Problem Solving

CS 695 / SWE 699: Programming Tools
Fall 2023
Today

• Part 1 (Lecture)(~85 mins)
• Break!

• Part 2 (In-Class Activity)(45 mins)

• Part 3 (Project group work)(20 mins)
  • Time to work in groups, ask questions
Logistics

• HW1 due next week
  • Project direction will evolve over time - that's ok!

• Anyone still looking for a group?

• Anyone who has not yet signed up for a Tech Talk?
A few minutes in the life of a developer
A few hours in the life of a professional software developer

<table>
<thead>
<tr>
<th>collaboration</th>
<th>Developer assigned bug by team</th>
</tr>
</thead>
<tbody>
<tr>
<td>programming</td>
<td>Reproduces error</td>
</tr>
<tr>
<td></td>
<td>Browser hits error message (500 internal error)</td>
</tr>
<tr>
<td></td>
<td>Attaches debugger</td>
</tr>
<tr>
<td></td>
<td>Browse to page again, hit null reference exception</td>
</tr>
<tr>
<td></td>
<td>Hypothesize from call stack which function might be responsible</td>
</tr>
<tr>
<td></td>
<td>Browse through code</td>
</tr>
<tr>
<td></td>
<td>Uses debugger to change values &amp; experiment</td>
</tr>
<tr>
<td></td>
<td>Make change, recompile, check, doesn’t work</td>
</tr>
<tr>
<td></td>
<td>Navigates slice, wrong values came from objects</td>
</tr>
<tr>
<td></td>
<td>In complicated code doesn’t understand</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>collaboration</th>
<th>Walks to B’s office and asks where data comes from</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B working on high profile feature in area</td>
</tr>
<tr>
<td>programming</td>
<td>Tries to make change, still doesn’t work</td>
</tr>
<tr>
<td>collaboration</td>
<td>Walks back to B, realize related to C’s feature, C at lunch</td>
</tr>
<tr>
<td></td>
<td>After lunch, A and B walk to C’s office</td>
</tr>
<tr>
<td>design</td>
<td>A, B, C change design to work with new feature</td>
</tr>
<tr>
<td>collaboration</td>
<td>Bug passed from A to C to change feature</td>
</tr>
</tbody>
</table>

Problem solving

**Goal**: where am I trying to go?
**Operators**: what actions can I take to get closer to the goal?

Apply operator, look at new state, apply another operator

Problem solving is recursive

<table>
<thead>
<tr>
<th>task</th>
<th>Investigate and fix a design problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>question</td>
<td>Why is an event being issued by forcing a cache update?</td>
</tr>
<tr>
<td></td>
<td>How is BufferHandler using its buffer field? Are there any other mutations on it?</td>
</tr>
<tr>
<td></td>
<td>Read methods of BufferHandler</td>
</tr>
<tr>
<td></td>
<td>Why is there a buffer member variable that is never used?</td>
</tr>
<tr>
<td></td>
<td>Investigate references to BufferHandler.buffer</td>
</tr>
<tr>
<td></td>
<td>Why is doDelayedUpdate() a member of BufferHandler?</td>
</tr>
<tr>
<td></td>
<td>Reads methods along path, concludes that BufferHandler tracks update delays</td>
</tr>
<tr>
<td></td>
<td>Why wouldn’t isFoldStart() call getFoldLevel()</td>
</tr>
<tr>
<td></td>
<td>Reads isFoldStart(), getFoldAtLine()</td>
</tr>
<tr>
<td></td>
<td>Concludes isFoldStart() doesn’t call because of short circuit evaluation</td>
</tr>
<tr>
<td>action</td>
<td>Implement fix</td>
</tr>
<tr>
<td></td>
<td>Assure correctness</td>
</tr>
<tr>
<td></td>
<td>Set conditional break point</td>
</tr>
<tr>
<td></td>
<td>Check that jEdit still appears to work correctly</td>
</tr>
<tr>
<td></td>
<td>Repro original bug by reinserting</td>
</tr>
</tbody>
</table>

LaToza and Myers. Designing useful tools for developers. PLATEAU 2011.
Problem solving is recursive
Problem solving involves answering questions

Goal: Fix issue

- Where is the defect?
  - Subgoal
    - Subgoal
      - Subgoal
        - Subgoal

- Which function generated the incorrect output?
  - Subgoal
    - Subgoal
      - Subgoal
        - Subgoal

- Where is this function invoked?
Problem solving involves strategies

Goal: Fix issue

- Where is the defect?
- Trace output backwards
- Which function generated the incorrect output?

Subgoal

Subgoal

Subgoal

Subgoal
Problem solving involves taking actions to answer questions and follow strategies

Goal: Fix issue

Where is the defect?

Trace output backwards

Which function generated the incorrect output?
Developers use a variety of techniques for obtaining information and answering questions.

Problem solving involves formulating hypotheses

Goal: Fix issue

Where is the defect?

Maybe it's in the registration code I just wrote

Is the registration code executing correctly?
Problem solving involves choices between strategies

Goal: Fix issue

Where is the defect?

Maybe it's in the registration code I just wrote

Trace output backwards

Is the registration code executing correctly?

Which function generated the incorrect output?
Problem solving in programming

• Developers have **tasks** (e.g., fix this defect, implement this feature) which they work to complete.

• Developers ask **questions** reflecting information they need in order to complete tasks.
  • e.g., What’s the best design for implementing this?

• Developers may formulate **hypotheses** representing answers to questions.

• Developers select **strategies** to gather evidence answer questions and to support or reject hypotheses.
  • From code, from external resources, from teammates

• Developers often have multiple strategies to answer questions.
Program comprehension as fact finding

- **SEEK**
  - Read relevant methods looking for facts

- **CRITIQUE**
  - Fact A is bad design

- **LEARN**
  - Fact A is true

- **EXPLAIN**
  - Fact A is true to make fact B true

- **PROPOSE**
  - Change facts A1, B1 to facts A2, B2

- **IMPLEMENT**
  - Change code to reflect facts A2, B2

LaToza, Garlan, Herbsleb, Myers. Program comprehension as fact finding. FSE 07.
Supporting programming activities

Goal: Fix issue

Where is the defect?

Maybe it's in the registration code I just wrote

Is the registration code executing

Trace output backwards

Which function generated the incorrect output?

• Many potential points of intervention, supporting subgoals / strategies / question answering / testing hypotheses
Useful interventions solve important problems

LaToza and Myers. Designing useful tools for developers. PLATEAU 2011.
What percentage of the last week have you spent...

