Mental Models

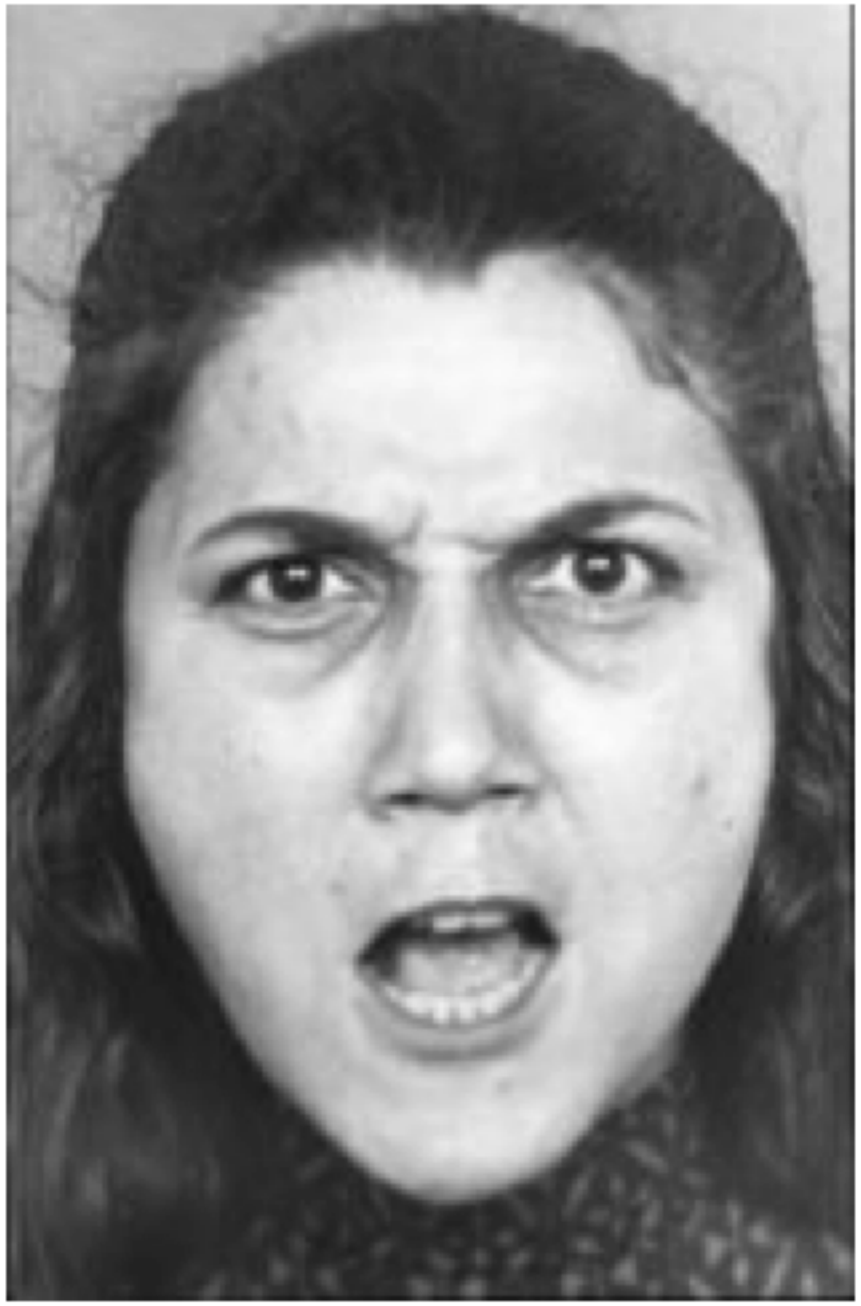
SWE 795, Spring 2017

Software Engineering Environments

Today

- Part 1 (Lecture)(~50 mins)
  - Mental Models
- Part 2 (Project Presentations)(~40 mins)
- Break!
- Part 2 (Discussion)(45 mins)
  - Discussion of readings

What is this emotion?



System 1 vs System 2

System 1	System 2
Automatic (unconscious)	Voluntary (conscious)
Effortless	Effortful
“Fast” thinking	“Slow” thinking
Associative	Planning
Heuristic	Logical
Gullible	Lazy
Can’t be turned off	Usually only partly on

Examples of System 1

- Detect that one object is more distant than another.
- Orient to the source of a sudden sound.
- Complete the phrase “bread and…”
- Make a “disgust face” when shown a horrible picture.
- Answer to 2 + 2 = ?
- Drive a car on an empty road.
- Understand simple sentences.

Examples of System 2

- When System 1 does not offer an answer (e.g., 17 x 24)
- When an event is detected that violates the model of the world that System 1 maintains (e.g., cat that barks)
- Continuous monitoring of behavior—(keeps you polite when you are angry)
- Normally has the last word

Attentional resources are fixed

- Demo

Attentional resources are fixed

- System 2 activity requires attention
- Attentional resources are fixed
- Pupils dilate as mental effort increase
- If demands exceed max, tasks prioritized.

Examples of attention limitations

- Can walk and talk
- But not walk and compute 23 x 78
- Constructing complex argument better when still

Coexistence of Systems 1 and 2

- System 1 processes normal, everyday, expected activities at low cost.
- System 2 takes over when necessary, at higher cost.
- Law of least effort: pays for System 2 to be lazy.

