

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 hours in the life of a professional software developer

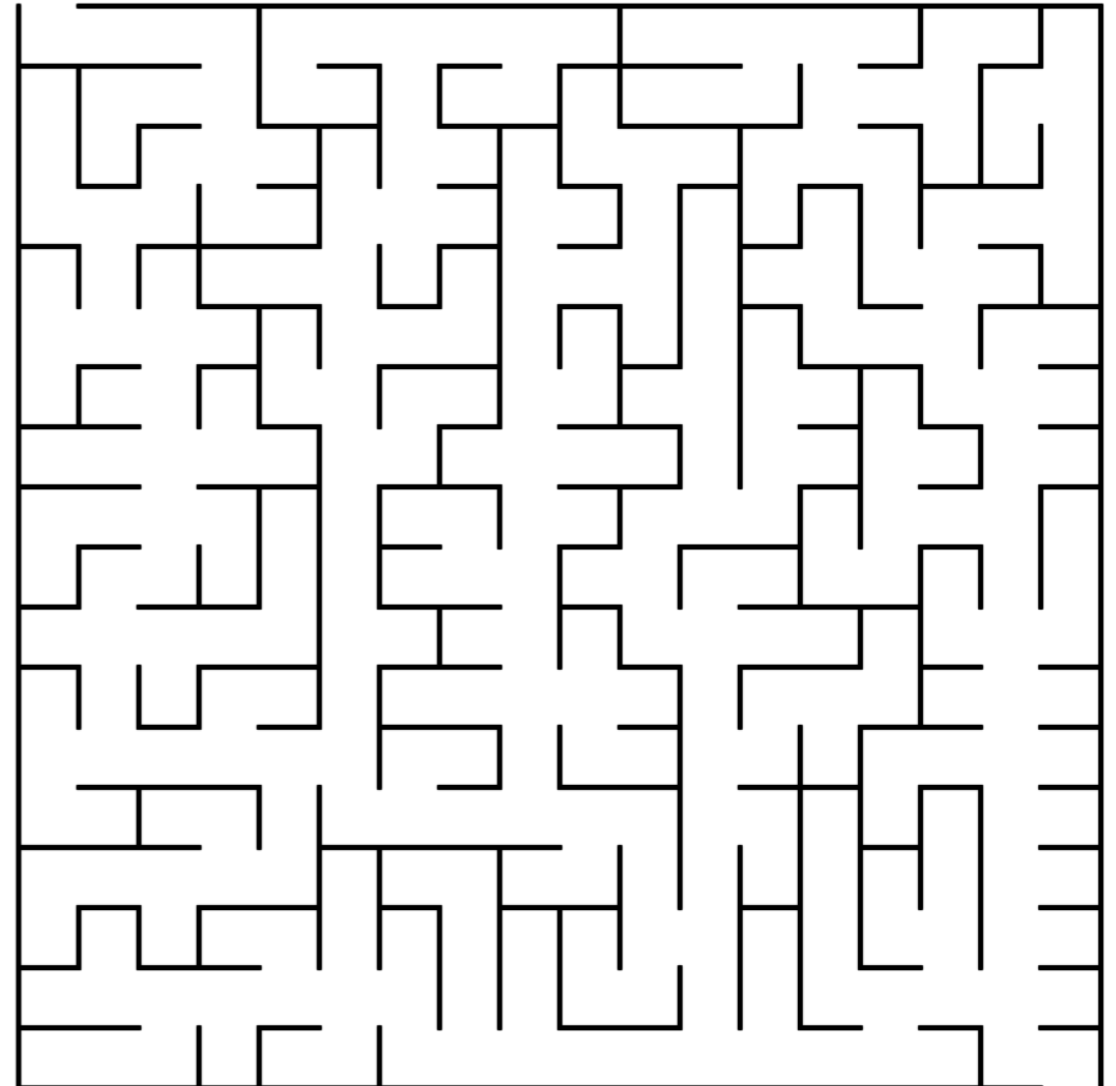
collaboration	Developer assigned bug by team
programming	Reproduces error Browser hits error message (500 internal error) Attaches debugger Browse to page again, hit null reference exception Hypothesize from call stack which function might be responsible Browse through code Uses debugger to change values & experiment Make change, recompile, check, doesn't work Navigates slice, wrong values came from objects In complicated code doesn't understand
collaboration	Walks to B's office and asks where data comes from B working on high profile feature in area
programming	Tries to make change, still doesn't work
collaboration	Walks back to B, realize related to C's feature, C at lunch After lunch, A and B walk to C's office,
design	A, B, C change design to work with new feature
collaboration	Bug passed from A to C to change feature

