 25,000 and less than 35,000. (To altitude-test first seek "upper-left" and then search-down for "alt" at a location then read-next from the location a value then check-less 25000 than the value and then check-less the value than 35000)

Instructions for CMU-ASP

5. To confirm that a plane is flying at the commercial speed, look in the upper left, search down for “speed”, read the value to the right, and confirm that it is more than 350 and less than 550. (To speed-test first seek "upper-left" and then search-down for "speed" at a location then read-next from the location a value then check-less 350 than the value and then check-less the value than 550)
6. To request the EWS identity of a track, select the “ews” key, then select “query sensor status” key, and encode the value that you are told. (To ews a track for a ews-signal first select "ews" then select "query sensor status" and then encode-ews the ews-signal)
7. To classify a track whose EWS identity is ARINC record it as “friendly” primary id and “non-military” air id. (To classify a track according to a ews-signal first match the ews-signal to "arinc564" and then record it as "friend" "non- military")
8. To classify a track whose EWS identity is APQ record it as hostile primary id and strike air id. (To classify a track according to a ews-signal first match the ews-signal to "apq" and then record it as "hostile" "strike")

Instructions for CMU-ASP

9. To classify a track whose EWS identity is APG record it as friendly primary id and strike air id. (To classify a track according to a ews-signal first match the ews-signal to "apg" and then record it as "friend" "strike")
10. To classify a track whose EWS identity is negative treat it as unclassifiable. (To classify a track according to a ews-signal first match the ews-signal to "negative" and then mark-node the track)
11. To record a primary id and a secondary id select the following sequence of keys: “track manager”, “update hooked track”, “class/amp”, “primary-id”, the primary id, “air-id”, the air-id, “save” and then you have succeeded. (To record a primary-id and a air-id first select "track manager" then select "update hooked track" then select "class/ amp" then select "primary id" then select the primary-id then select "air id amp" then select the air-idthen select "save changes" and then success)
12. To select a key, find where it is in the menu and hit the corresponding F-key. (To select a option first find-menu the option at a location and then hit-key corresponding to the location.
13. To find where an item is in the menu, look to the lower left and search to the right for the term. (To find-menu a option at a location first seek "lower-left" and then search-right for the option at a location)

The Basic Plan for Learning from Instruction

- Instructions are encoded as declarative structures characterizing the sequence of goals that must be achieved.
- There are a set of production rules that will interpret any such sequence of instructions.
- Production compilation will convert this into task specific procedures.
- As an aside we solve the mystery of task instructions that has haunted Experimental Psychology.

Production Compilation: Applied to CMU-ASP

IF trying to retrieve a rule to achieve a goal
  and rule for achieving that goal has been retrieved
THEN retrieve the first step of that rule
    and note trying to recall the first step

The first step in recording an id is to select “track”.

IF trying to retrieve the first step of a goal
  and a step has been retrieved involving a subgoal
THEN change goal to trying to achieve that subgoal
    and try to retrieve a rule to achieve that subgoal

Results in:

IF trying to retrieve a rule for recording an id
THEN set a subgoal to select “track”
    and try to retrieve rule for selecting “track”.

Eventually production rules are learned like:

IF trying to retrieve a rule for recording an id
THEN set a hit F1
        and set a subgoal to select "update"

The model moves from taking over 100 seconds to classify a plane to less than 10 seconds. Part of the learning depends on production compilation and part of it depends on location learning.

It sort of learns like participants but does their learning really correspond in detail to the predictions of production compilation?