Short term memory (STM)

- Primary, active memory used for holding current context for System 2
- Unless actively maintained (or encoded to long-term memory), decays after seconds
- Capacity ~ 4 items
  - (classic estimate of 7 +/- 2 is wrong)

Chunking: What’s easiest to remember?

- A lock combination with 8 numbers in order: 10, 20, 30, 40, 50, 60, 70, 80
- A lock combination with 8 numbers in order: 50, 30, 60, 20, 80, 10, 40, 70
- A string of 10 letter: R, P, L, B, V, Q, M, S, D, G
- A string of 52 letters: I pledge allegiance to the flag of the United State of America.

Chunking

- Items in memory encoded as **chunks**
- A chunk may be anything that has meaning
- # of chunks in STM fixed, but remembering bigger chunks lets you remember more
- Memory retention relative to the concepts you already have

Long term memory (LTM)

- Items in short term memory may be encoded into storage in long term memory
- LTM capacity not limited
- Information must be retrieved from long term memory (i.e., through System 1)
- Many factors influence what is encoded into LTM and how it is encoded

Memory is reconstructive - example

- How fast was the car going when it hit the other vehicle?

vs.

- How fast was the the car going when it smashed into the other vehicle?
- 2x more remember seeing broken glass

Memory is reconstructive

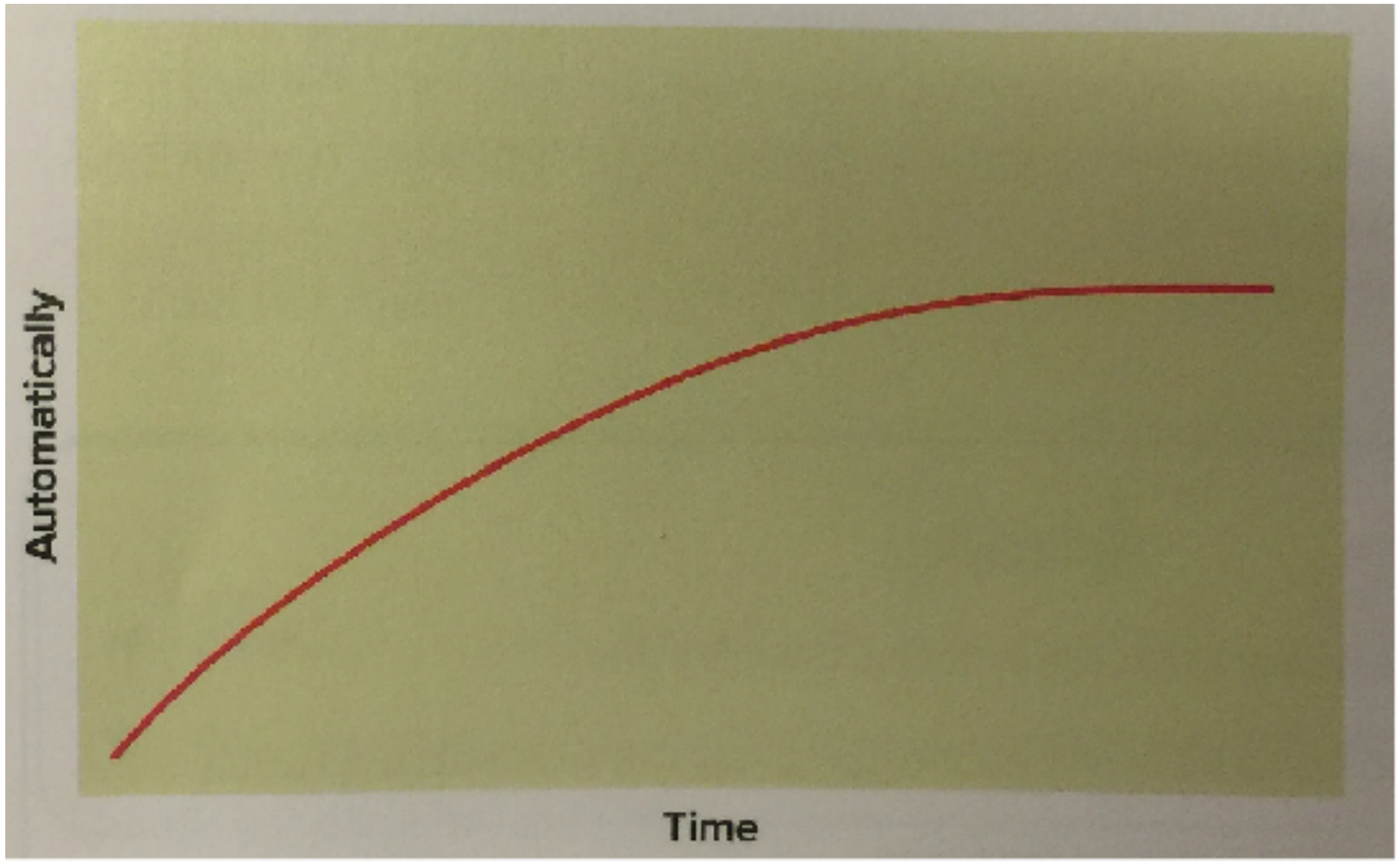
- Not stored files on a disk
- Encoded in brain, may be different every time retrieved
- Remember pieces, reconstruct other details based on expectations on what must have occurred
- Hard to distinguish similar memories

