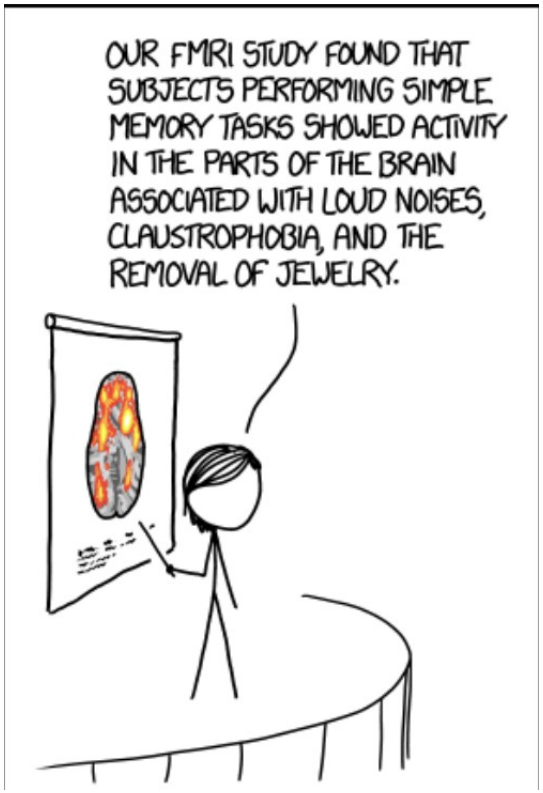
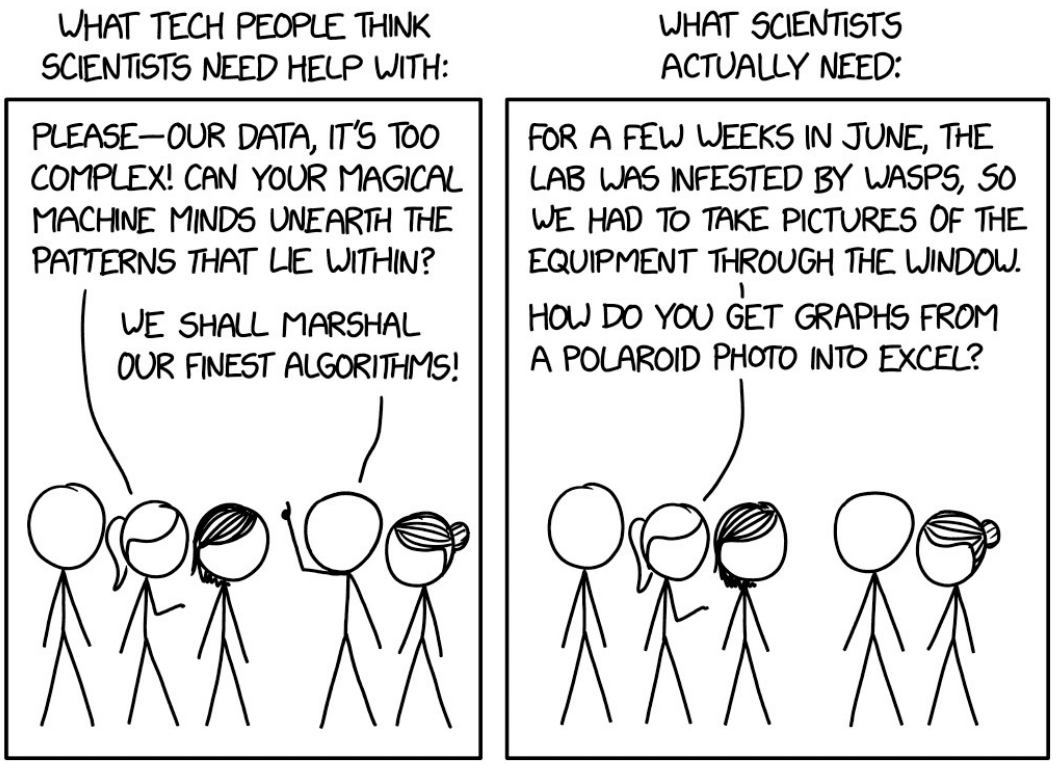
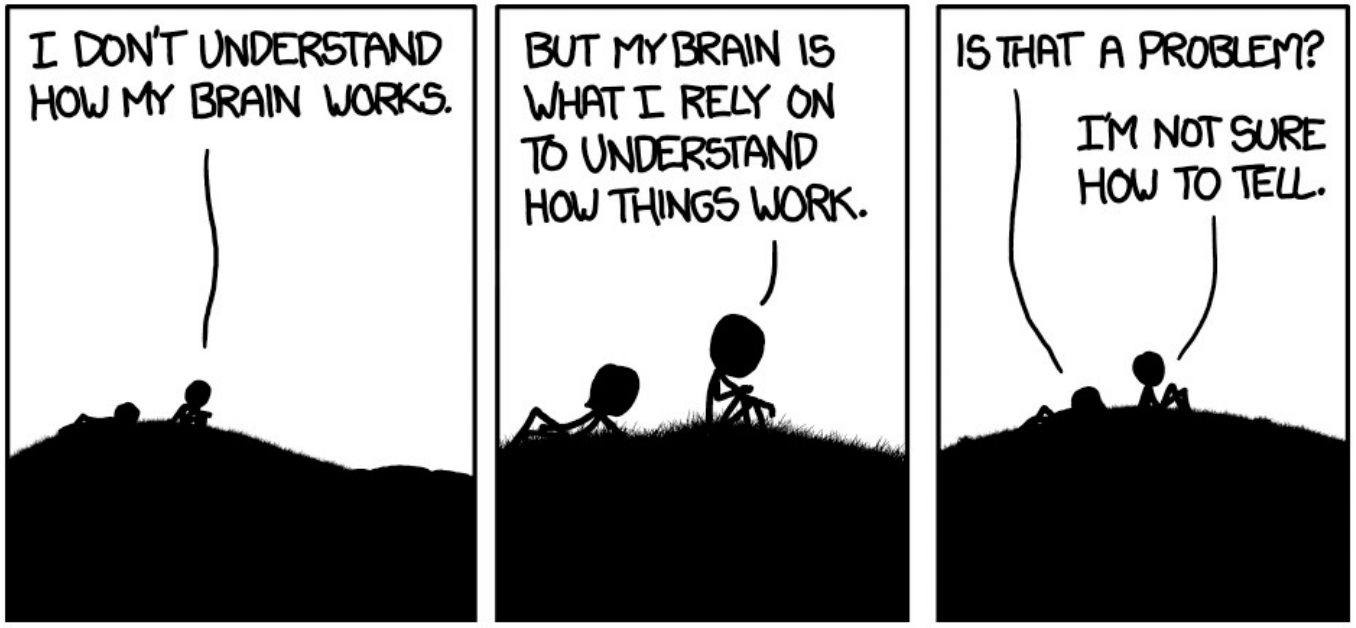
Debugging Minds