Niels Taatgen's Subsymbolic Model

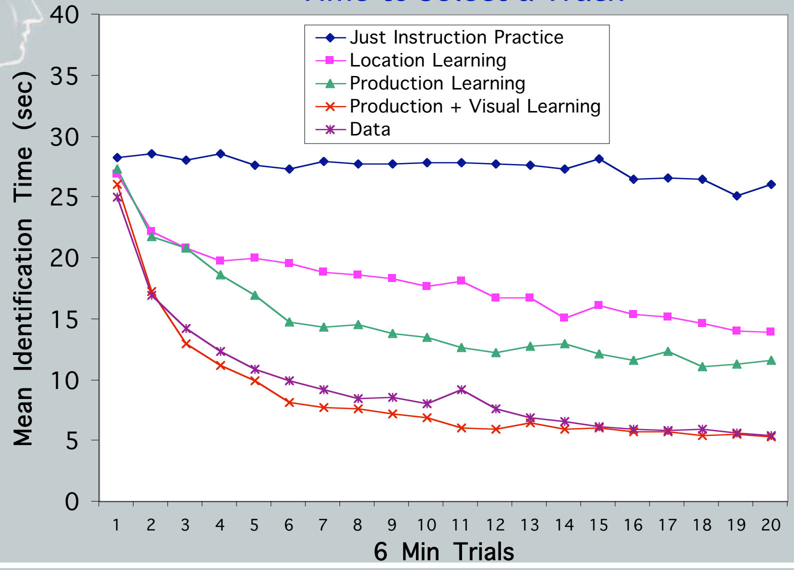
1. Allows for more gradual introduction of rules
2. Based on Rescorla-Wagner Rule
3. More robust across a range of applications

$$EG = \frac{n \bullet priorEG + m \bullet ExperiencedEG}{n + m}$$

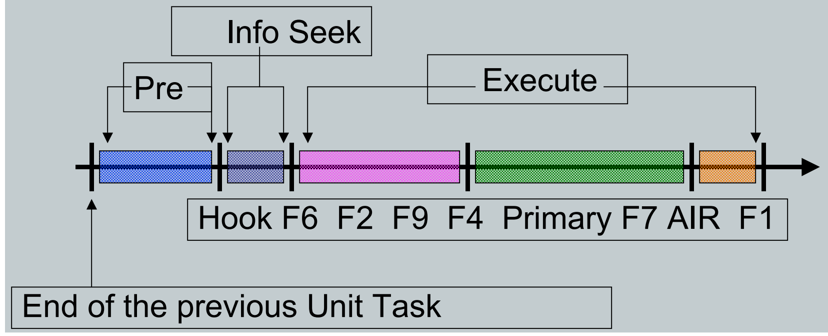
$$priorEG = priorEG + \alpha(parentEG - priorEG)$$

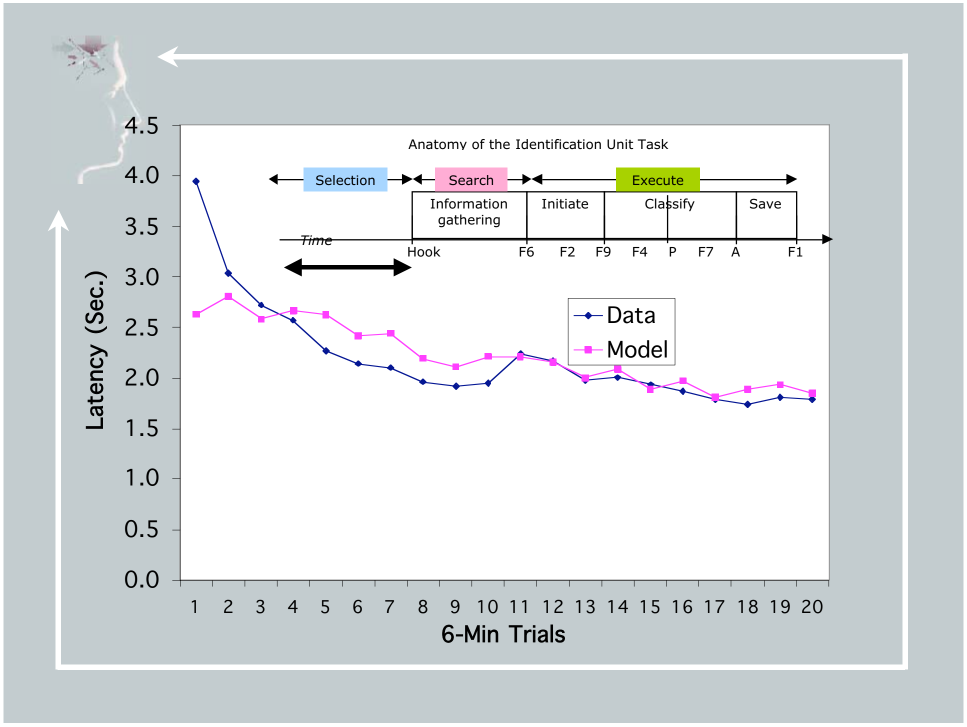
n = 10; α = .05; egs = .4; control rate of production learning
Activation threshold = 1, Noise = .4; controls location learning