Automaticity

- This effect happens for sequences of actions ("**scripts**") as well.
    - Example: tying shoelaces
- More repetitions, faster, requires less conscious attention.
- Responsibility shifts from System 2 —> System 1

Habit formation takes time

- How long does it take to form a eating, drinking, or activity habit?
- Mean: 66 days, Min: 18 days, Max: 254 days
- More complex behaviors take longer to become habit

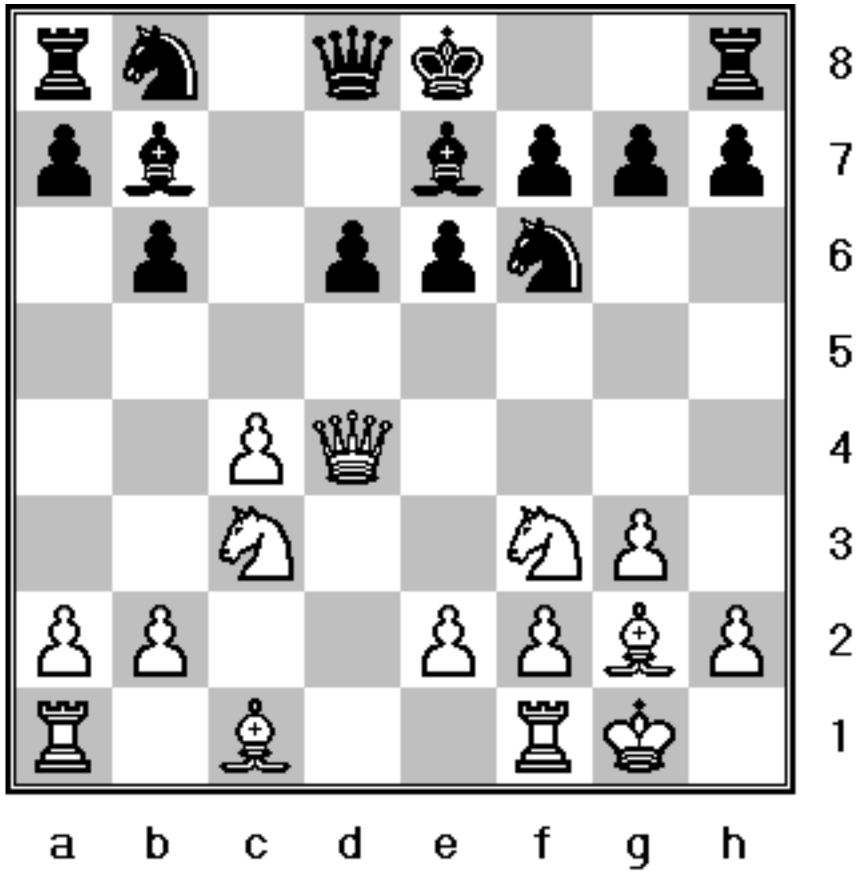


What makes an expert?

- Experts are more intelligent?
  - IQ doesn’t distinguish best chess players or most successful artists or scientists (Doll & Mayr 1987) (Taylor 1975)

- Experts think faster or have larger memory?
  - World class chess experts don’t differ from experts

What makes a grand master a chess expert?



- Memory for **random** chess boards: **same** for experts and novices
- Memory for position from **actual** game: much better for **experts** than novices
- [deGroot 1946; Chase & Simon 1973]