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

task Investigate and fix a design problem

question Why is an event being issued by forcing a cache update?

How is BufferHandler using its buffer field? Are there any other mutations on it?

Read methods of BufferHandler

Why is there a buffer member variable that is never used?

Investigate references to BufferHandler.buffer

Why is doDelayedUpdate() a member of BufferHandler?

Reads methods along path, concludes that BufferHandler tracks update delays

Why wouldn't isFoldStart() call getFoldLevel()

Reads isFoldStart(), getFoldAtLine()
Concludes isFoldStart() doesn't call because of short circuit evaluation

Implement fix

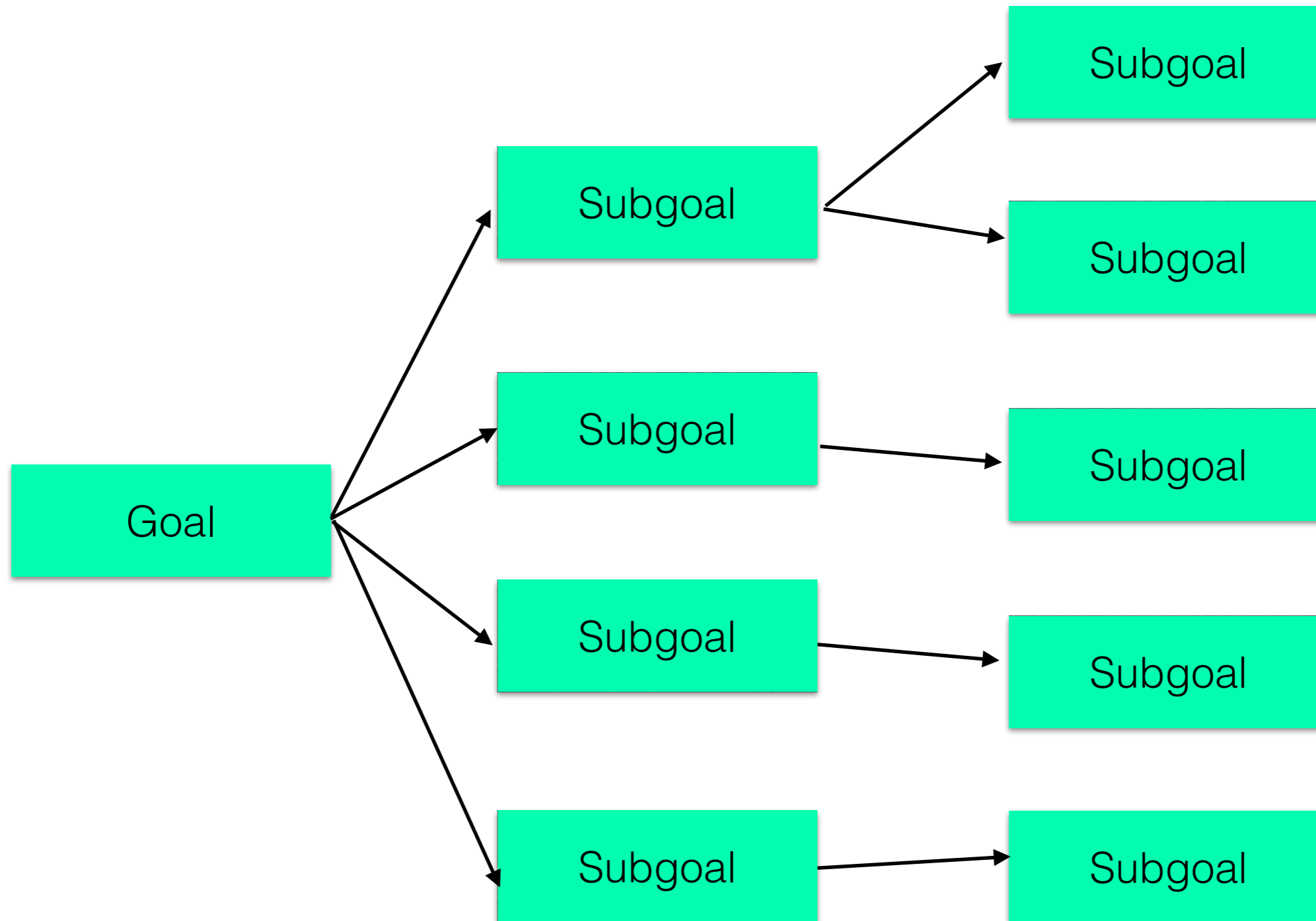
Assure correctness

Set conditional break point
Check that jEdit still appears to work correctly
Repro original bug by reinserting