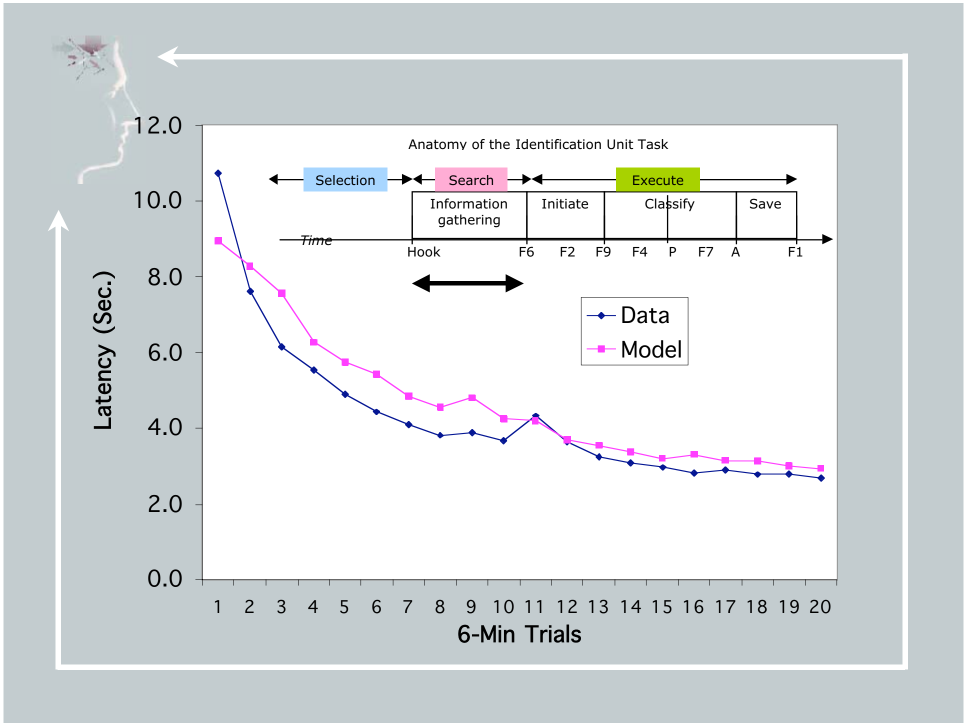
Time to Select a Track

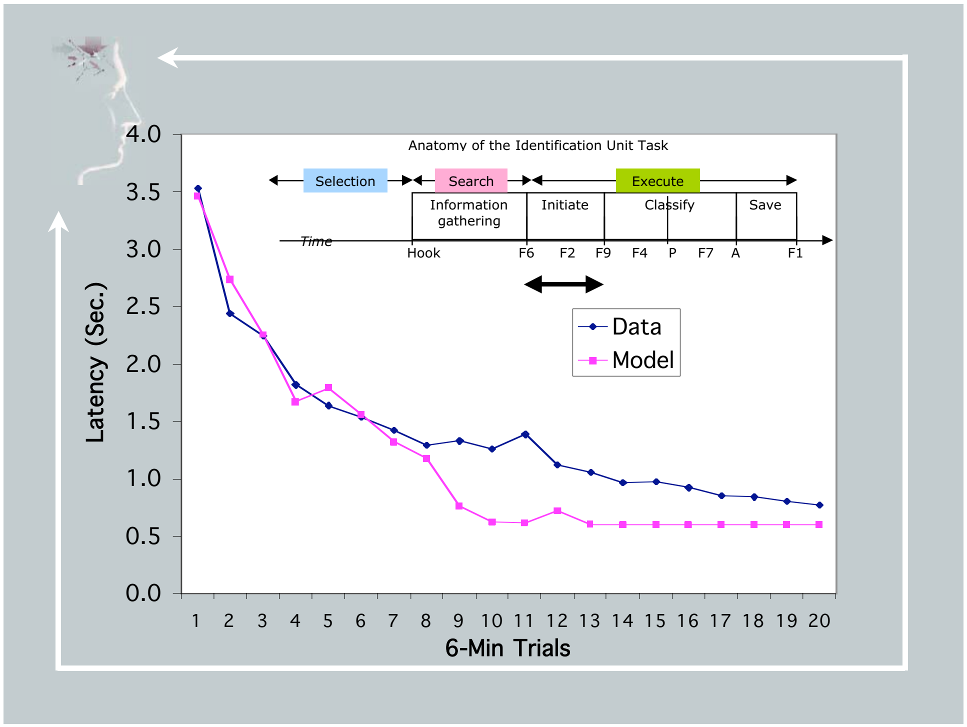


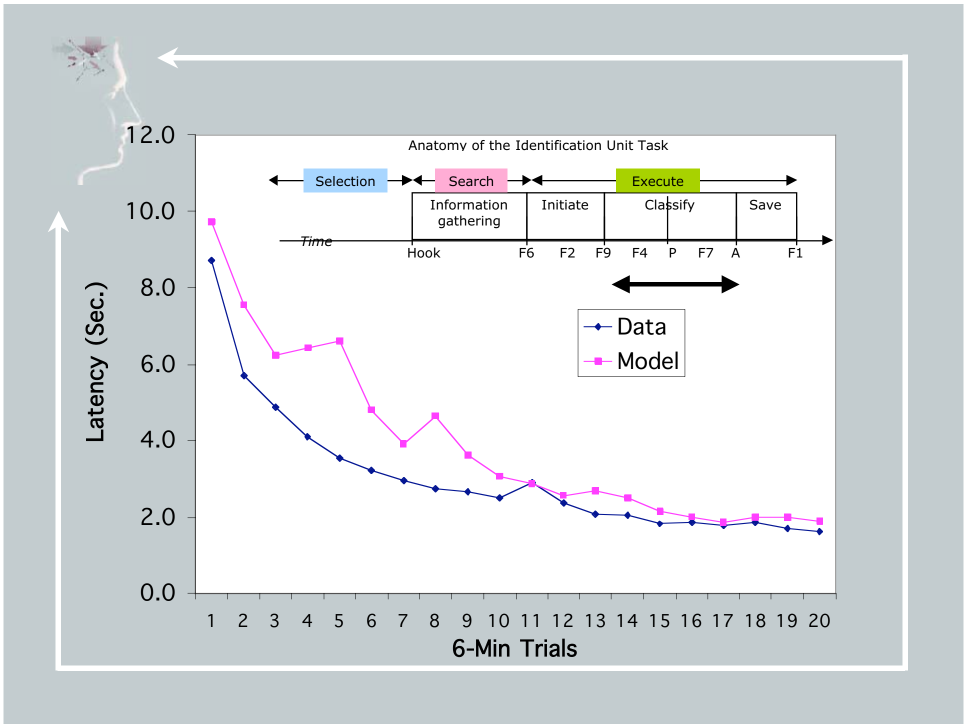