The Story So Far …

- We want to deliver and support a quality software product
- Software processes are carried out by humans
  - We can measure brain activity
  - Productivity and expertise involve different levels of brain activity
- Today: What neural activities are associated with SE and how can their measurements guide process decisions?

One-Slide Summary

- **Software measurement** can be misleading for process decisions involving humans.
- Many **complexity metrics** do not correlate with neural activity. Metrics based on data dependency do correlate with cognitive load.
- Context is seen as a positive factor for code comprehension, but more context *hurts* quality.
- We can control neural activity to probe causality and learn which measurements are valid. **Cognitive enhancements** (based on valid measurements) can impact software engineering outcomes.

Setting the Stage

- In this class, you've practiced and considered decisions for improving SE outcomes:
  - HW2: Which test generator tool … ?
  - HW4: Which static analysis tool … ?
  - HW5: Is delta debugging good for task X … ?
- We want more practice making critical decisions
  - How do we pick the good thing and not the fad
  - Especially when human cognition is involved!
  - We will begin with code comprehension

Complexity Metrics in Practice

- Lines of Code, McCabe's Cyclomatic Complexity, and Halstead Volume are **software metrics**
  - e.g., NASA (SWE-220) requires McCabe's <= 15 for safety critical modules
- These metrics often have validity problems
  - Can be misinterpreted or encourage wrong behaviors
  - We end up measuring what's easy, not what matters
- What is the underlying assumption here?
  - 'Higher Complexity Score = Harder for Humans to understand and thus to maintain' ???

What Metrics Measure

(Do Metrics Actually Measure How Hard it is to Read Code?)

- Studies suggest developers use strategies like **Top-down** or **Bottom-up comprehension** to understand code
  - We may use beacons and rely on our experience to ease code understanding
- *Core question:* do structural metrics like McCabe actually reflect the mental effort required for human comprehension?
- Let's see!

Study Overview

- Developers (n=19) read short code snippets, answered comprehension questions, and rated how difficult it was.
- Record cognitive effort (fMRI), response times, accuracy and subjective judgments of difficulty
- The 16 Java code snippets varied in
  - LOC, Vocabulary size (Halstead), Control flow (McCabe), Data-flow dependencies (**DepDegree 'NEW'**)

[Siegmund et al., "Measuring Neural Efficiency of Program Comprehension", ESEC/FSE 2017]

Finding #1: Code Size and Vocabulary Impact Cognitive Load

TABLE II: Kendall's τ and the explained variance (r^2, in brackets) of the dependent variables. A darker cell shading indicates a stronger correlation: none ($\tau < 0.1$), small ($0.1 < \tau < 0.3$), medium ($0.3 < \tau < 0.5$), and strong ($0.5 < \tau$) [49].

		Complexity Metrics			
		LOC	Halstead	McCabe	DepDegree
Complexity Metrics	LOC				
	Halstead	.32			
	McCabe	.57	.25		
	DepDegree	.59	.50	.49	
Activation	BA 6	.26 (.05)	.38 (.20)	.04 (.00)	.32 (.11)
	BA 21	.43 (.39)	.32 (.27)	.09 (.04)	.41 (.24)
	BA 39	.17 (.10)	.40 (.18)	.07 (.01)	.36 (.21)
	BA 44/45	.15 (.04)	.17 (.06)	−.04 (.00)	.22 (.09)
Deactivation	BA 31	−.30 (.21)	−.30 (.11)	.05 (.00)	−.24 (.04)
	BA 32	−.39 (.24)	−.42 (.22)	.04 (.00)	−.29 (.04)
Behavioral Data	Correctness	−.46 (.29)	−.45 (.26)	−.09 (.02)	−.41 (.22)
	Time	.22 (.10)	.24 (.09)	.06 (.00)	.26 (.07)

"Cognitive load"

SE

With your team, what patterns do you spot?

Finding #1: Code Size and Vocabulary Impact Cognitive Load

- LOC → BA 21 activation: $\tau = .32$
- Halstead → BA 6/21/39: $\tau = .32$–$.40$
- LOC & Halstead → DMN deactivation: $\tau = -.30$
- Meaning, longer or symbol-heavy code increases semantic processing and working memory demand
- Size matters because it affects mental workload, not because it contains more logic.

[Siegmund et al., "Measuring Neural Efficiency of Program Comprehension", ESEC/FSE 2017]

Explaining DepDegree

- **DepDegree** is a simple indicator of complex dependencies. The more dependencies a program operation has, the more different program states have to be considered "by the human" and the more difficult "for the human" it is to understand the operation.

With your team, which ones of these has a bug ?

Which is harder to understand?

```
int a, b;

void swap ()  {
      a += b;
      b = a – b;
      a – = b;
}
```

```
int a, b;

void swap ()  {
      int temp = a;
      a = b;
      b = temp;
}
```

Explaining DepDegree

- **DepDegree** is a simple indicator of complex dependencies. The more dependencies a program operation has, the more different program states have to be considered "by the human" and the more difficult "for the human" it is to understand the operation.

What scores would these methods get in other metrics?