Example: Activities in fixing a defect

Circle size: % of time
Edge thickness: % of transitions observed

For tasks in code in your own codebase that you haven’t seen recently

LaToza and Myers. Developers ask reachability questions. ICSE 2010.
Figure 2. The backgrounds and task structures of the 17 observed developers. The right edge of each task block indicates the reason for the task switch (thin line for done, thick line for blocked, jagged line for interrupted). When a task gets broken up by interruptions or blocking, we draw its fragments at the same vertical level to show resumption.
Some methods for supporting problem solving

• Find an important question, build tool that makes it easier to answer

• Find an action that helps developers answer questions, make it easier to take

• Find a new strategy that helps developers answer question more effectively
Many other factors influence difficulty answering questions

- expertise
- development environments
- programming languages
- code quality
- team practices
- time to market
- software quality

• Interventions might also target these factors
Some methods for supporting problem solving

• Find an important question, build tool that makes it easier to answer

• Find an action that helps developers answer questions, make it easier to take

• Find a new strategy that helps developers answer question more effectively
Making questions easier to answer

• Tools help developers be more productive by reducing the time to answer questions, increasing likelihood of success

• This requires
  • understanding precisely the information required and context available to developers
  • insight into a mechanism to make a question easier to answer
Example: Questions about object structure

Is a
Who implements type X? [who can be an object or a type]

Navigability
Let's say I am in the StandardDrawing class and I want the JavaDrawApp object which is a DrawingEditor [...]. What would save me a lot of time is to say now I am at the Drawing and I want to go to the DrawingEditor, show me my options.

Part of
Maybe I would start with the Drawing object and that should have a list of listeners?

How to get
How I will get hold of the DrawingEditor object? [...] Basically I need to know the instance of the current window.

I know I need to get the view from here; so how do I do that?

Is in tier
What I would be interested in is looking in the code to try to understand where are the view and model

Cardinality
The class diagram says that the DrawingEditor has one DrawingView and the StandardDrawingView may or may not have a Drawing.

I would like to know the cardinality: so Window has one or more StandardDrawingViews?

Is owned
Maybe I would start with the Drawing object and that should have a list of listeners

May alias
[...] the window itself has a reference to the UndoManager but you can’t tell from this diagram whether each window has its own UndoManager, or whether it is just one global manager.

May (not) alias
So I have different selections in the different views.

Points to
Both of them are two views on the same Drawing, but if there are two windows...

The class diagram says that the DrawingEditor has one DrawingView and the StandardDrawingView may or may not have a Drawing.

I would like to know the cardinality: so Window has one or more StandardDrawingViews?

Let's say I am in the StandardDrawing class and I want the JavaDrawApp object which is a DrawingEditor [...]. What would save me a lot of time is to say now I am at the Drawing and I want to go to the DrawingEditor, show me my options.

How I will get hold of the DrawingEditor object? [...] Basically I need to know the instance of the current window.

I know I need to get the view from here; so how do I do that?

What I would be interested in is looking in the code to try to understand where are the view and model

I would like to know the cardinality: so Window has one or more StandardDrawingViews?

[...] the window itself has a reference to the UndoManager but you can’t tell from this diagram whether each window has its own UndoManager, or whether it is just one global manager.

So I have different selections in the different views.

Both of them are two views on the same Drawing, but if there are two windows...
## Example: Programming questions

<table>
<thead>
<tr>
<th>Information type</th>
<th>Search times</th>
<th>% Agreed info is...</th>
<th>Frequency and outcome of searches</th>
<th>Frequency of sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>mid</td>
<td>max</td>
<td></td>
</tr>
<tr>
<td>s1  Did I make any mistakes in my new code?</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>a2  What have my coworkers been doing?</td>
<td>0</td>
<td>11</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>u3  What code caused this program state?</td>
<td>0</td>
<td>2</td>
<td>21</td>
<td>90</td>
</tr>
<tr>
<td>r2  In what situations does this failure occur?</td>
<td>0</td>
<td>2</td>
<td>49</td>
<td>80</td>
</tr>
<tr>
<td>d2  What is the program supposed to do?</td>
<td>0</td>
<td>1</td>
<td>21</td>
<td>93</td>
</tr>
<tr>
<td>a1  How have resources I depend on changed?</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>41</td>
</tr>
<tr>
<td>u  What code could have caused this behavior?</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>73</td>
</tr>
<tr>
<td>c2  How do I use this data structure or function?</td>
<td>0</td>
<td>1</td>
<td>14</td>
<td>71</td>
</tr>
<tr>
<td>d3  Why was this code implemented this way?</td>
<td>0</td>
<td>2</td>
<td>21</td>
<td>61</td>
</tr>
<tr>
<td>b3  Is this problem worth fixing?</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>44</td>
</tr>
<tr>
<td>d4  What are the implications of this change?</td>
<td>0</td>
<td>2</td>
<td>9</td>
<td>85</td>
</tr>
<tr>
<td>d1  What is the purpose of this code?</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>56</td>
</tr>
<tr>
<td>b2  What's statically related to this code?</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>66</td>
</tr>
<tr>
<td>s2  Did I follow my team's conventions?</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>49</td>
</tr>
<tr>
<td>r1  What does the failure look like?</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>88</td>
</tr>
<tr>
<td>s3  Which changes are part of this submission?</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>61</td>
</tr>
<tr>
<td>c3  How can I coordinate this with this other code?</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>75</td>
</tr>
<tr>
<td>b1  Is this a legitimate problem?</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>88</td>
</tr>
<tr>
<td>r3  Why are these changes necessary?</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>61</td>
</tr>
<tr>
<td>c1  What is the purpose of this code?</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>This is a serious problem for me</th>
<th>% agree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code Understanding</strong></td>
<td></td>
</tr>
<tr>
<td>Understanding the rationale behind a piece of code</td>
<td>66%</td>
</tr>
<tr>
<td>Understanding code that someone else wrote</td>
<td>56%</td>
</tr>
<tr>
<td>Understanding the history of a piece of code</td>
<td>51%</td>
</tr>
<tr>
<td>Understanding code that I wrote a while ago</td>
<td>17%</td>
</tr>
<tr>
<td><strong>Task Switching</strong></td>
<td></td>
</tr>
<tr>
<td>Having to switch tasks often because of requests from my teammates or manager</td>
<td>62%</td>
</tr>
<tr>
<td>Having to switch tasks because my current task gets blocked</td>
<td>50%</td>
</tr>
<tr>
<td><strong>Modularity</strong></td>
<td></td>
</tr>
<tr>
<td>Being aware of changes to code elsewhere that impact my code</td>
<td>61%</td>
</tr>
<tr>
<td>Understanding the impact of changes I make on code elsewhere</td>
<td>55%</td>
</tr>
<tr>
<td><strong>Links between Artifacts</strong></td>
<td></td>
</tr>
<tr>
<td>Finding all the places code has been duplicated</td>
<td>59%</td>
</tr>
<tr>
<td>Understanding who “owns” a piece of code</td>
<td>50%</td>
</tr>
</tbody>
</table>
Questions developers report as hard to answer span many topics