Experts create schemas by chunking world

- Schema: a template (struct) describing a set of slots

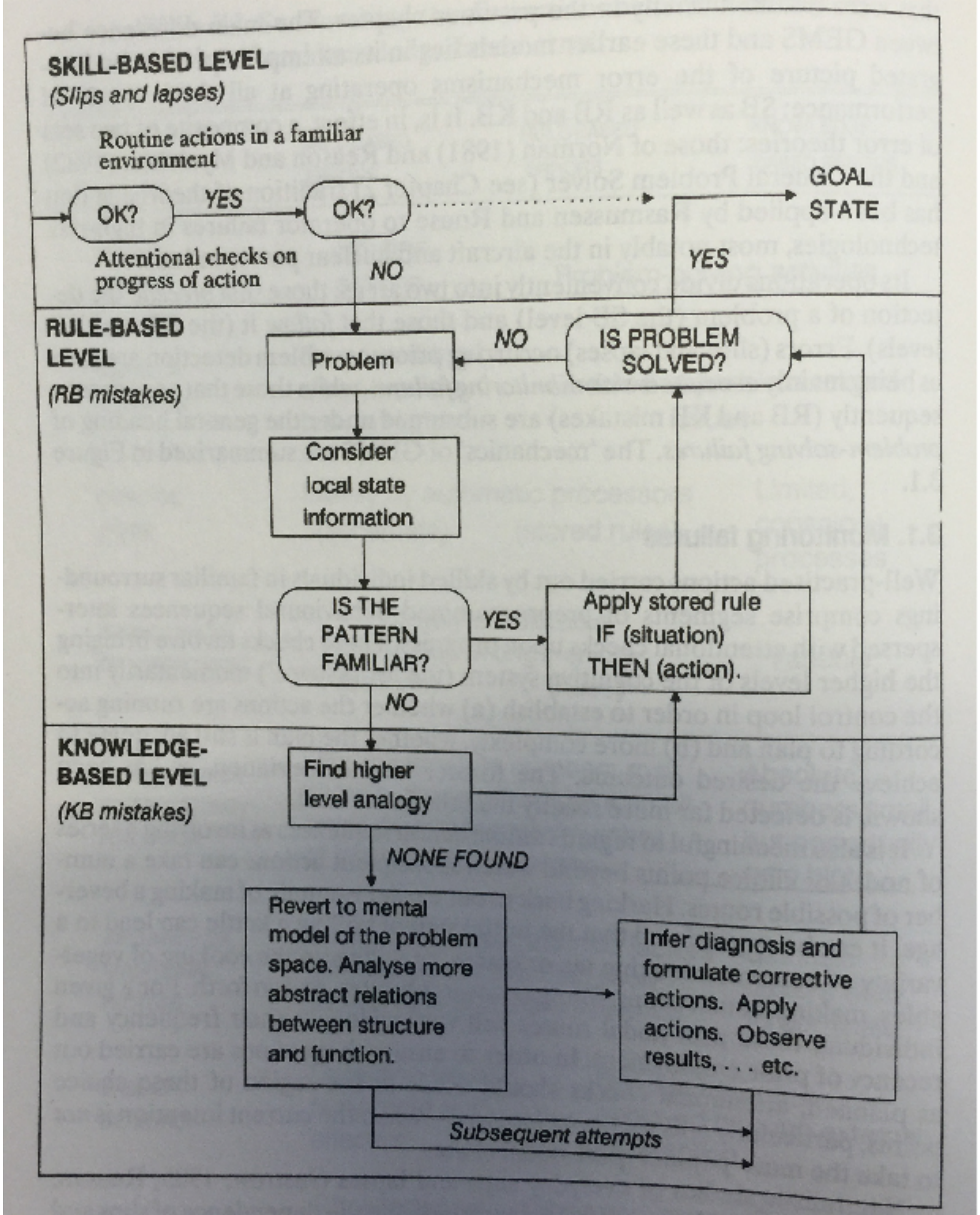
```
while (x > 0)
{
    invokeAction(actions[x]);
    x—;
}
```

- Experts perceive the world through schemas
  - “Chunk” and interpret visual stimuli to determine which schemas are present
  - Form concepts that help developers think in abstractions

Experts have different types of knowledge

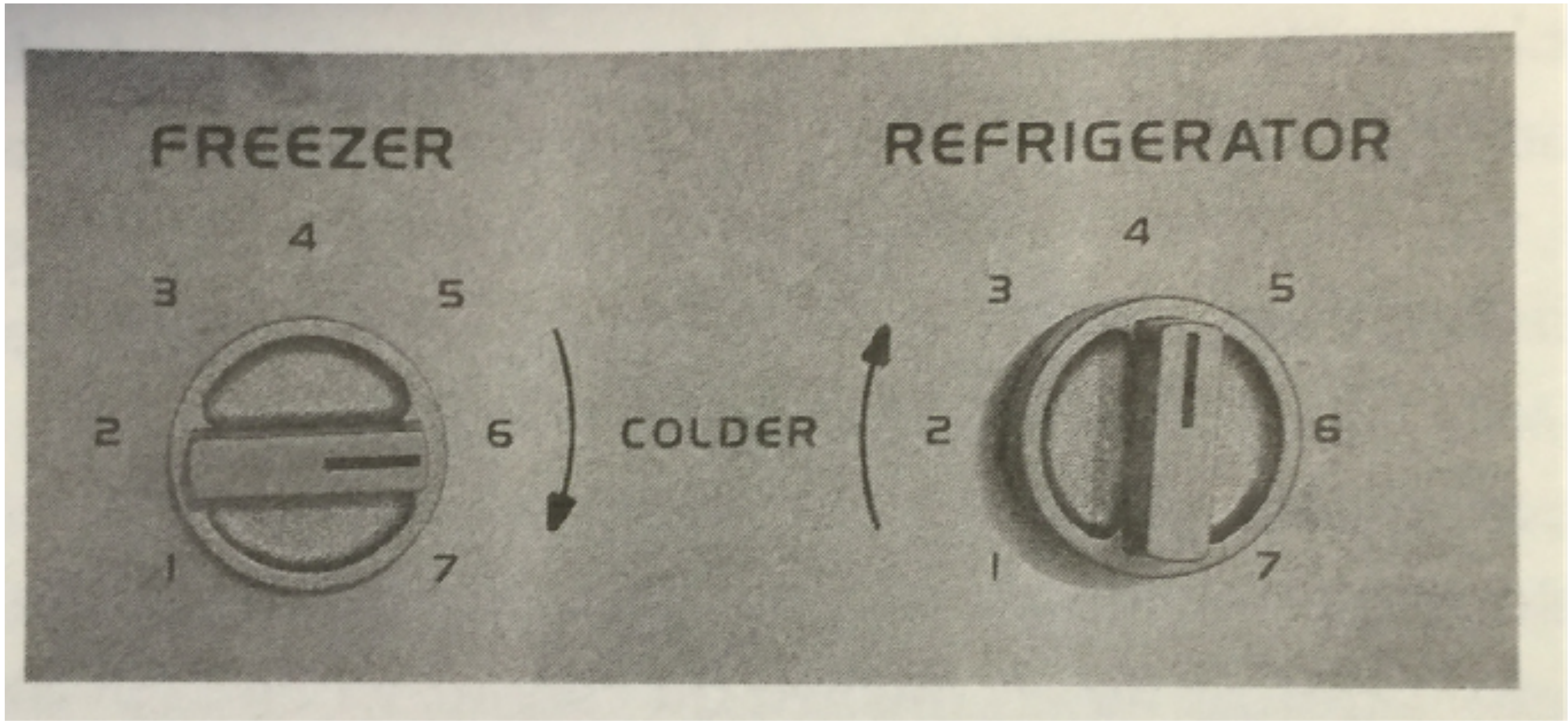
- Skills — requires very little or no conscious control to perform or execute an action once an intention is formed
  - e.g., riding a bicycle
- Rules— use of rules and procedures to select a course of action in a familiar situation.
  - e.g., following fire alarm procedure
- Knowledge— a more advanced level of reasoning employed when the situation is novel and unexpected. Operators are required to know the fundamental principles and laws by which the system is governed.