LOC = 3
McCabe = 1

```
int a, b;

void swap ()  {
      a += b;
      b = a – b;
      a – = b;
}
```

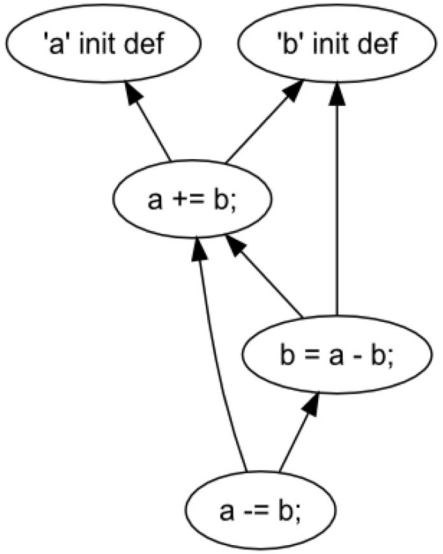
```
int a, b;

void swap ()  {
      int temp = a;
      a = b;
      b = temp;
}
```

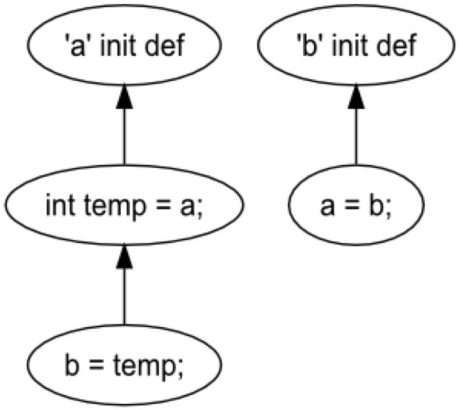
LOC = 3
McCabe = 1

Explain ‘DepDegree’

```
int a, b;

void swap ()  {
      a += b;
      b = a – b;
      a – = b;
}
```

```
int a, b;

void swap ()  {
      int temp = a;
      a = b;
      b = temp;
}
```



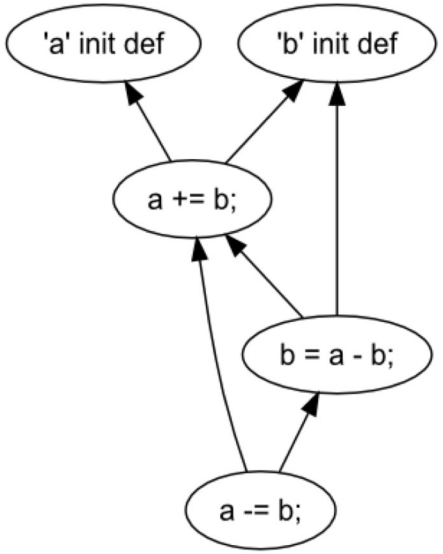


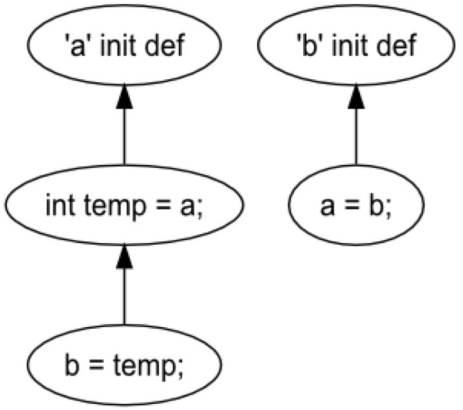
Explain ‘DepDegree’

```
int a, b;

void swap ()  {
      a += b;
      b = a – b;
      a – = b;
}
```

```
int a, b;

void swap ()  {
      int temp = a;
      a = b;
      b = temp;
}
```

Which data flow analysis determines if a variable will be used later (or is dead)?

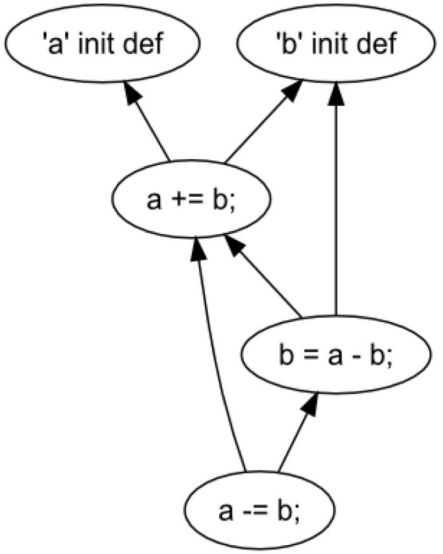




Explain DepDegree

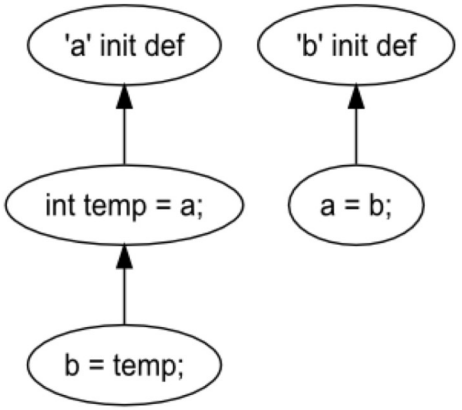
```
int a, b;

void swap ()  {
      a += b;
      b = a – b;
      a – = b;
}
```