Rationale (42)
Why was it done this way? (14) [15][7]
Why wasn’t it done this other way? (15)
Was this intentional, accidental, or a hack? (9)[15]
How did this ever work? (4)

Debugging (26)
How did this runtime state occur? (12) [15]
What runtime state changed when this executed? (2)
Where was this variable last changed? (1)
How is this object different from that object? (1)
Why didn’t this happen? (3)
How do I debug this bug in this environment? (3)
In what circumstances does this bug occur? (3) [15]
Which team’s component caused this bug? (1)

Intent and Implementation (32)
What is the intent of this code? (12) [15]
What does this do (6) in this case (10)? (16) [24]
How does it implement this behavior? (4) [24]

Refactoring (25)
Is there functionality or code that could be refactored? (4)
Is the existing design a good design? (2)
Is it possible to refactor this? (9)
How can I refactor this (2) without breaking existing users? (9)
Should I refactor this? (1)
Are the benefits of this refactoring worth the time investment? (3)

History (23)
When, how, by whom, and why was this code changed or inserted? (13) [7]
What else changed when this code was changed or inserted? (2)
How has it changed over time? (4)[7]
Has this code always been this way? (2)
What recent changes have been made? (1) [15][7]
Have changes in another branch been integrated into this branch? (1)

Implications (21)
What are the implications of this change for (5) API clients (5), security (3), concurrency (3), performance (2), platforms (1), tests (1), or obfuscation (1)? (21) [15][24]

Testing (20)
Is this code correct? (6) [15]
How can I test this code or functionality? (9)
Is this tested? (3)
Is the test or code responsible for this test failure? (1)
Is the documentation wrong, or is the code wrong? (1)

Implementing (19)
How do I implement this (8), given this constraint (2)? (10)
Which function or object should I pick? (2)
What’s the best design for implementing this? (7)

Control flow (19)
In what situations or user scenarios is this called? (3) [15][24]
What parameter values does each situation pass to this method? (1)
What parameter values could lead to this case? (1)
What are the possible actual methods called by dynamic dispatch here? (6)
How do calls flow across process boundaries? (1)
How many recursive calls happen during this operation? (1)
Is this method or code path called frequently, or is it dead? (4)
What throws this exception? (1)
What is catching this exception? (1)

Contracts (17)
What assumptions about preconditions does this code make? (5)
What assumptions about post(pre)conditions can be made? (2)
What exceptions or errors can this method generate? (2)
What are the constraints on or normal values of this variable? (2)
What is the correct order for calling these methods or initializing these objects? (2)
What is responsible for updating this field? (1)

Performance (16)
What is the performance of this code (5) on a large, real dataset (3)? (8)
Which part of this code takes the most time? (4)
Can this method have high stack consumption from recursion? (1)
How big is this in memory? (2)
How many of these objects get created? (1)

Teammates (16)
Who is the owner or expert for this code? (3) [7]
How do I convince my teammates to do this the “right way”? (12)
Did my teammates do this? (1)

Policies (15)
What is the policy for doing this? (10) [24]
Is this the correct policy for doing this? (2) [15]
How is the allocation lifetime of this object maintained? (3)

Type relationships (15)
What are the composition, ownership, or usage relationships of this type? (5) [24]
What is this type’s type hierarchy? (4) [24]
What implements this interface? (4) [24]
Where is this method overridden? (2)

Data flow (14)
What is the original source of this data? (2) [15]
What code directly or indirectly uses this data? (5)
Where is the data referenced by this variable modified? (2)
Where can this global variable be changed? (1)
Where is this data structure used (1) for this purpose? (1) [24]
Which parts of this data structure are modified by this code? (1) [24]
What resources is this code using? (1)

Location (13)
Where is this functionality already implemented? (5) [24]
Where is this defined? (3)

Building and branching (11)
Should I branch or code against the main branch? (1)
How can I move this code to this branch? (1)
What do I need to include to build this? (3)
Which includes are unnecessary? (2)
How do I build this without doing a full build? (1)
Why did the build break? (2) [59]
Which preprocessor definitions were active when this was built? (1)

Architecture (11)
How does this code interact with libraries? (4)
What is the architecture of the code base? (3)
How is this functionality organized into layers? (1)
Is our API understandable and flexible? (3)

Concurrency (9)
What threads reach this code (4) or data structure (2)? (6)
Is this class or method thread-safe? (2)
What members of this class does this lock protect? (1)

Dependencies (5)
What depends on this code or design decision? (4) [7]
What does this code depend on? (1)

Method properties (2)
How big is this code? (1)
How overloaded are the parameters to this function? (1)

Many of these already have tools that support them

• Debugging

• Refactoring

• Design Rationale

• So if there's already a tool designed to support this, why is it still so hard??
Supporting information needs

• Debugging is hard.
  • Tool $\text{x}$ claims to make debugging easier!

• Does tool $\text{x}$ help?

• Depends…
  • Does tool $\text{x}$ apply in the situations that make debugging challenging?
  • Do developers have the context they need to invoke tool $\text{x}$
  • Does tool $\text{x}$ reliably produce the information required
  • Are the interactions for using tool $\text{x}$ usable
Debugging (26)

* How did this runtime state occur? (12)
  data, memory corruption, race conditions, hangs, crashes, failed API calls, test failures, null pointers

* Where was this variable last changed? (1)

* Why didn’t this happen? (3)

omniscient debuggers

Record execution history
Provide interactions for browsing or searching

WhyLine [1]
directly supports all 3 questions in some situations


LaToza
CS 695 / SWE 699 Fall 2023
32
Debugging (26)

How do I debug this bug in this environment? (3)

In what circumstances does this bug occur? (3)

statistical debugging [1]

- Sample execution traces on user computers
- Find correlations between crashes and predicates

No need to reproduce environment on developer computer

Examine correlations between crashes and predicates

Debugging (26)

- How is this object different from that object? (1)

- What runtime state changed when this executed? (2)

- Which team’s component caused this bug? (1)
  Which team should I assign this bug to?
Rationale (42)

Why wasn’t it done this other way? (15)

Why was it done this way? (14)

naming, code structure, inheritance relationships, where resources freed, code duplication, algorithm choice, optimization, parameter validation visibility, exception policies

How did this ever work? (4)

Was this intentional, accidental, or a hack? (9)
Refactoring (25)

* Is the existing design a good design? (2)

smell detectors [1], metrics

Look for bad design idioms
Suggests developer refactor

data clumps
feature envy
refused bequest
typecast

instanceof
magic number
long method
large class

* Is there functionality or code that could be refactored? (4)

clone detectors [2]

Detects syntactically similar code
Suggests developer refactor

ComponentUI mui = new MultiButtonUI();
return MultiLookAndFeel.createUIs(mui, (MultiButtonUI) mui);

ComponentUI mui = new MultiColorChooserUI();
return MultiLookAndFeel.createUIs(mui, (MultiColorChooserUI) mui);

obsolete code, duplicated functionality, redundant data between equally accessible data structures


Refactoring (25)

Should I refactor this? (1)

Are the benefits of this refactoring worth the time investment? (3)
Refactoring (25)

* Is it possible to refactor this? (9)

* How can I refactor this (2) without breaking existing users(7)?