The Anatomy of Track Identification

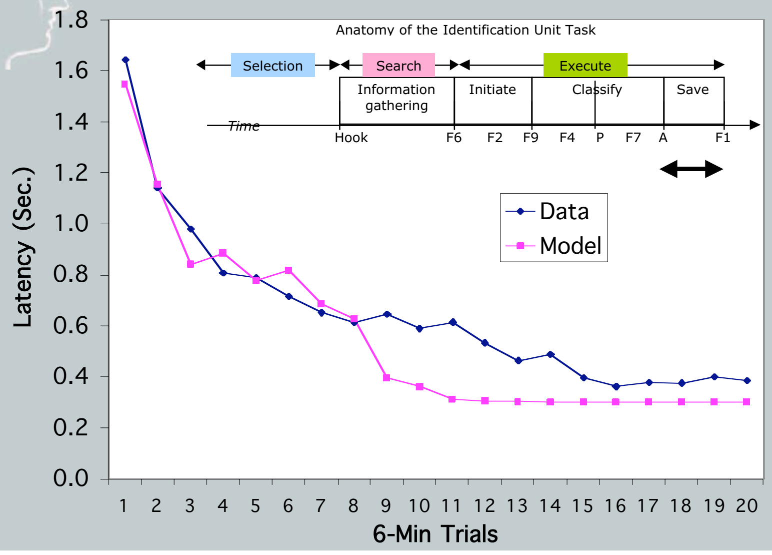

Info Seek
Pre
Execute
Hook F6 F2 F9 F4 Primary F7 AIR F1
End of the previous Unit Task