LOC = 3
McCabe = 1
DepDegree = 6

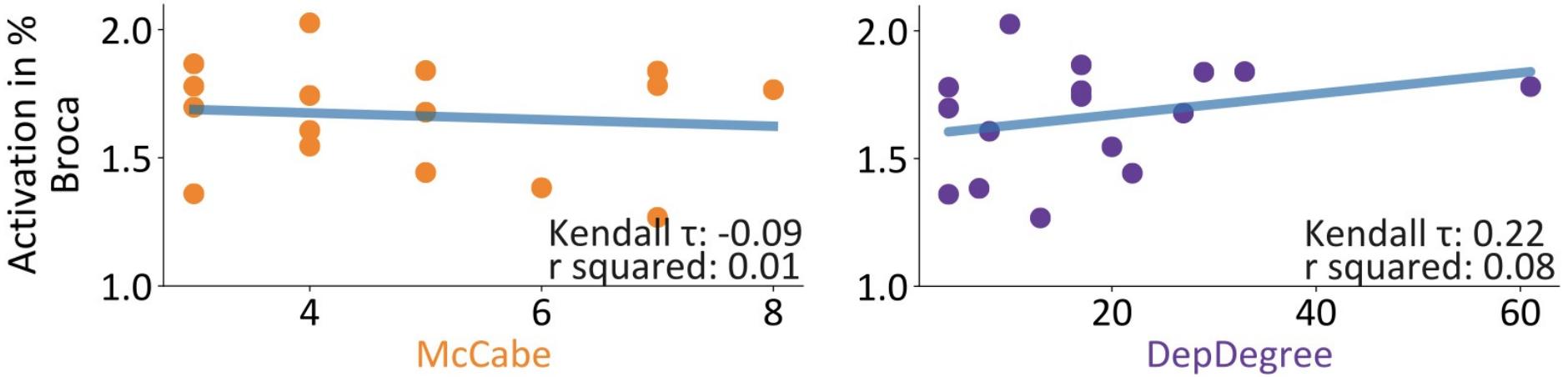



```
int a, b;

void swap ()  {
      int temp = a;
      a = b;
      b = temp;
}
```

LOC = 3
McCabe = 1
DepDegree = 3



Finding #2: Data flow complexity makes comprehension hard



- Working memory and semantic processing have a medium-strong correlation with DepDegree

  (vs. no correlation with McCabe …)

- Interpretation: tracking state and variable dependencies is hard for humans!

Prior Lecture

- [Code Inspection and The Brain Lecture]
- Do you think it would take you longer to maintain code if it looked like this?

Listing 2: Code snippet with no beacons and disrupted layout (*BN, LD*)

```
1  public float ayyaoAwyyaky(int[] array) {
2     int
3           mgqakyy
4     = 0;
5     int sum = 0;
6
7     while (mgqakyy
8           < array.length) {
9        sum =
10       sum + array[mgqakyy];
11       mgqakyy
12       = mgqakyy + 1;
13    }
14
15    float average
16          = sum /
17          (float) mgqakyy;
18    return
19          average;
20 }
```

Prior Lecture

- [Code Inspection and The Brain Lecture]
- Do you think it would take you longer to maintain code if it looked like this?
  - Perhaps counter-intuitively: no!

Listing 2: Code snippet with no beacons and disrupted layout (*BN, LD*)

```
1  public float ayyaoAwyyaky(int[] array) {
2      int
3              mgqakyy
4      = 0;
5      int sum = 0;
6
7      while (mgqakyy
8              < array.length) {
9          sum =
10         sum + array[mgqakyy];
11         mgqakyy
12         = mgqakyy + 1;
13     }
14
15     float average
16             = sum /
17             (float) mgqakyy;
18     return
19             average;
20 }
```

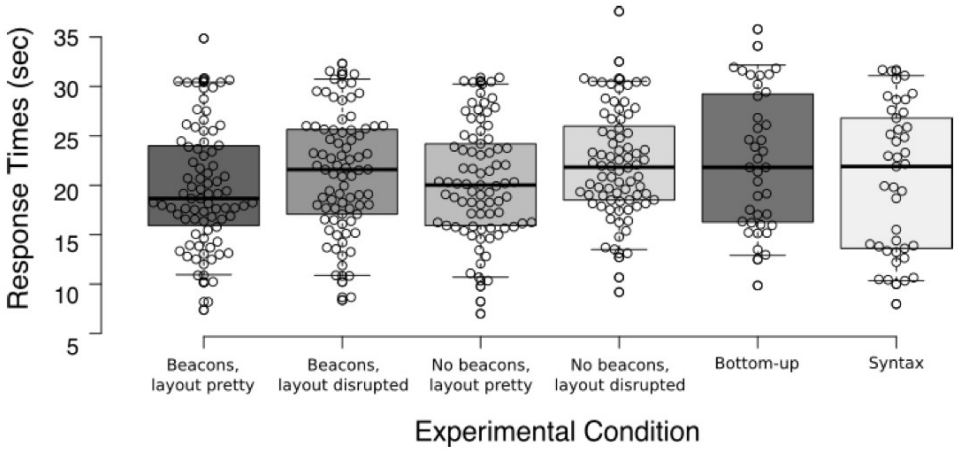


Figure 4: Response times in seconds per condition.

When can we trust self-reporting?

(In what areas do we not trust self-reporting?)

- ~~How good you are at something (self-reported efficacy)~~ [Productivity Lecture] (Humans aren't good at judging self-expertise)
- ~~What makes code hard to read (presence of comments vs blank lines)~~ [Design for Maintainability Lecture] (normative vs. descriptive metrics)
- ~~How long will it take me to comprehend code~~
- How hard is it to comprehend code (brain activity) [Code Inspection and the Brain Lecture]

- This paper also finds developers' own feelings of difficulty to correlate with correctness and mental effort.
  - Subjective rating → correctness: $\tau = -.77$ (strong)
  - Subjective rating → cognitive load: $\tau = -.69$

Finding #3: Subjective complexity predicts performance

- Developers' own feelings of difficulty correlated strongly with correctness and mental effort.
- We don't have evidence that developers know how hard something will be in the future, **but** they can more confidently say how hard something is currently
- Takeaways for future devs and team leads:
  - Self-reporting about future scheduling may be less reliable (unlike COCOMO, which *can* predict costs). Self-reporting for current difficulties is OK.
  - Use the right information source when making process decisions involving humans

Example Industry Use: SonarQube

USER GUIDE > MONITORING CODE METRICS

Ask

Understanding measures and metrics

Measures and metrics used in SonarQube to evaluate your code.