IDE refactoring automation
rename
move
pull up
push down
encapsulate field
convert local variable to field
....

changing a method’s scope, moving functionality between layers, changing semantics of config values, making operations more data driven, generalizing code to be more reusable

higher-level refactorings
Barriers in Front-End Web Development

• Where do developers encounter barriers answering questions and get stuck?
  • --> Opportunities to make better tools by reducing barriers

• Let's look at the sorts of questions developers ask on StackOverflow
  • Common challenges that are hard
  • If we find patterns, maybe a question indicates a bigger issue?
**setState doesn't update the state immediately [duplicate]**
I would like to ask why my state is not changing when I do an onClick event. I've search a while ago that I need to bind the onClick function in constructor but still the state is not updating....

Sydney Loteria 9,471 asked Dec 22, 2016 at 8:03

**Why can't I directly modify a component's state, really?**
I understand that React tutorials and documentation warn in no uncertain terms that state should not be directly mutated and that everything should go through setState. I would like to understand...

Marcus Junius Brutus 25k asked Jun 10, 2016 at 19:40

**Programmatically navigate using React router**
With react-router I can use the Link element to create links which are natively handled by react router. I see internally it calls this.context.transitionTo(...). I want to do a navigation. Not ...

George Mauer 113k asked Jun 26, 2015 at 17:38

**How to update nested state properties in React**
I'm trying to organize my state by using nested property like this: this.state = { someProperty: { flag:true } } But updating state like this, this.setState({ someProperty.flag: false })...

Alex Yong 7,025 asked Mar 27, 2017 at 7:51
StackExchange API, posts from 5/2016 - 10/2016 including 24 most frequent front-end web

286,000

Excluding posts unanswered, duplicates, no upvoted questions

50,000

Randomly sampled

1000

Manually excluded posts without upvotes for top answer

666

Excluding posts unrelated to front-end web development

301
I have a constructor function which registers an event handler:

```javascript
function MyConstructor(data, transport) {
  this.data = data;
  transport.on('data', function () {
    alert(this.data);
  });
}

// Mock transport object
var transport = {
  on: function(event, callback) {
    setTimeout(callback, 1000);
  }
};

// called as
var obj = new MyConstructor('foo', transport);
```

However, I'm not able to access the `data` property of the created object inside the callback. It looks like `this` does not refer to the object that was created, but to another one.

I also tried to use an object method instead of an anonymous function:

```javascript
function MyConstructor(data, transport) {
  this.data = data;
  transport.on('data', this.alert);
}

MyConstructor.prototype.alert = function() {
  alert(this.name);
};
```

but it exhibits the same problems.
Programming activities

- implement code from scratch
- comprehend code
- change the behavior of existing code
- resolve a compile time or runtime error
- refactoring
- performance optimization
Evidence referenced in questions and answers

- code
- executable code within a pastebin
- official documentation by the software’s author or standards body
- alternate documentation offered by others including tutorials
- program output
- execution state describing intermediate values computed and observed through debugging aids such as console logging or the debugger
- other
Idioms

• Callback idioms: bind targets, callback contexts, bind configurations

• Graphical idioms: queries, getters, setters

• Object-interaction idioms: valid references, back-end requests, this scope, collections and formats, method chains
Coding Barriers: Example

Q: "...[code snippet] works fine if i remove "400, function()", when i click the menu-trigger, the menu appears. but with it added, the menu appears then disappears too quickly,..."

A: "Remove the display setting which jQuerys slideToggle() sets, that why the menu gets hidden...:"

```javascript
$(this).toggleClass("nav-expanded").css("display",""")
```

GB1 Unidentified Setter
visual property change → code fragment to mutate property
**CALLBACK IDIOMS**

**BIND TARGETS**
Identifying or choosing an event, lifecycle hook, or trigger to register a callback.

- CB1 *Unidentified Target*:
  desired bind target \(\rightarrow\) target name & code fragment

- CB2 *Constrained Target*:
  bind target code fragment \(\rightarrow\) API rules making fragment invalid

- CB3 *Confused Target*:
  current & desired bind targets \(\rightarrow\) API use differences, new target's code fragment

**CALLBACK CONTEXTS**
Identifying when the callback is dispatched, using its arguments, or other related objects.

- CB4 *Improper Scheduling*:
  callback code fragments & desired schedule \(\rightarrow\) correct callback order & code fix

- CB5 *Unidentified State*:
  desired state \(\rightarrow\) API rationale for identifying state & code fragment to obtain it

- CB6 *Missed Callbacks*:
  callback code fragment \(\rightarrow\) API rationale & state required for callback to occur

**BIND CONFIGURATIONS**
Setting options of a callback trigger, or modifying parameters of its bind mechanism.

- CB7 *Incorrect Bind Parameters*:
  callback parameter fragments & desired behavior \(\rightarrow\) correct code fragments

- CB8 *Misconfigured Framework*:
  framework configuration fragments & desired behavior \(\rightarrow\) correct framework code

**OBJECT-INTERACTION IDIOMS**

**VALID REFERENCES**
Determining defined standard, or framework identifiers at compile time or runtime.

- OB1 *Inactionable Reference Error*:
  statement generating error & error message \(\rightarrow\) explanation of error message

- OB2 *Silent Invalid Reference*:
  invalid statement \(\rightarrow\) warning message & statement fixing warning

**BACK-END REQUESTS**
Sending structured data to a server, or handling server responses.

- OB6 *Misconfigured Request*:
  back-end request & desired behavior \(\rightarrow\) modified request matching behavior

- OB7 *Unclear Transmission*:
  back-end request as sent \(\rightarrow\) back-end request as received

- OB8 *Mishandled Response*:
  back-end request \(\rightarrow\) code fragment for response(s) listening and parsing

**SCOPE CONTEXTS**
Identifying the context given to the keyword `this` within a code block, or a variable's visibility.

- OB12 *Unclear Scope*:
  `this` statement \(\rightarrow\) owner scope of `this`

**GRAPHICAL IDIOMS**

**GRAPHICAL SETTERS**
Updating graphical properties of the layout via API (DOM access methods, CSS selectors).

- GB1 *Unidentified Setter*:
  visual property change \(\rightarrow\) code fragment to mutate property

- GB2 *Unobservable Setter*:
  setterA & visual property change \(\rightarrow\) setterB to mutate property

- GB3 *Indirect Setter*:
  setterA \(\rightarrow\) elements which inherit properties from setterA or occlude mutations

- GB4 *Overwritten Setter*:
  setterA \(\rightarrow\) setterB overwriting setterA & code fragments with alternative fixes

**GRAPHICAL QUERIES**
Retrieving graphical elements or similar representations via API (DOM access methods, CSS selectors).