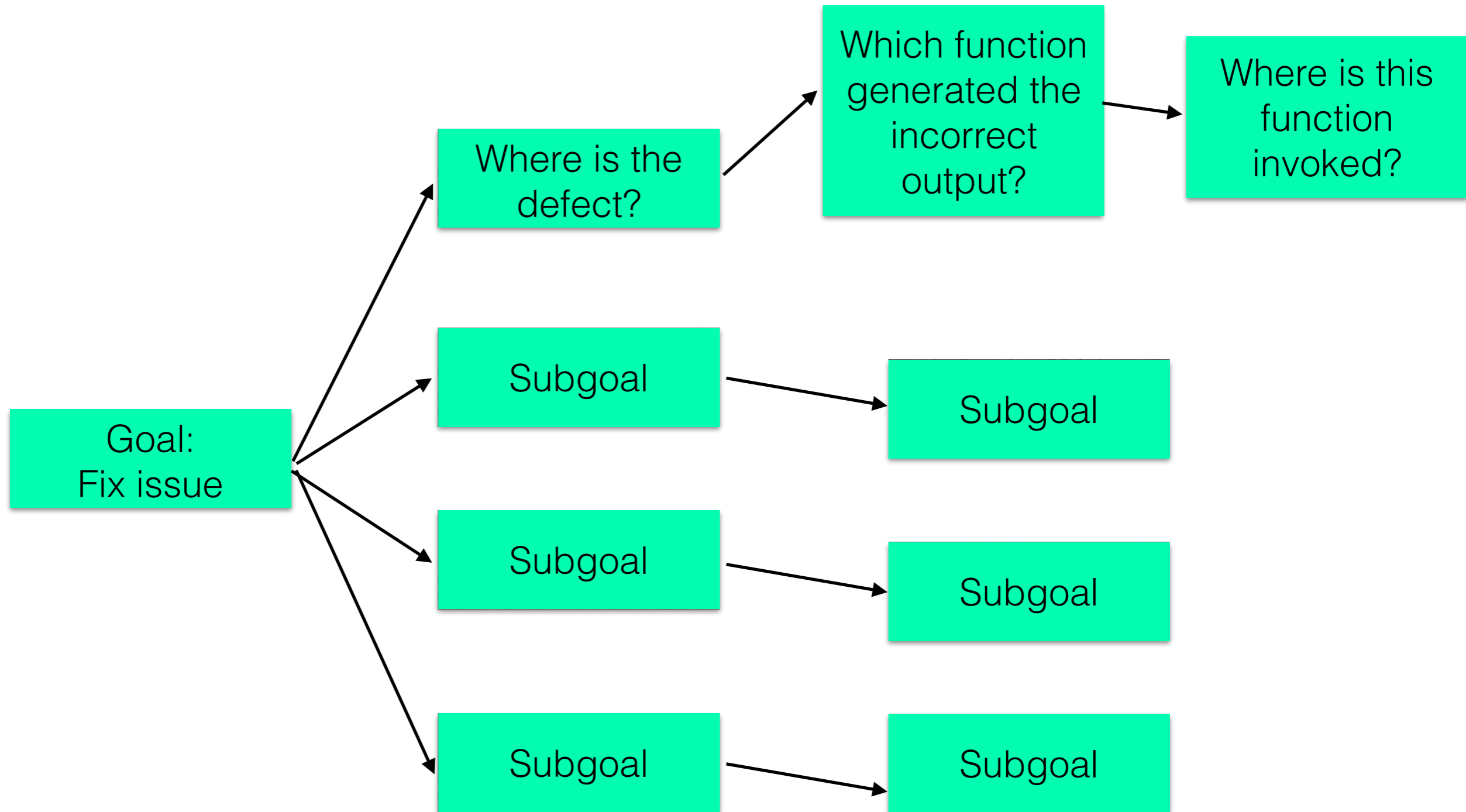


IDE

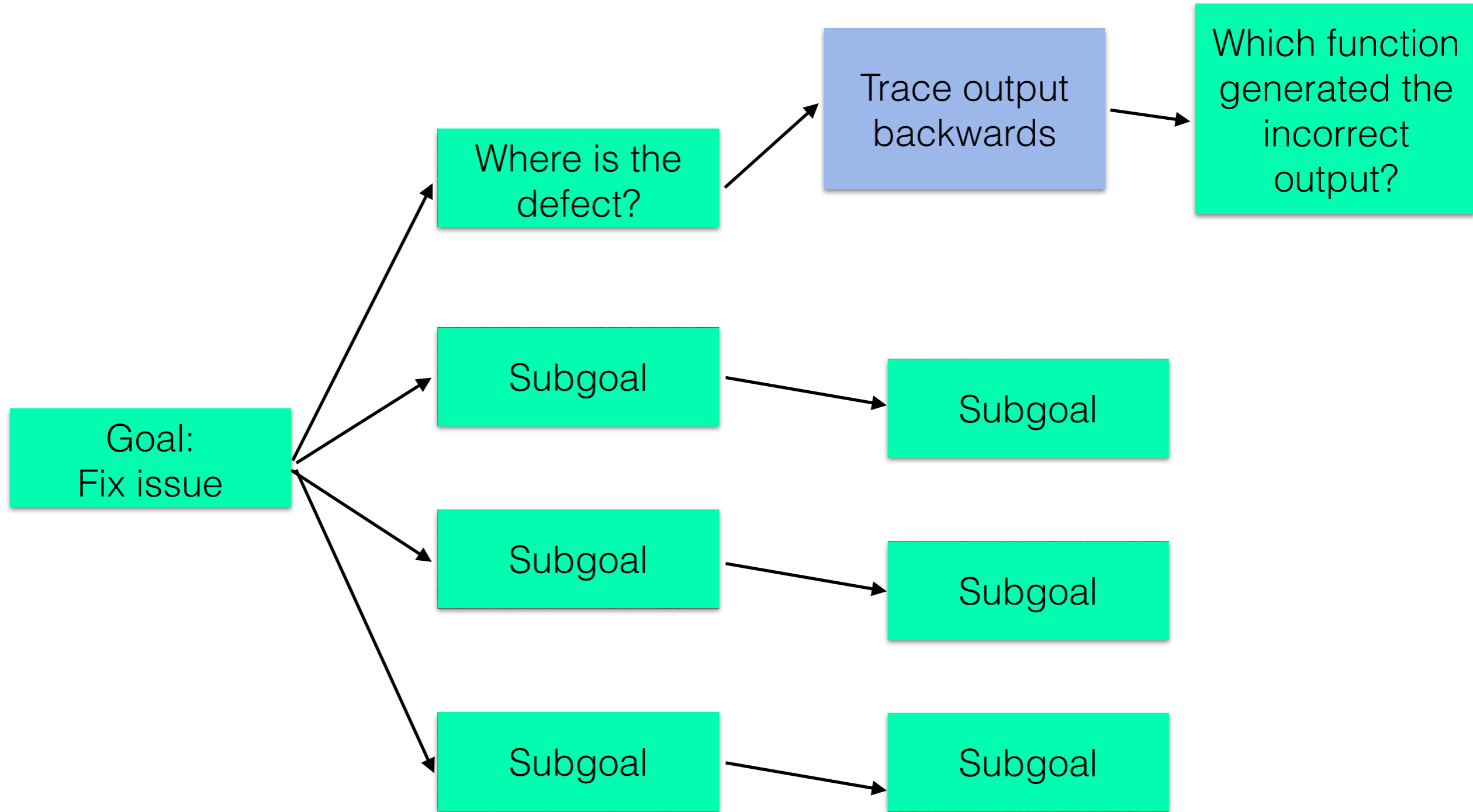