Metrics are used to measure:

- Security, maintainability, and reliability attributes on the basis of statistics on the detected security, maintainability, and reliability issues, respectively.
- Test coverage on the basis of coverage statistics on executable lines and evaluated conditions.
- Code cyclomatic and cognitive complexities.
- Security review level on the basis of statistics on reviewed security hotspots.

Complexity

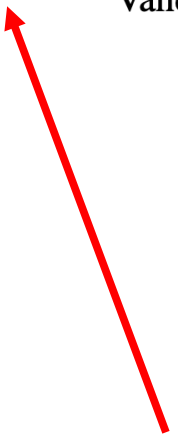
Complexity metrics used in the Sonar solution.

Metric	Metric key	Definition
Cyclomatic complexity	`complexity`	A quantitative metric used to calculate the number of paths through the code.
Cognitive complexity	`cognitive_complexity`	A qualification of how hard it is to understand the code's control flow. See the Cognitive Complexity white paper ↗ for a complete description of the mathematical model applied to compute this measure.

Next Paper ...

Programmers' Visual Attention on Function Call Graphs During Code Summarization

Samantha McLoughlin*, Zachary Karas*, Robert Wallace†, Aakash Bansal‡, Collin McMillan†, Yu Huang*,
*Vanderbilt University: {samantha.m.mcloughlin, z.karas, yu.huang}@vanderbilt.edu
†University of Notre Dame {rwallac1, cmc}@nd.edu
‡Louisiana State University abansal@lsu.edu



Going Beyond Beacons

(Here's the Motivation)

- Code features like **beacons** (semantic cues, method names) impact code comprehension
- But developers might also want to know:
  - What calls this method? (**caller context**)
  - What does this method call? (**callee context**)
  - Static and dynamic analysis represent these as a **Call Graph**, but an explicit graph is not usually shown to developers (cf. *gprof* paper)
- It seems intuitive that this context would help
  - 'More context = better code comprehension' **???**

Let's Investigate Context and Comprehension

- To make this tractable, let's focus on one software maintenance task: **code summarization**
  - HW6 → Figuring out what is going on in the project
- Does exploring calling contexts really help developer summarize code?
- Many would assume "yes". Developers …
  - Might navigate between callers and callees to gather details (?)
  - Might gain info from reading nearby methods (?)
- Lets find out …

Study Overview

- *Question:* How do developers use caller/callee context during code summarization, and does exploring this context help or hurt the resulting code summary?
- Developers (n=22) wrote code summaries for methods in a Java project via the Eclipse IDE
  - Developers saw source code (not nodes/edges)
- Eye-tracking recorded their behavior

[McLoughlin et al., "Programmers' Visual Attention on Function Call Graphs During Code Summarization", ASE 2025]

Main Finding: Call-Graph Coverage Leads to lower quality summaries

- The more methods they looked at that were callers/callees → *worse* summary quality
  - Effect was stronger for callee than caller
- Why? Perhaps deep callee exploration can break abstraction and modularity [Design for Maintainability Lecture]
- Looking at more caller/callee functions was also associated with decreased self confidence in summary quality

Implications

- Developers don't benefit from exploring large amounts of caller/callee context when it comes to comprehension
  - Developers perform worse when having to consider large contexts.

- Example Developer/Team Lead consideration:
  - How much context and information does a new developer you are onboarding actually need?

Making Informed Process Decisions

- Today, we have discussed three challenges:
  - We assumed code complexity reflects difficulty
    → but complexity metrics don’t
  - We assume self-reporting helps
    → but it doesn't for *future* predictions
  - We assumed more context helps comprehension
    → but caller/callee context doesn’t

Making Informed Process Decisions

- Today, we have discussed three challenges:
  - We assumed code complexity reflects difficulty?
  - We assume self-reporting helps?
  - We assumed more context helps comprehension?
- Informally, the problem is that a lot of these ideas/metrics are like the "Blanks lines help readability" claim
  - Blank lines do *correlate* with readability in current code, but adding many blank lines doesn't *cause* something to be more readable in the future

Making Informed Process Decisions

- Blank lines do *correlate* with readability in current code, but adding many blank lines doesn't *cause* something to be more readable in the future
- To tease apart correlation and causation, we use a controlled experiment and manipulate the presence or absence of a relevant feature
  - e.g., add or remove blank lines from code
- What if we want to look at causation within the brain?
- How do we add or remove "brain activity" (like working memory) from a human?

Trivia: Medieval History

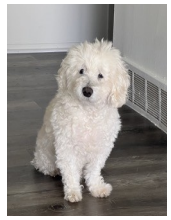
- This Greek-speaking descendant of the Roman Empire centered around Istanbul (was Constantinople) and conquered much of the Mediterranean coast. Greek fire, mosaics, orthodox Christianity, the crusades, and the Hagia Sophia are all associated with this empire.

Trivia: Cuisine

- This salty, crumbly cheese is from the highlands of Michoacán, Mexico. Made from cow's milk and aged to develop a sharp flavor, it softens with heat but does not melt. It is typically grated or crumbled over dishes such as elotes, tacos, soups, and salads, and is widely used across Mexico.

Trivia: Games

- This cheerful, dinosaur-like companion first appeared in a classic platforming game and is known for funny jumps, swallowing foes, and producing useful eggs. It appears across many adventure and racing titles in the franchise.



Trivia: Fashion

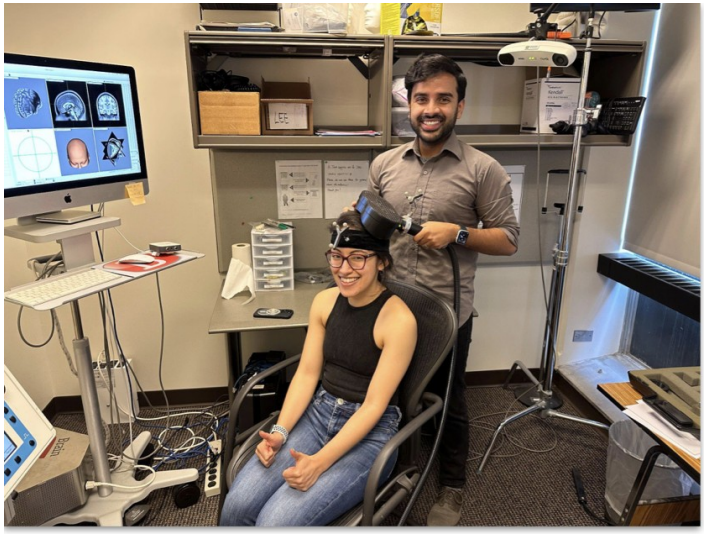
- These types pants were first used as military trousers from the middle of the 19th century onward. These trousers are made from “twill”, a classic cotton fabric that is described as being naturally light.