- GB5 *Incomplete Query*:
  queryA and desired elements to be matched \(\rightarrow\) queryB matching those elements

- GB6 *Outdated Query*:
  queryA \(\rightarrow\) changes to query result set over time & code fragment fixing it

- GB7 *Overwritten Query*:
  queryA \(\rightarrow\) queryB intersecting queryA's mutations & code fragment fixing queryA

**GRAPHICAL GETTERS**
Obtaining graphical properties of the layout via API methods.

- GB8 *Unidentified Getter*:
  visual property \(\rightarrow\) getter code fragment to retrieve it

**COLLECTIONS AND FORMATS**
Creating or manipulating a collection, or formatting data for use in a framework or library.

- OB3 *Unidentified Iteration Construct*:
  collection object \(\rightarrow\) code fragment with corresponding iteration construct

- OB4 *Occluded Modification*:
  collection object & loop fragment \(\rightarrow\) modifications of collection per iteration

- OB5 *Confused Formatting*:
  object in format A \(\rightarrow\) code fragment converting object to format B

**METHOD CHAINS**
Determining the effects of a method invocation within a sequence of consecutive calls.

- OB9 *Incomplete Sequence*:
  \(o.m1(\ldots.m2(\ldots.mn(\ldots))\rightarrow o.m1(\ldots.m2(\ldots.mn(\ldots))\)

- OB10 *Incorrect Sequence*:
  \(o.m1(\ldots.m2(\ldots.mn(\ldots))\rightarrow o.mk(\ldots.m1(\ldots.mn(\ldots))\)

- OB11 *Overwritten Effect*:
  \(o.m1(\ldots.m2(\ldots.mn(\ldots))\rightarrow methods mk and mi where both mutate object\)
CALLBACK IDIOMS

BIND TARGETS IDENTIFYING OR CHOOSING AN EVENT, LIFECYCLE HOOK, OR TRIGGER TO REGISTER A CALLBACK

CB1 Unidentified Target:
desired bind target → target name & code fragment

CB2 Constrained Target:
bind target code fragment → API rules making fragment (in)valid

CB3 Confused Target:
current & desired bind targets → API use differences, new target’s code fragment

CALLBACK CONTEXTS IDENTIFYING WHEN THE CALLBACK IS DISPATCHED, USING ITS ARGUMENTS, OR OTHER RELATED OBJECTS

CB4 Improper Scheduling:
callback code fragments & desired schedule → correct callback order & code fix

CB5 Unidentified State:
desired state → API rationale for identifying state & code fragment to obtain it

CB6 Missed Callbacks:
callback code fragment → API rationale & state required for callback to occur

BIND CONFIGURATIONS SETTING OPTIONS OF A CALLBACK TRIGGER, OR MODIFYING PARAMETERS OF ITS BIND MECHANISM

CB7 Incorrect Bind Parameters:
callback parameter fragments & desired behavior → correct code fragments

CB8 Misconfigured Framework:
foundation configuration fragments & desired behavior → correct framework code
Callbacks

Callback context: what args and additional state is available when invoked

bind target: event
bind configuration: parameters
subscribe to controlling behavior

```javascript
x.on("event", ..., function callback(arg){/***/})
```
Common problems with callbacks
Bind Targets: Identifying or choosing an event, life cycle hook, or trigger to register a callback

- **CB1 Unidentified** Target
desired bind target $\rightarrow$ API name & code fragment

- **CB2 Constrained** Target/bind target code fragment $\rightarrow$ API rules making fragment (in)valid

- **CB3 Confused** Target/current & desired bind targets $\rightarrow$ API use differences, new target’s code fragment
OGRAPHICAL IDIOMS

GRAPHICAL SETTERS UPDATING GRAPHICAL PROPERTIES OF THE LAYOUT VIA API (DOM ACCESS METHODS, CSS SELECTORS)

GB1 Unidentified Setter:
visual property change \(\rightarrow\) code fragment to mutate property

GB2 Unobservable Setter:
setterA & visual property change \(\rightarrow\) setterB to mutate property

GB3 Indirect Setter:
setterA \(\rightarrow\) elements which inherit properties from setterA or occlude mutations

GB4 Overwritten Setter:
setterA \(\rightarrow\) setterB overwriting setterA & code fragments with alternative fixes

GRAPHICAL QUERIES RETRIEVING GRAPHICAL ELEMENTS OR SIMILAR REPRESENTATIONS VIA API (DOM ACCESS METHODS, CSS SELECTORS)

GB5 Incomplete Query:
queryA and desired elements to be matched \(\rightarrow\) queryB matching those elements

GB6 Outdated Query:
queryA \(\rightarrow\) changes to query result set over time & code fragment fixing it

GB7 Overwritten Query:
queryA \(\rightarrow\) queryB intersecting queryA’s mutations & code fragment fixing queryA

GRAPHICAL GETTERS OBTAINING GRAPHICAL PROPERTIES OF THE LAYOUT VIA API METHODS

GB8 Unidentified Getter:
visual property \(\rightarrow\) getter code fragment to retrieve it

OBJECT-INTERACTION IDIOMS

COLLECTIONS AND FORMATS CREATING OR MANIPULATING A COLLECTION, OR FORMATTING DATA FOR USE IN A FRAMEWORK OR LIBRARY
Graphical idioms

const [r1, r2] = queryInterface(params);

r1.get("prop") && r2.set({aProp: value, ...})

classical getter    classical setter

graphical query

graphical getter    graphical setter
Common problems with graphical idioms

Graphical Setters: Updating graphical properties of the layout via API (DOM access methods, CSS selectors)

- **GB1 Unidentified** Setter
  visual property change $\rightarrow$ code fragment to mutate property

- **GB2 Unobservable** Setter
  setterA & visual property change $\rightarrow$ setterB to mutate property

- **GB3 Indirect** Setter
  setterA $\rightarrow$ elements which inherit properties from setterA or occlude mutations

- **GB4 Overwritten** Setter
  setterA $\rightarrow$ setterB overwriting setterA & code fragments with alternative fixes
### OBJECT-INTERACTION IDIOMS