Problem solving is recursive



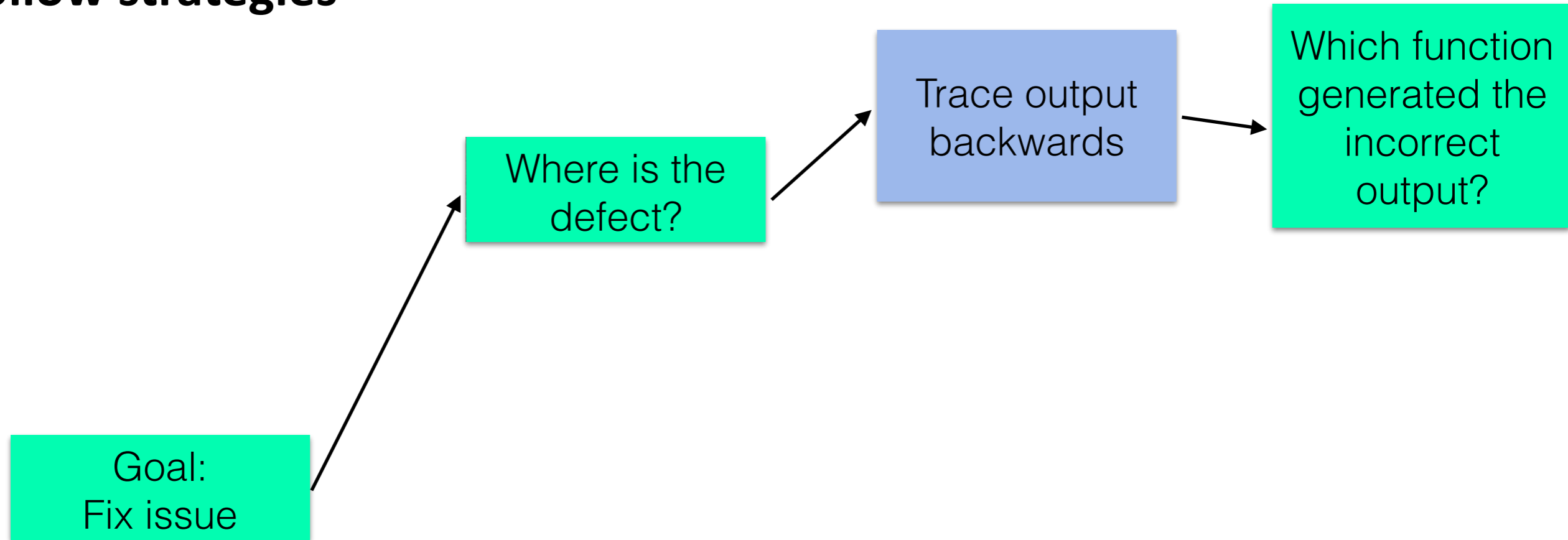
Problem solving involves answering questions