Experimenting on Software Cognition

- Many SE theories assume certain cognitive processes (e.g., working memory, language) affect how developers understand code
- But most prior studies only observe the brain, not manipulate it
- To know whether a cognitive factor *causes* performance changes, we need a way to manipulate that cognitive factor
- If we could do so, it would allow us to ask:
  - "If we nudge this brain system, does the developer perform differently?"

Enter: Transcranial Magnetic Stimulation

- **Transcranial Magnetic stimulation** is a noninvasive method that uses brief magnetic pulses on the scalp to temporarily increase or decrease activity in a specific brain region
- It lets us manipulate cognitive regions (e.g., working memory), then measure whether performance changes.



- [Ahmad, et al., "Causal Relationships and Programming Outcomes: A Transcranial Magnetic Stimulation Experiment", ICSE 2024]

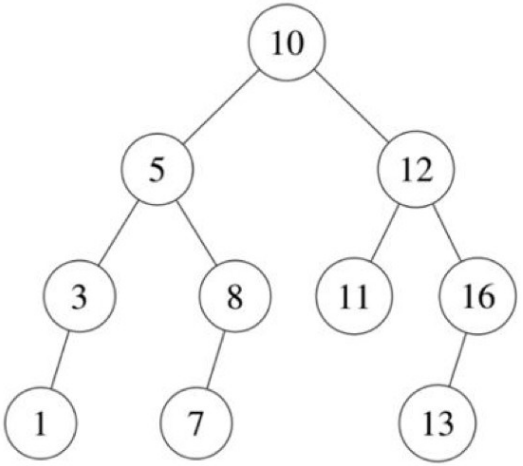
Introducing TMS Part 2

- TMS is a safe technique that is used in medical contexts
  - For example, Mayo Clinic → depression
- TMS has also been used to investigate causality in other fields:
  - Math: TMS resulted in 30% accuracy increase in memorization and addition of numbers [1]
  - Language: TMS resulted in faster verb processing speed for manual actions [2]
- Effect of TMS is temporary (45-60 min)

[1] J. Gill et al. It's the thought that counts: Examining the task-dependent effects of transcranial direct current stimulation on executive function. 2015. *[2]* R. M. Willems et al. A functional role for the motor system in language understanding: Evidence from theta-burst transcranial magnetic stimulation. 2011.

What are we going to study?
(SE Data Structure Problem)

Consider the AVL tree below. After inserting 14 into the tree (and performing rotations to keep the tree balanced as necessary), which of the following will be produced by a pre-order traversal of the resulting tree?



A: 10, 5, 3, 1, 8, 7, 12, 11, 14, 13, 16

B: 10, 5, 3, 1, 8, 7, 12, 11, 16, 13, 14

What are we going to study?
(SE Code Comprehension Problem)

What is true about the following code for reversing a vector?
Assume that n = *v*.size().

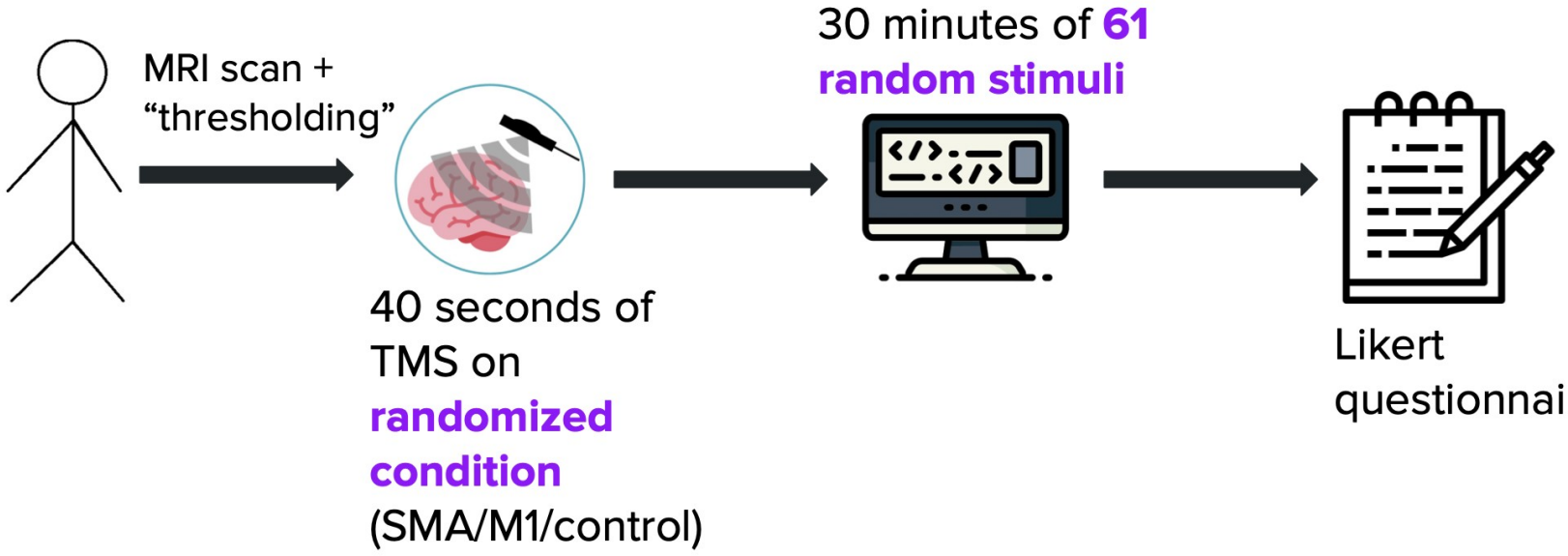
```
vector<int> v{1, 2, 3, 4};
stack<int> s;

for (size_t i = 0; i < v.size(); ++i)
    s.push(v[i]);

for (size_t i = 0; i < v.size(); ++i) {
    v[i] = s.top();
    s.pop();
}  // for ..i
```

A: The memory complexity is O(1).