#### VALID REFERENCES
- **OB1 Inactionable Reference Error:**
  statement generating error & error message → explanation of error message
- **OB2 Silent Invalid Reference:**
  invalid statement → warning message & statement fixing warning

#### BACK-END REQUESTS
- **OB6 Misconfigured Request:**
  back-end request & desired behavior → modified request matching behavior
- **OB7 Unclear Transmission:**
  back-end request as sent → back-end request as received
- **OB8 Mishandled Response:**
  back-end request → code fragment for response(s) listening and parsing

#### OBJECT-REFERENCE CONTEXTS
- **OB12 Unclear Scope:**
  `this` statement → owner scope of `this`

#### COLLECTIONS AND FORMATS
- **OB3 Unidentified Iteration Construct:**
  collection object → code fragment with corresponding iteration construct
- **OB4 Occluded Modification:**
  collection object & loop fragment → modifications of collection per iteration
- **OB5 Confused Formatting:**
  object in format A → code fragment converting object to format B

#### METHOD CHAINS
- **OB9 Incomplete Sequence:**
  `o.m1(...).m2(...).mn(...)` → `o.m1(...).m2(...).mn(...)
- **OB10 Incorrect Sequence:**
  `o.m1(...).m2(...).mn(...)` → `o.mk(...).m1(...).mn(...)
- **OB11 Overwritten Effect:**
  `o.m1(...).m2(...).mn(...)` → methods `mk` and `ml` where both mutate object
Common problems w/ object-interaction idioms
Valid References: Determining defined standard or framework identifiers at compile time or runtime

- OB1 **Inactionable** Reference Error
  statement generating error & error message → explanation of error message
- OB2 **Silent** Invalid Reference
  invalid statement → warning message & statement fixing warning
Find a new strategy that makes question easier to answer
The way the game is supposed to work is that the snake moves up, down, left, and right (using the keyboard). Every time the snake eats a dot, it grows in length by one. If the snake collides with itself, the game is over.

As you'll see when you play the game, the snake does not move up, down, left, and right. It just seems to move diagonally, and when you press the arrow keys in certain directions, the game ends.
Find an event immediately before the incorrect behavior
Trace control forwards, observing each statement until something incorrect happens
Find the statement that generated the incorrect output. Keep following the data used backwards until you find something that is wrong.
guess and check
backwards search
forwards search
read the docs
check StackOverflow
ask a coworker
draw a whiteboard diagram
Example of a programming strategy

# This Strategy helps you merge 2 branches in github and resolve conflicts

## Required Tools and Environments

**Installing git**

**Github account**

**Ongoing project which is progressing in at least 2 branches**

**Git repository that is not associated with Github**

## Required Knowledge

- Basic git command knowledge
- Knowledge of how to work with terminal and run commands

### STRATEGY GitMerge()

```
# Open the terminal, and use cd(change directory) command to move to the local git project directory
Open the terminal and navigate to your git project directory

IF you are not in the master branch
    # Run the command git checkout master
    checkout to the master branch

IF you are in the master branch
    # To merge the second branch with the master branch run the command "git merge secondBranch", which secondBranch is the name of your git second branch
    Merge the two branches

IF the merge has a conflict
    SET 'conflictedFiles' TO the project files that have a conflict
    FOR EACH 'file' IN 'conflictedFiles'
        DO fixConflict('file')
    # Run GIT STATUS to see the latest changes
    # Run GIT ADD
    # Run GIT COMMIT -m ""
    # Run GIT PUSH
    Commit and push the changes

RETURN nothing
```
Developers often have choices between strategies

**Question** Can I remove this call?

**Strategy:** *Implement & test*  
Remove the call & test behavior change

**Strategy:** *Understand*  
Understand implications before editing by investigating callees
Guess and check debugging

1. Describe in words how the program is failing

2. Brainstorm a list of possible causes of this failure

3. For each possible cause:
   1. Read the potentially defective code.
   2. Gather data about program execution to verify that it is the defect.
   3. If it is the defect, repair it.

4. If you didn’t find the defect, return to 2
1. **STRATEGY** debug()
2. IF the faulty output is logged to a command line
3. SET outputLines TO the line numbers of calls to console logging functions
4. IF the faulty output is graphical output
5. SET outputLines TO the line numbers of function calls that directly render graphics to the screen
6. IF the faulty output is graphical output
7. To find print statements, try searching for keywords related to 'log' or 'print'
8. SET outputLines TO the line numbers of calls to console logging functions
9. To find these lines, try searching for keywords related to graphical output, like 'draw' or 'fill'. Focus on lines that directly render something, not on higher-level functions that indirectly call rendering functions.
10. SET outputLines TO the line numbers of function calls that directly render graphics to the screen
11. Now that you have some lines that could have directly produced the faulty output, you're going to check each line, see if it executed, and then find the cause of it executing. If you're lucky, you only have one output line to check.
12. FOR EACH 'line' IN 'outputLines'
13. Analyze the line to determine its role in the overall behavior of the program
14. Check for errors such as the wrong function being called, the wrong argument being passed to a function, the wrong variable being referenced, or a wrong operator being used.
15. IF any part of 'line' is inconsistent with its purpose
16. You found the bug
17. RETURN 'line'
18. IF the output statement is not wrong, perhaps the line was not supposed to execute at all?
19. IF 'line' was not supposed to execute at all
20. The conditional might be in the same function as the output statement, or it might have been a conditional in a function that called this function. Check the call stack if necessary by setting a breakpoint. Find the conditional that led this line to being executed.
21. Some value in the conditional's boolean expression must have been wrong. Which value was it?
22. SET 'wrongValue' to the value in the conditional's boolean expression that ultimately allowed the faulty output to execute
23. We'll use another strategy to find the source of the incorrect value.
24. RETURN localizeWrongValue('wrongValue')
25. IF the line was supposed to execute, but it executed with an incorrect value, find that value.
26. IF 'line' executed with an incorrect value
27. SET 'wrongValue' TO the incorrect value
28. We'll use another strategy to find the cause of the incorrect value.
29. RETURN localizeWrongValue('wrongValue')
30. # If you made it to this line, then you must have missed something. Is it possible you made a mistake above? If so, go back and verify your work, because something caused the faulty output.
31. RETURN nothing
Many factors influence the effectiveness of a strategy in a situation

<table>
<thead>
<tr>
<th>Influencing factor</th>
<th>Strategy: <em>Implement &amp; test</em> vs.</th>
<th>Strategy: <em>Understand</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Work style [Clarke+04]</td>
<td>Opportunistic</td>
<td>Systematic</td>
</tr>
<tr>
<td>Development process</td>
<td>Test-driven development</td>
<td>Few unit tests</td>
</tr>
<tr>
<td>Cost of bugs</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Time to implement</td>
<td>Easy to implement</td>
<td>Hard to implement</td>
</tr>
<tr>
<td>Difficulty of testing</td>
<td>An easily tested property (e.g., performance)</td>
<td>Non-functional property (e.g., testing usability)</td>
</tr>
<tr>
<td>Test execution time</td>
<td>Short-running test suites</td>
<td>Long-running test suites</td>
</tr>
</tbody>
</table>
Developers often rapidly switch between alternative actions or strategies