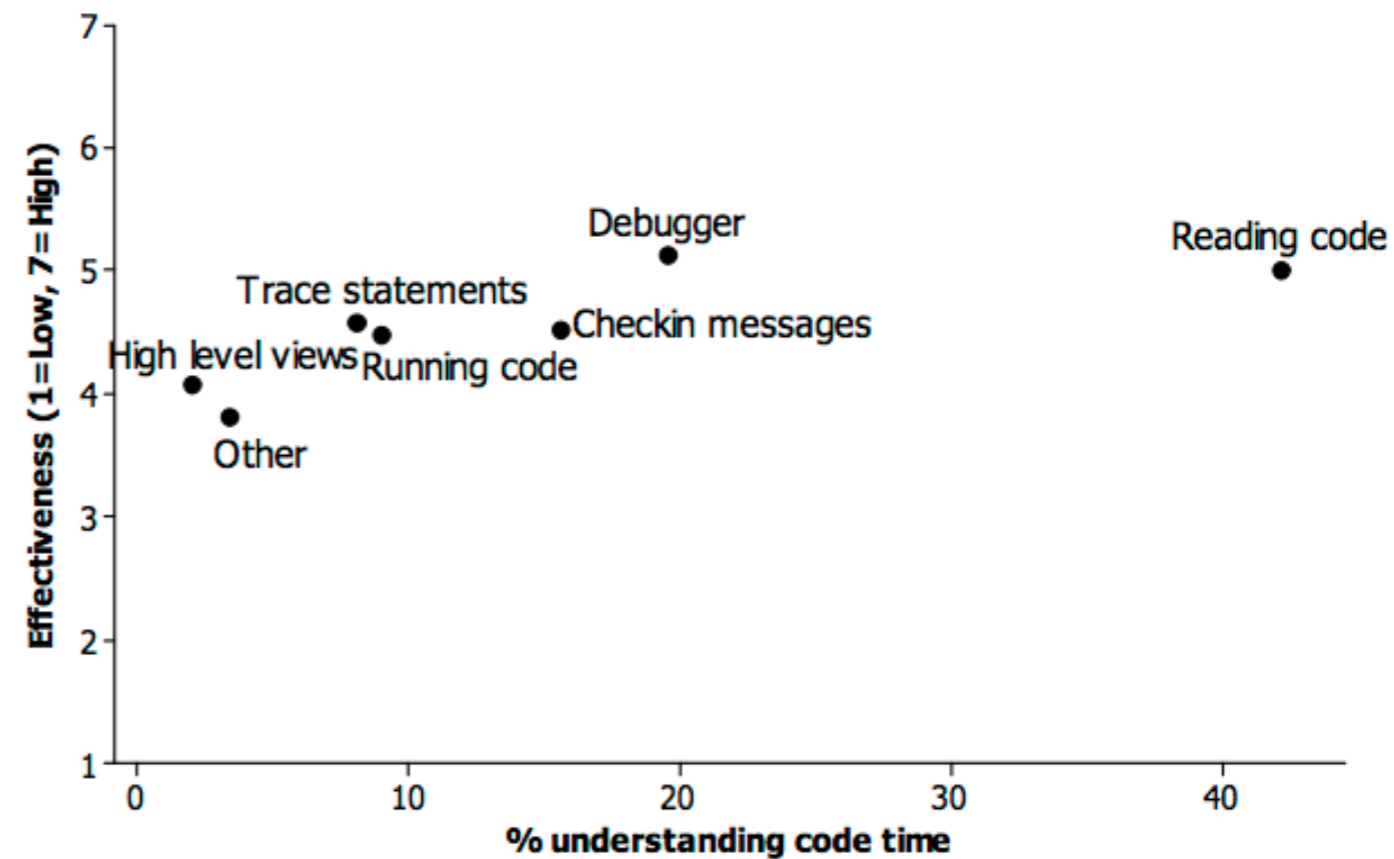