B: The memory complexity is O(n).

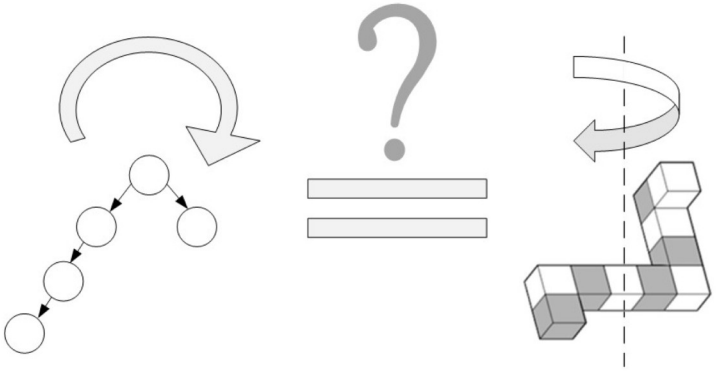
Setup TMS Experiment for SE

- Programmers (n=16) received 3 different sessions of TMS on three different days
- Record outcomes like timing (time taken)



Finding #1: ???

- Prior work found a correlation between mental rotation (spatial reasoning) and data structure manipulation.

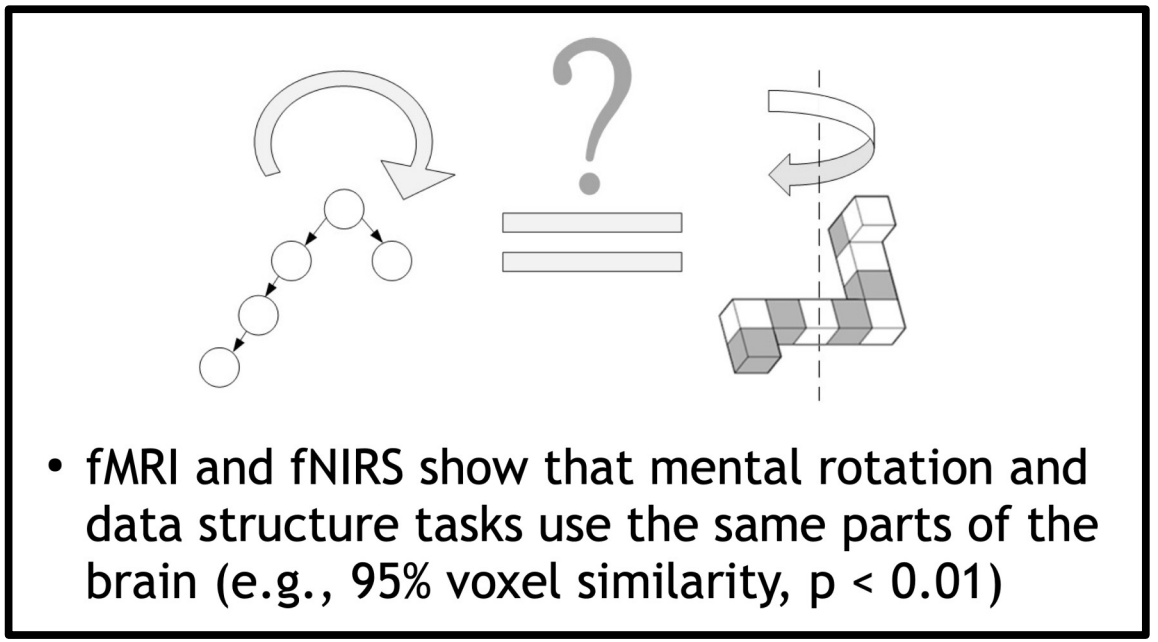


- fMRI and fNIRS show that mental rotation and data structure tasks use the same parts of the brain (e.g., 95% voxel similarity, $p < 0.01$)

- Should SE managers train spatial reasoning to help devs with data structures?

Finding #1: Causal Relationships

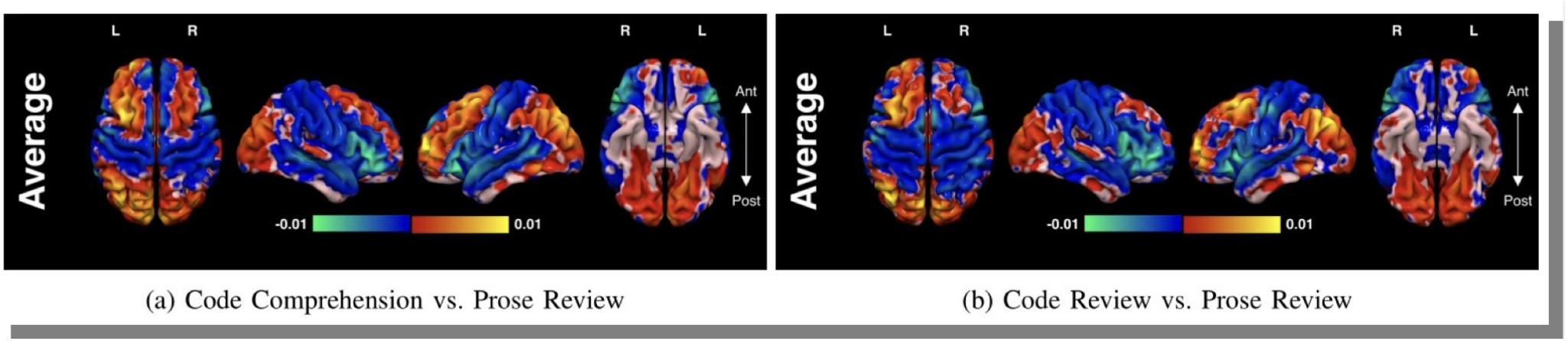
- Prior work found a *correlation* between mental rotation (spatial reasoning) and data structure manipulation.