[Rasmussen, 1990]


SKILL-BASED LEVEL
(Slips and lapses)
Routine actions in a familiar environment
OK?
YES
OK?
GOAL STATE
Attentional checks on progress of action
NO
YES
RULE-BASED LEVEL
(RB mistakes)
Problem
NO
IS PROBLEM SOLVED?
Consider local state information
IS THE PATTERN FAMILIAR?
YES
Apply stored rule IF (situation) THEN (action).
NO
KNOWLEDGE-BASED LEVEL
(KB mistakes)
Find higher level analogy
NONE FOUND
Revert to mental model of the problem space. Analyse more abstract relations between structure and function.
Infer diagnosis and formulate corrective actions. Apply actions. Observe results, . . . etc.
Subsequent attempts

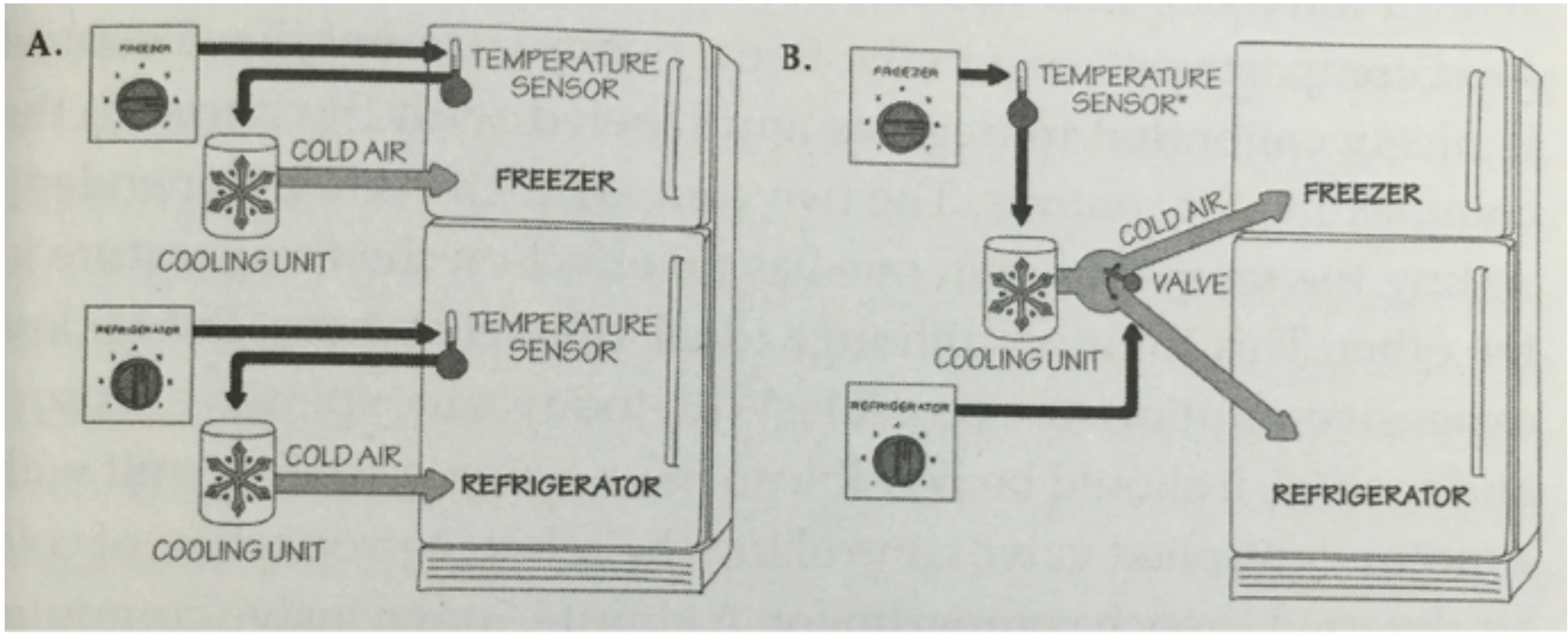
Mental models (a.k.a conceptual models)

- Internal representation in the head of how something works in the real world
- E.g., changing appropriate knob adjusts temperature in freezer or refrigerator



Mental models: Refrigerator Example

- Only single temperature sensor.
- Controls not independent, need to adjust both.
- (also delayed feedback)



Mental Models of Programs

- Program comprehension as text comprehension
  - Schemas, beacons, and plans
- Effects of expertise on mental models
- Program comprehension as fact finding

Program comprehension as text comprehension

- Developers recognize specific “beacons” (a.k.a. features) in code that activate schemas
    - e.g., for (elem in elements)

- Developers mentally represent programs in terms of schemas
    - Reason about behavior of program using schemas
    - Recall what code is or is not present using schemas

Developers perceive programming plan, control flow, data flow representations

- Build and possess different abstractions of code

- Programming plan
  - Hierarchic decomposition of goals in program

- Control flow
  - Control flow in a method

- Data flow
  - Data flow in a method

Implications of text comprehension

- Distortions of form in recall
  - Developers more likely to recall prototypical schema values rather than actual.