Anatomy of the Identification Unit Task
Selection
Search
Execute
Information gathering
Initiate
Classify
Save
Time
Hook F6 F2 F9 F4 P F7 A F1
Data
Model
Latency (Sec.)
6-Min Trials

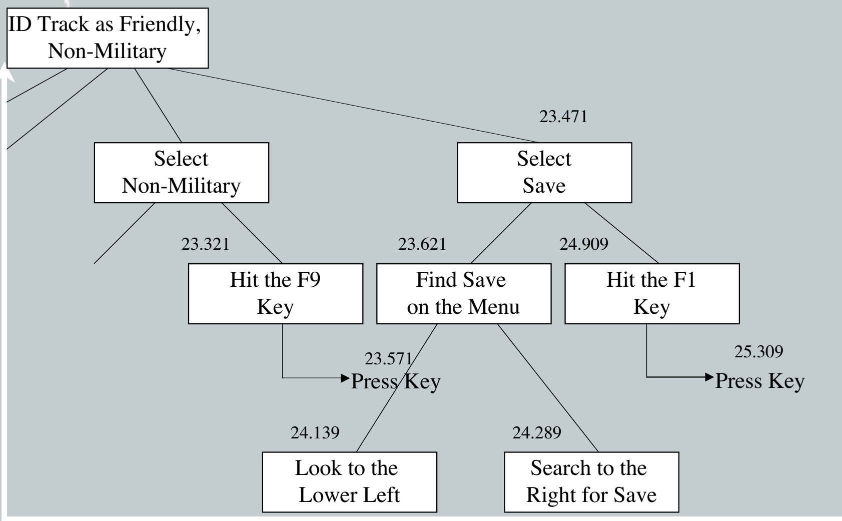

Anatomy of the Identification Unit Task
Selection
Search
Execute
Information gathering
Initiate
Classify
Save
Time
Hook F6 F2 F9 F4 P F7 A F1
Data
Model
Latency (Sec.)
6-Min Trials


Anatomy of the Identification Unit Task
Selection
Search
Execute
Information gathering
Initiate
Classify
Save
Time
Hook F6 F2 F9 F4 P F7 A F1
Data
Model
Latency (Sec.)
6-Min Trials

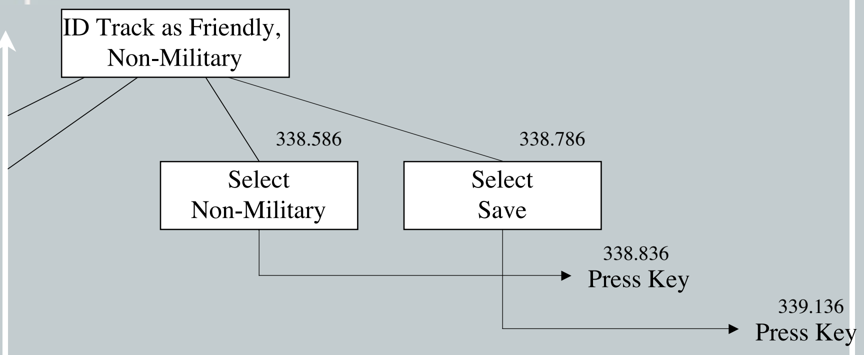

Anatomy of the Identification Unit Task
Selection
Search
Execute
Information gathering
Initiate
Classify
Save
Time
Hook F6 F2 F9 F4 P F7 A F1
Data
Model
Latency (Sec.)
6-Min Trials