- We found **no** significant direct *causal* relationship between spatial reasoning and DS outcomes

Finding #2: ???

- Researchers had found a correlation between language and code comprehension [Code Inspection and the Brain Lecture]



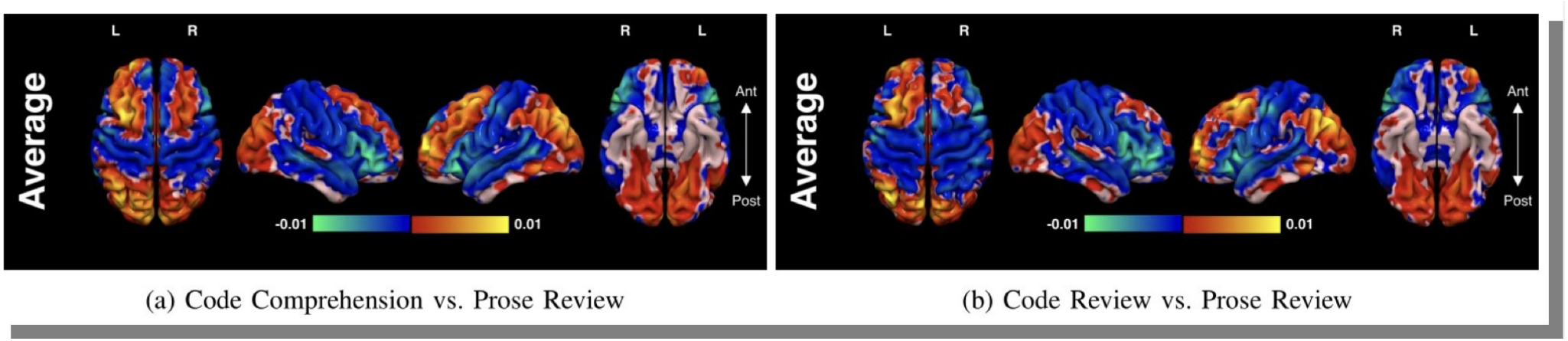
(a) Code Comprehension vs. Prose Review

(b) Code Review vs. Prose Review

- Should SE managers consider cognitive trainings to help devs with code comprehension?

Finding #2: Causal Relationships

- Researchers had found a correlation between language and code comprehension [Code Inspection and the Brain Lecture]



(a) Code Comprehension vs. Prose Review

(b) Code Review vs. Prose Review

- Potentially Yes!
  - (Let's dive in …)

Finding #2: Causal Relationships

- TMS *can* affect programming tasks
  - Which factors matter more for time taken by software developers (task completion time)?

Factor	Effect Size (Normalized)
"How hard is the question?"	1.00
"Participant expertise"	0.18
"TMS"	0.05

- We found that the 'participant by brain region stimulated' factor significantly accounted for 2.2% of the variance in the response time when controlling for other plausible effects.

Is This a Big Effect?

- A recurring theme in class is deciding between two process decisions
  - If pair programming reduced the defect rate by X% but increases coding time by Y% amount of time ...
  - If design for maintainability increases coding time by A% but decreases maintenance time by B% …
- These percentages don't have to be big for us to want to implement such decisions
  - e.g., modest pair programming benefits can add up if debugging is more expensive than code writing
- Is 2.2% coding time decrease via TMS a lot?

Takeaway: It Depends!

- 2.2% variance is a small, indirect effect …
- But for the future, TMS:
  - Can be done alongside other approaches
  - Doesn’t require shared language
  - Only takes five minutes

- 481: We show you this because we suspect cognitive interventions will be increasingly deployed in the next decade or two (cf. AI now)
  - You'll be more senior. How will you decide?

Real World Use of SE + Cognition

- Suppose you are manager, and you are going to assign your developers an extra hour of training each week for 11 weeks: what do you do?
  - Do you have them train spatial ability (shown to *correlate* with SE tasks but no causal relationship was found)?

    OR

  - Do you have them train language ability (shown to correlate with SE tasks AND a causal relationship was found)?
- Let's try it!

Real World Interpretation?

- UM EECS 183 Students (n=57) were randomly divided into two training groups (spatial training vs language training)
  - Attended one hour training sessions each week for 11 weeks, while taking EECS183 as normal
  - At the end the semester, they were given a “final exam”
- Even when controlling for differences in incoming preparation (e.g., CS knowledge) ...

  How did the two groups do?

Real World Use of SE + Cognition

- Suppose you are manager, and you are going to assign your developers an extra hour of training each week for 11 weeks: what do you do?

and spatial ability correlate with programming success. Significantly, however, we find that those in our reading training exhibit larger programming ability gains than those in the standard spatial training ($p = 0.02$, $f^2 = 0.10$). We also find that reading trained participants perform particularly well on programming problems that require tracing through code ($p = 0.03$, $f^2 = 0.10$). Our results suggest that technical reading training could be beneficial for novice programmers. Finally, we discuss the implications of

- [Endres, et al., "To Read or To Rotate? Comparing the Effects of Technical Reading Training and Spatial Skills Training on Novice Programming Ability", FSE 2021]

Overall Conclusion

- As a software engineer lead or manager, you will have to make critical decisions about humans
  - Classic question: Should I invest more in testing?
  - Current modern question: Should I invest more in AI tools?
  - Future question(?): Should I invest more in cognitive enhancements?
- These feel different on the surface, but they are all the same sort of question, so use the same tools/reasoning you've learned in 481
  - Avoid cognitive biases (streetlight effect and McCabe)
  - Use valid measurements (self-reporting and COCOMO)
  - Avoid mistaking correlation and causation

Questions?

- Focus on HW6