- Distortions of content
  - Developers more likely to recall values inferred from schemas that were not present in code.

Program comprehension as fact finding

- Traditional models of program comprehension built for tiny programs (<100 lines)
  - How do developers comprehend realistic (e.g., 100K) sized programs?

- Recognition of schemas usually treated as a binary state (e.g., present or not present)
  - What happens when developers have uncertainty about what they believe?

Developers seek and learn facts about code

JEditBuffer

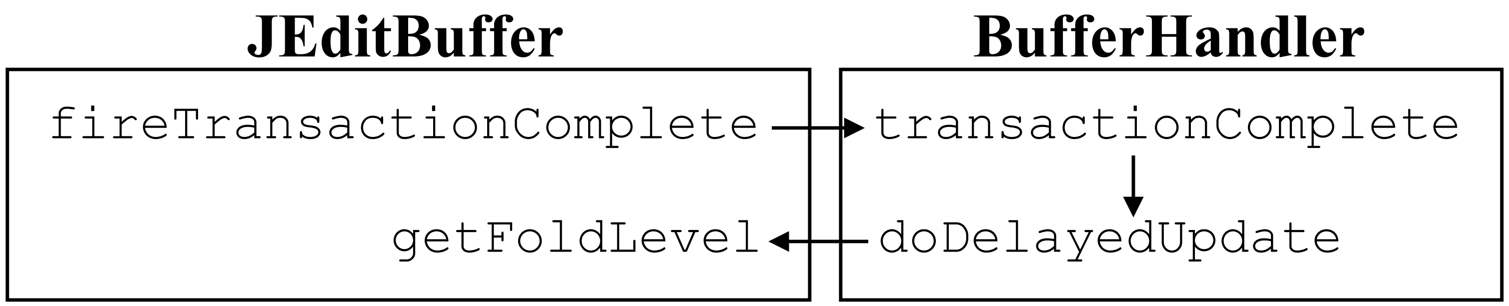
getFoldLevel

getFoldLevel has effects

READ getFoldLevel LEARN *getFoldLevel has effects*

“Yes, I am indeed surprised that get fold level has side effects. That is surprising to me.”

Developers seek to explain facts to discover hidden constraints which might be violated by a change



Folds updated by buffer changes are updated on screen

getFoldLevel has effects

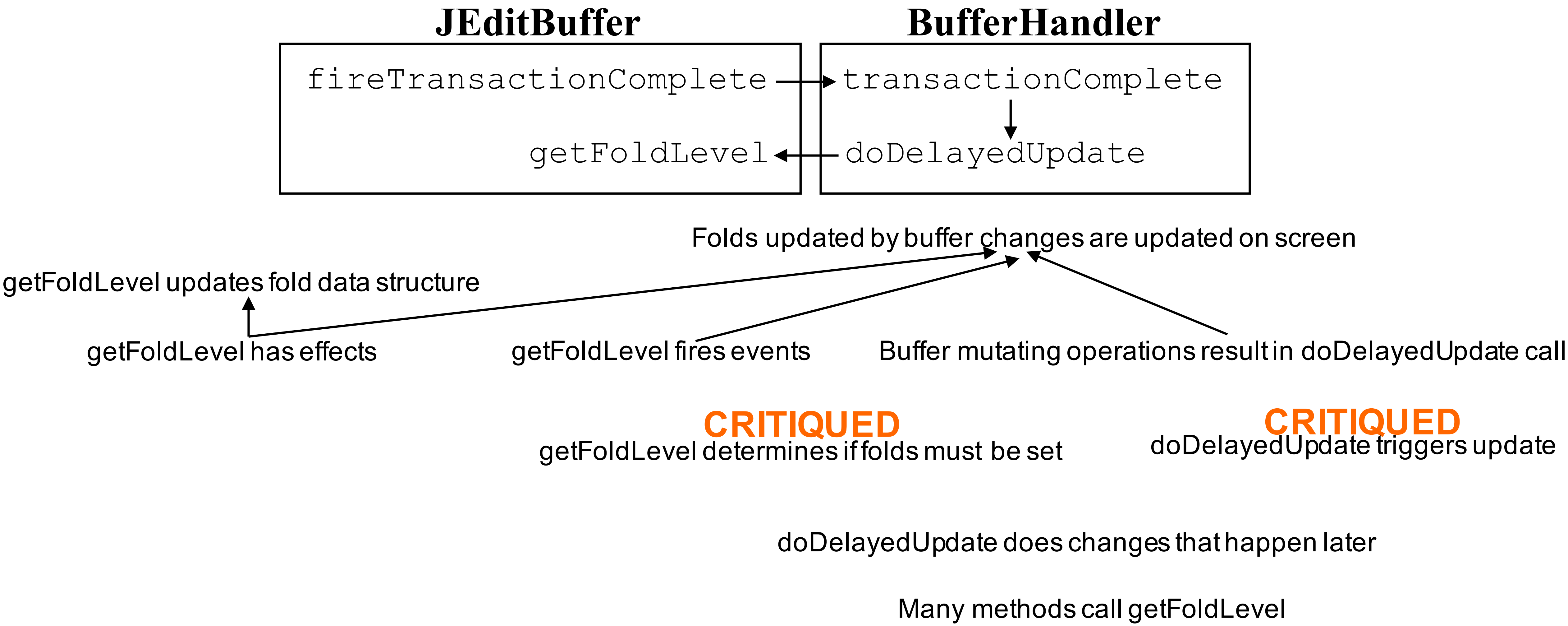
Buffer mutating operations result in doDelayedUpdate call