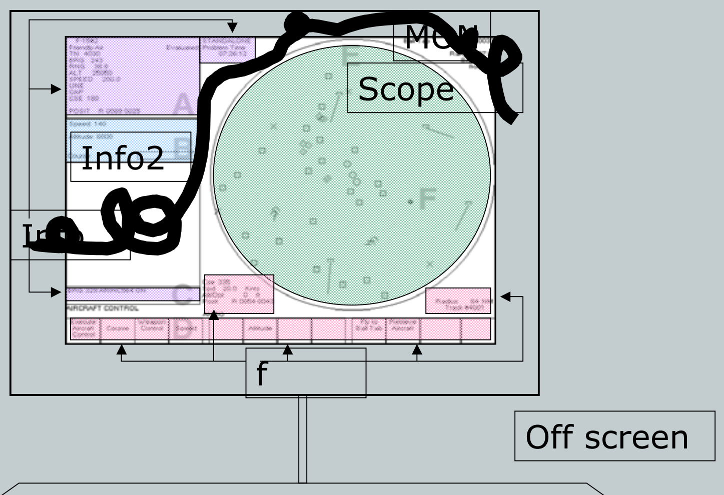


The First Interkey Press for Save

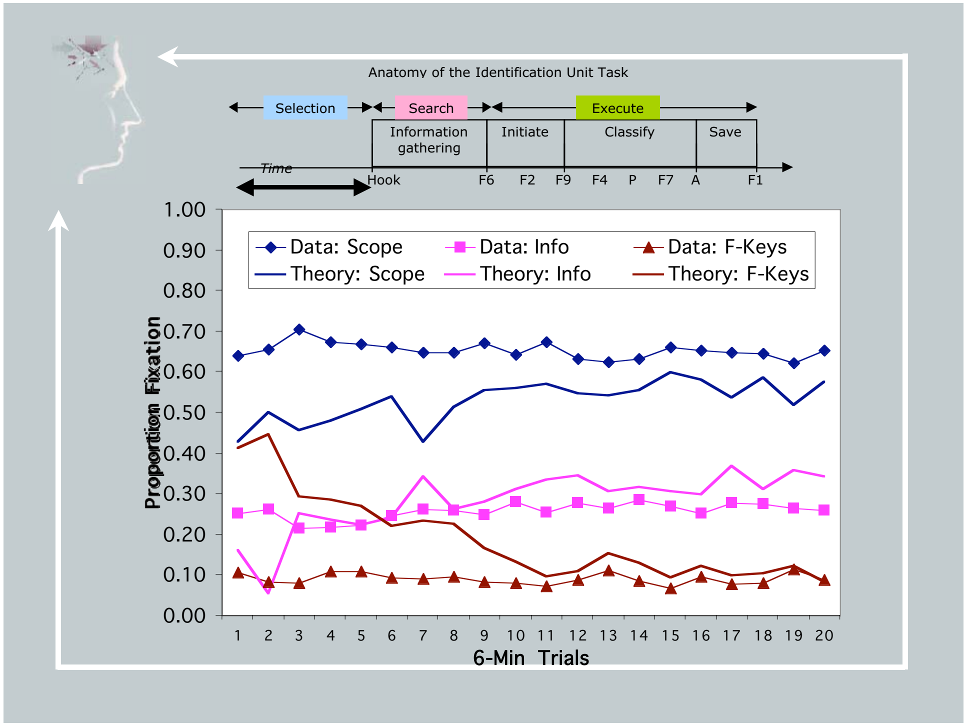


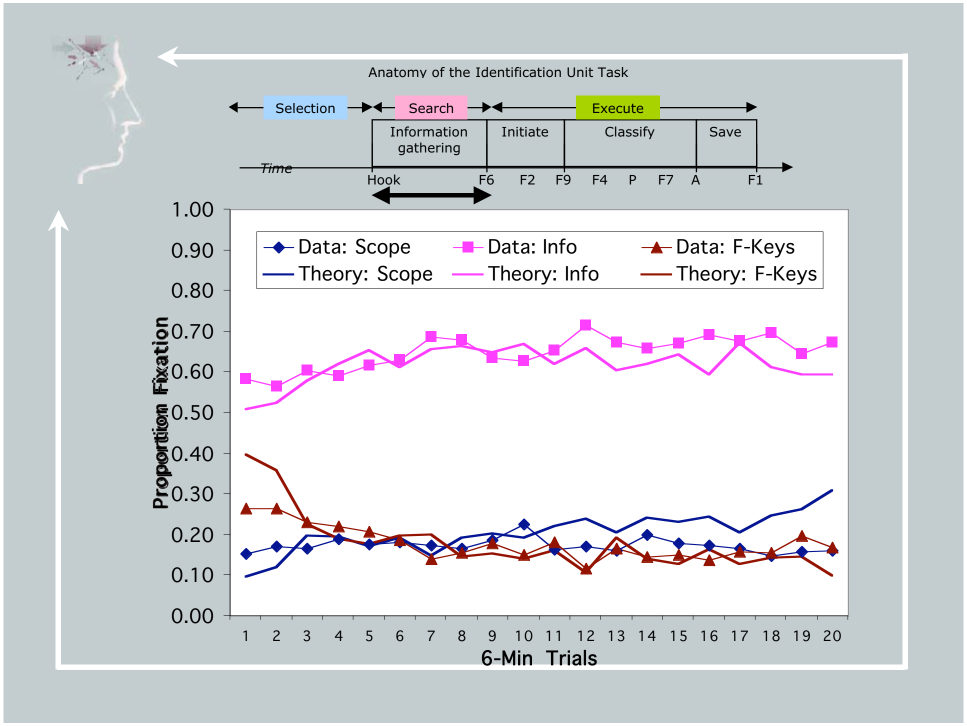
The Last Interkey Press for Save

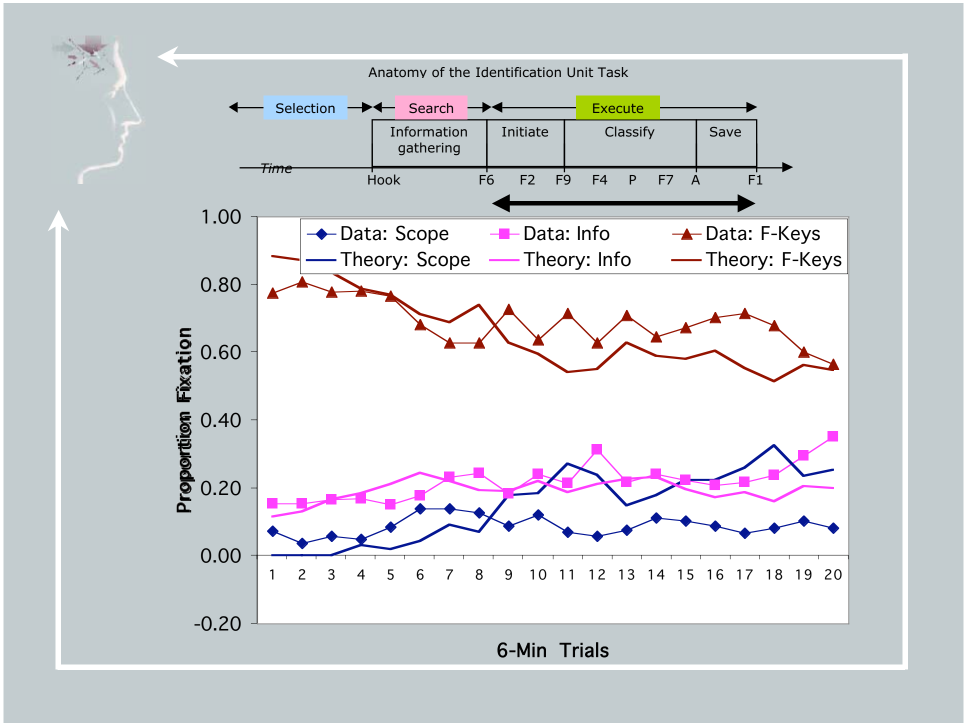




51% of all eye fixation times are not to relevent regions







Conclusions

1. We seem to a a viable mechanism for creating productions and learning from instruction. Why did it take so long?
2. There still are open issues as to what the best way to represent instructions and interpret them are and how to deal with flexibility
   (a) My solution involves default seriality
   (b) Niels solution involves explicitly representing ordering constraints
3. There are open issues about how to deal with control
   (a) Inserting override instructions
   (b) Evoking the instructions upon condition