Where is method $m$ generating an error?
Rapidly found method $m$ implementing command
Unsure where it generated error

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static call traversal</td>
<td>Statically traversed calls looking for something that would generate error</td>
</tr>
<tr>
<td>debugger</td>
<td>Tried debugger</td>
</tr>
<tr>
<td>grep</td>
<td>Did string search for error, found it, but many callers</td>
</tr>
<tr>
<td>debugger</td>
<td>Stepped in debugger to find something relevant</td>
</tr>
<tr>
<td>static call traversal</td>
<td>Statically traversed calls to explore</td>
</tr>
<tr>
<td>debugger</td>
<td>Went back to stepping debugger to inspect values</td>
</tr>
<tr>
<td></td>
<td>Found the answer</td>
</tr>
</tbody>
</table>

(66 minutes)
Developers often rapidly switch between alternative actions or strategies

Lacks knowledge to determine how these lines influence program behavior

Tries to recover rationale, but no explanation in check-in message

Tests might have identified a bug, but don't prove absence.

Teammates remembered another scenario.

Strategy 1. Guess the answer.
— This was a quick hack, not a reasoned change because otherwise they would have been removed. But what would break if they were here?

Strategy 2. Check code history.
— I commented these out 2 years ago along with many other changes. But why?

— Removed comments, all tests still pass. But did I break anything?

— Sent an email. Teammates replied with a description of a rare input which causes it to break. Success!
Some strategies are more effective than others in a specific situation
Strategies can make a large difference in task performance
programming strategy  a procedure for accomplishing a programming task

Developers work more systematically and efficiently when given effective explicit programming strategies

“Strategies determine success more than does the programmer’s available knowledge”

“Experts seem to acquire a collection of strategies for performing programming tasks.”


better CSS debugging strategy
Q: Was function $F$’s implementation the ideal design, a hack, or accidental?

Strategy for answering:

1. Begin procedure $\text{RetrieveRationaleFromCode}$
   a. Initialize an empty set of rationales $R$
   b. For each comment in the function:
      i. If the comment provides information about the rationale for the implementation
         1. Add the rationale to $R$
   c. If $R$ is non-empty
      i. Synthesize the rationales in $R$ into an answer to the question.
      ii. If you successfully synthesized the rationales
           1. Stop, you have an answer.
      d. This strategy failed. Begin procedure $\text{RetrieveRationaleFromDevelopers}$

2. Begin procedure $\text{RetrieveRationaleFromDevelopers}$
   a. Initialize an empty set of developers $D$
   b. Use version control (e.g., git blame) to identify the developers in the entire history of the function who wrote or modified code, adding each developer to $D$
   c. Use your organization’s default communication channels (e.g., email, IRC, Slack), writing a message to everyone in $D$ asking $Q$
   d. Wait until:
      i. Someone in $D$ responds with the answer, then stop, or
      ii. All in $D$ respond without the answer, or
      iii. You cannot wait any longer.
   e. This strategy failed. Begin procedure $\text{InferRationaleFromCode}$.

3. Begin procedure $\text{InferRationaleFromCode}$
   a. Fully comprehend the behavior of $F$ at the level of computation
   b. Infer the intraprocedural intent of $F$, understanding how $F$ interacts with all of the functions that call it and all of the functions that it calls.
   c. Using the intraprocedural intent of $F$, infer the possible architectural intents of $F$.
   d. Estimate the likelihood of each possible architectural intents of $F$. Which intent is most likely given the intents of the intraprocedural intent of $F$ and the architectural intent of the software?
   e. Select the intent with the highest likelihood, and stop.
   f. If you were unable to infer intents, this strategy failed.
Please select your Strategy

```javascript
const strategyEdit = (response, data) => {
  try {
    localStorage.setItem("strategy", JSON.stringify(data));
    console.log('strategy saved');
  } catch (error) {
    console.log(error);
  }
};

updateKnowledge = (array) => {
  const uniqueItems = new Set();
  array.forEach((item) => {
    uniqueItems.add(item.toLowerCase());
  });

  this.setState({
    requiredKnowledge: Array.from(uniqueItems)
  })
  localStorage.setItem("new_requiredKnowledge", JSON.stringify(Array.from(uniqueItems)));}

updateTools = async (array) => { let uniqueItems = new Set(); let formattedAllTools = new Map();
  Object.values(this.state.allTools).forEach((item) => {
    formattedAllTools.set(item.toLowerCase(), item);
  });

  for (let i = 0; i < array.length; i++) {
    let item = array[i];
    let key = item.toLowerCase();
    if (formattedAllTools.has(key)) {
      uniqueItems.add(formattedAllTools.get(key));
    } else {
      let success = true;
      try {
        await axios.post("/dataManagement/technologies", {name: item});
      } catch (error) {
        console.log(error);
      }
    }
  }
```
STRATEGY :: strategy IDENTIFIER (IDENTIFIER+) STATEMENTS

STATEMENTS :: STATEMENT+

STATEMENT :: * (ACTION | CALL | CONDITIONAL | FOREACH | ASSIGNMENT | RETURN )+

ACTION :: (word I IDENTIFIER)+ .

CALL :: do identifier ( IDENTIFIER* )

CONDITIONAL :: if QUERY STATEMENTS

FOREACH :: for each IDENTIFIER in identifier STATEMENTS

UNTIL :: until QUERY STATEMENTS

ASSIGNMENT :: set IDENTIFIER to QUERY

RETURN :: return QUERY

QUERY :: (word I IDENTIFIER I CALL)+

IDENTIFIER :: ' identifier '
ASSIGNMENT :: set IDENTIFIER to QUERY

SET 'conflictedFiles' TO the project files that have a conflict

FOREACH :: for each IDENTIFIER in identifier STATEMENTS

FOR EACH 'file' IN 'conflictedFiles'
# If you've spent a lot of time debugging unfamiliar code, the way that you probably debug is
# to first look at the failure, then look at the code to understand how it's architected, and
# then look for possible reasons for why the program failed. Once you have a guess, you
# probably then check it with things like breakpoints and logging. This strategy often works
# if you can have a lot of prior experience with debugging and inspecting program state. But
# if you don't have that experience, or you happen to guess wrong, this approach can lead to
# a lot of dead ends.
#
# The strategy you're about to use is different. Instead of guessing and checking, this
# strategy involves systematically working backwards from the code that directly caused the
# failed output to all of the code that caused that failed output to occur. As you work
# backwards, you'll check each statement for defects. If you work backwards like this,
# following the chain of causality from failure to cause, you will almost certainly find the
# bug.