doDelayedUpdate does changes that happen later

Many methods call getFoldLevel

LEARN 2 - *folds updated by buffer changes are updated on screen*
2 EXPLAINS *buffer mutating operations result in doDelayedUpdate call*
“When you're inserting text you could actually doing something that makes the folds status wrong. … In the quick brown fox. If fox is under brown and I'm right at fox and I hit backspace. Then I would need to update my fold display to reflect the new reality which is that it's in a different place. It's now a child of quick, not a child of brown.”

Developers critique poor facts, proposing changes to make



JEditBuffer

fireTransactionComplete

getFoldLevel

BufferHandler

transactionComplete

doDelayedUpdate

Folds updated by buffer changes are updated on screen

getFoldLevel updates fold data structure

getFoldLevel has effects

getFoldLevel fires events

Buffer mutating operations result in doDelayedUpdate call

CRITIQUED

getFoldLevel determines if folds must be set

CRITIQUED

doDelayedUpdate triggers update

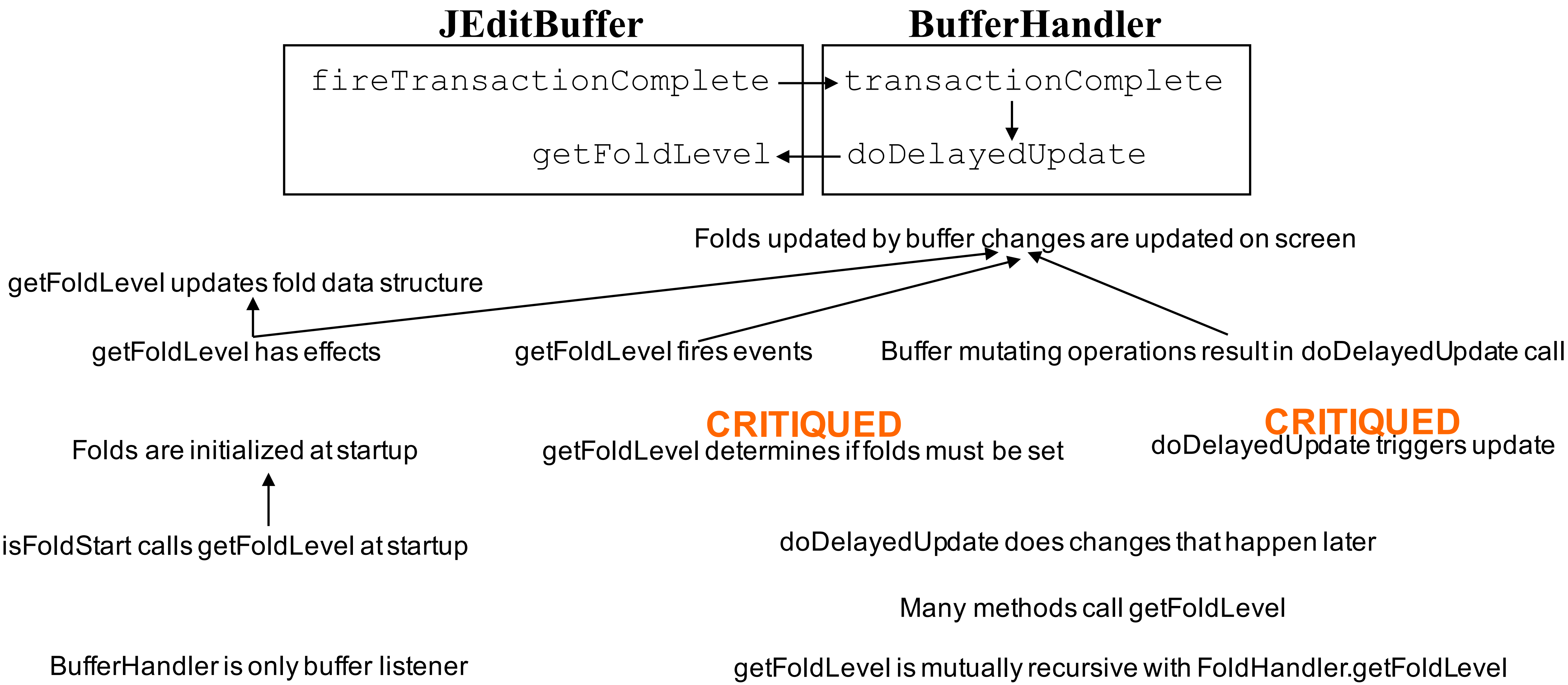
doDelayedUpdate does changes that happen later

Many methods call getFoldLevel

CRITIQUE *doDelayedUpdate triggers update*

"And the second thing that I don't like is that it is firing these updates. ... It shouldn't be relying on one of these guys to be calling this update routine manually."

Developers propose changes to accomplish changes, subject to respecting discovered constraints



JEditBuffer

fireTransactionComplete

getFoldLevel

BufferHandler

transactionComplete

doDelayedUpdate

Folds updated by buffer changes are updated on screen

getFoldLevel updates fold data structure

getFoldLevel has effects

getFoldLevel fires events

Buffer mutating operations result in doDelayedUpdate call

CRITIQUED

getFoldLevel determines if folds must be set

CRITIQUED

doDelayedUpdate triggers update

Folds are initialized at startup

isFoldStart calls getFoldLevel at startup

doDelayedUpdate does changes that happen later

Many methods call getFoldLevel

BufferHandler is only buffer listener

getFoldLevel is mutually recursive with FoldHandler.getFoldLevel

PROPOSE

“So, in an ideal world, when would this notification going out go out. Cause there's this insert here. Then there's a fireTransactionComplete from the insert. This might be a good place to notify somebody to update this data structure about the fold levels.”

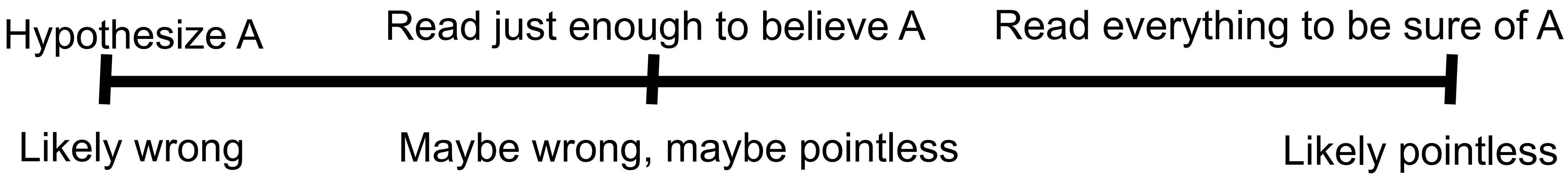
Developers seek changeable facts to accomplish their goals

START *updateCaretStatus called 7 times*

GOAL *updateCaretStatus called fewer times*

Schema recognition under uncertainty

Uncertainty about facts



Hypothesize A

Read just enough to believe A

Read everything to be sure of A

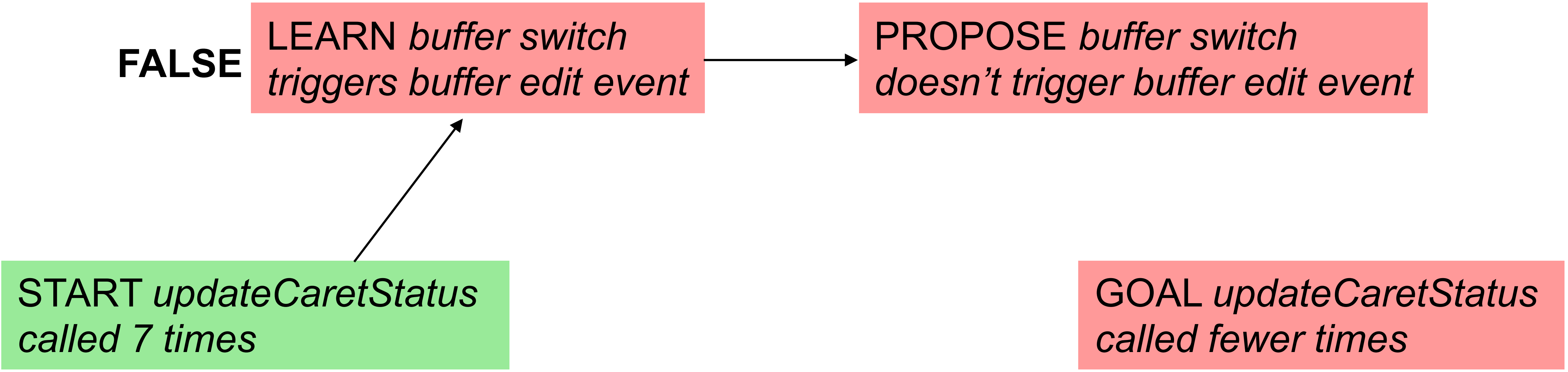
Likely wrong

Maybe wrong, maybe pointless

Likely pointless

Knowledge shifts how much need to know
Could also think about cost benefit of being wrong, reading as shifting this

Developers learn false facts, leading to changes that will not work



FALSE

LEARN *buffer switch triggers buffer edit event*

PROPOSE *buffer switch doesn't trigger buffer edit event*

START *updateCaretStatus called 7 times*

GOAL *updateCaretStatus called fewer times*

"'Cause I'm thinking that when I perform the action of switching from one buffer to another buffer, somewhere it calls a method that indicates that the buffer has been edited. But I didn't edit the buffer. I'm just switching between buffers. So that has to be removed."