**STRATEGY** debug()

# This first step will give you enough familiarity to find lines in the program that create
# the program's output. Read the names of all of the functions and variables in the program
# Some programs produce command line output with print statements.
# Is the faulty output you're investigating printed to a command line?

**IF** the faulty output is logged to a command line

# To find print statements, try searching for keywords related to 'log' or 'print'
**SET** outputLines TO the line numbers of calls to console logging functions
# Graphical output includes things like colored lines and rectangles

**IF** the faulty output is graphical output

# To find these lines, try searching for keywords related to graphical output, like '
# draw' or 'fill'. Focus on lines that directly render something, not on higher-level
# functions that indirectly call rendering functions.
**SET** outputLines TO the line numbers of function calls that directly render graphics to
# the screen

# Now that you have some lines that could have directly produced the faulty output, you're
# going to check each line, see if it executed, and then find the cause of it executing. If
# you're lucky, you only have one output line to check.

**FOR EACH** 'line' IN 'outputLines'

# First, let's make sure the line executed. You want to be sure that this is actually the
# source of the wrong output. You can check this by inserting a logging statement, or
# setting a breakpoint on the line.
**IF** the program executed 'line'

# Analyze the line to determine its role in the overall behavior of the program
# Check for errors such as the wrong function being called, the wrong argument being
# passed to a function, the wrong variable being referenced, or a wrong operator being
**Strategy: Design task**

<table>
<thead>
<tr>
<th>Template</th>
<th>Self-guided</th>
<th>Guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Found and used example code as a template for implementation.</td>
<td>4/14 (29%)</td>
<td>0/14 (0%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decompose</th>
<th>Self-guided</th>
<th>Guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzed functional requirements for sub-problems, implementing each independently</td>
<td>9/14 (64%)</td>
<td>0/14 (0%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TDD</th>
<th>Self-guided</th>
<th>Guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translated functional requirements into test cases, identifying sub-problems from test case requirements.</td>
<td>2/14 (14%)</td>
<td>11/14 (79%)</td>
</tr>
</tbody>
</table>

**Strategy: Debugging task**

<table>
<thead>
<tr>
<th>Guess &amp; check</th>
<th>Self-guided</th>
<th>Guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants found suspicious lines of code, modifying them and checking the effects of their modification.</td>
<td>4/14 (29%)</td>
<td>0/14 (0%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Forward search</th>
<th>Self-guided</th>
<th>Guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants identified where the program began processing input, following its execution forward</td>
<td>9/14 (64%)</td>
<td>0/14 (0%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Backward search</th>
<th>Self-guided</th>
<th>Guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants identified faulty output and worked backwards through control and data flow dependencies</td>
<td>2/14 (14%)</td>
<td>11/14 (79%)</td>
</tr>
</tbody>
</table>

Design task

**1.30** times more likely to make more progress

\[ p < 0.023^* \]

Debugging task

**1.96** times more likely to make more progress

\[ p < 0.004^* \]

are programming strategies tacit?
STRATEGY FixCss(buggedElement)
# You can use filter input to search for it
# Or you can scroll through the styles manually
Search through the stylings to find where it gets its undesired value

SET 'undesiredStyling' TO the line number and css file found in the search

IF 'undesiredStyling' is not found
# You will find all stylings applied to the element here
# Once you found the stylings you were looking for
# You can click small arrow to jump to the place it gets its value

Click on Computed tab and use filter to search

SET 'undesiredStyling' TO line number found here

SET 'perfectStyleList' TO an empty list of css properties
UNTIL buggedElement has desired styling
# you can add or change different css styles to the element
# it then applies instantly to element stylings

Use element.Style to apply css to buggedElement
add the style proptery to 'perfectStyleList'

DO ApplyCssToElement(buggedElement, 'perfectStyleList')

STRATEGY ApplyCssToElement(element, style)
# Css rules are cascading. The one with most priority applies
# This is how priority gets evaluated
# !important | style="" | id selector | class attribute, psudo class selector | type selector and psudo element
# For easy explanation: use this url: http://qinimate.com/dive-into-css-specificity/
# Also if there are two css files having the same selector, the file placed last in order is evaluated

IF style has to be applied to only this element
# e.g. choose last css file in order, use id selector and so on
Use strongest selector, apply style to element
RETURN nothing

IF style has to be applied on many elements
use class selector, apply style to element
RETURN nothing

• Strategy-related
  • Generality
  • Ambiguity
  • Imprecise steps
  • Required tool use

“I used chrome but still I was not able to find the NET section to find the CSS component. It took me a long time to find the component.”

• Mismatch between the level of knowledge assumed by the strategy and possessed by the user

**code interacting with framework**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Likelihood (odds ratio)</th>
</tr>
</thead>
<tbody>
<tr>
<td>search online forum</td>
<td>3.84</td>
</tr>
<tr>
<td></td>
<td>(3.84x more likely)</td>
</tr>
<tr>
<td>create diagrams</td>
<td>0.51</td>
</tr>
<tr>
<td></td>
<td>(0.51x less likely)</td>
</tr>
</tbody>
</table>

feeling stressed / nervous (LVHA)

- add print statements: 2.42 (2.42x more likely)
- read surrounding code: 0.17 (0.17x less likely)

feeling sad / depressed (LVLA)

experiment with edits

likelihood (odds ratio)

0.09
(0.09x less likely)

feeling excited / enthusiastic (HVHA)

ask for help from a colleague

likelihood (odds ratio) 2.13 (2.13x more likely)

Ways to work more effectively
be more effective with

metacognition   be aware of your problem solving process
be more effective with

**self-regulation** monitor progress and use of time

(Robillard et al. 2004; Falkner et al. 2014)
be more effective with better strategies
be more effective with sharing strategies
be aware of impact of how you feel

- feeling stressed / nervous (LVHA)
- feeling sad / depressed (LVLA)
- feeling excited / enthusiastic (HVHA)
participate in a programming strategies mentoring session

email marab@gmu.edu
What's next: Supporting information needs

- **Editing** Code
  - Structured editors: writing code, without the syntax errors
  - Program transformation: editing code with GUI commands
  - GUI builders & No Code: generating code with GUI commands
  - Program synthesis: transforming text into code

- **Understanding** Code
  - Live Programming: working with immediate, real time feedback
  - Computational Notebooks
    - Reusing code - external APIs
    - Navigating code - getting around and reading internal code
    - Software visualization - using diagrams and pictures to make sense of code.

- **Fixing** Code
  - Detecting defects
  - Debugging