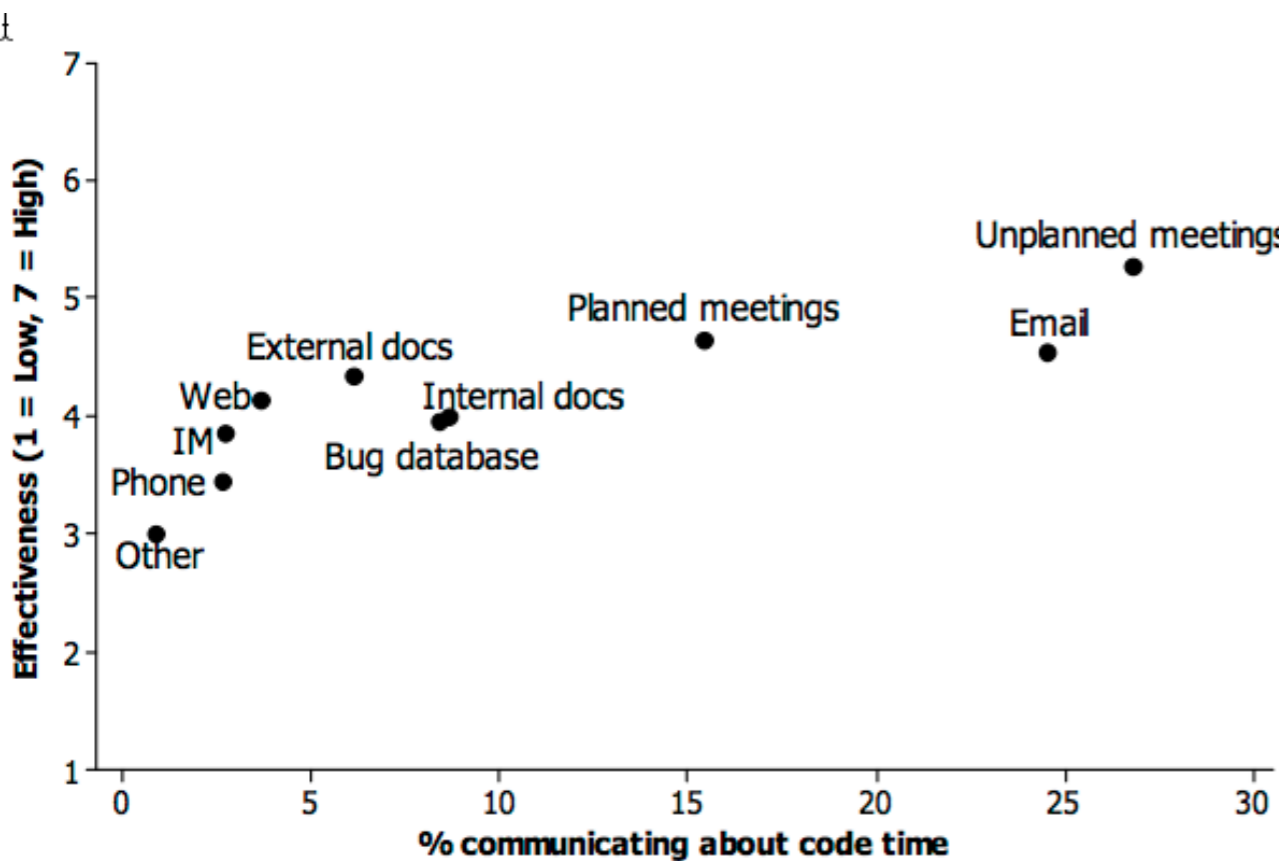
Problem solving involves strategies



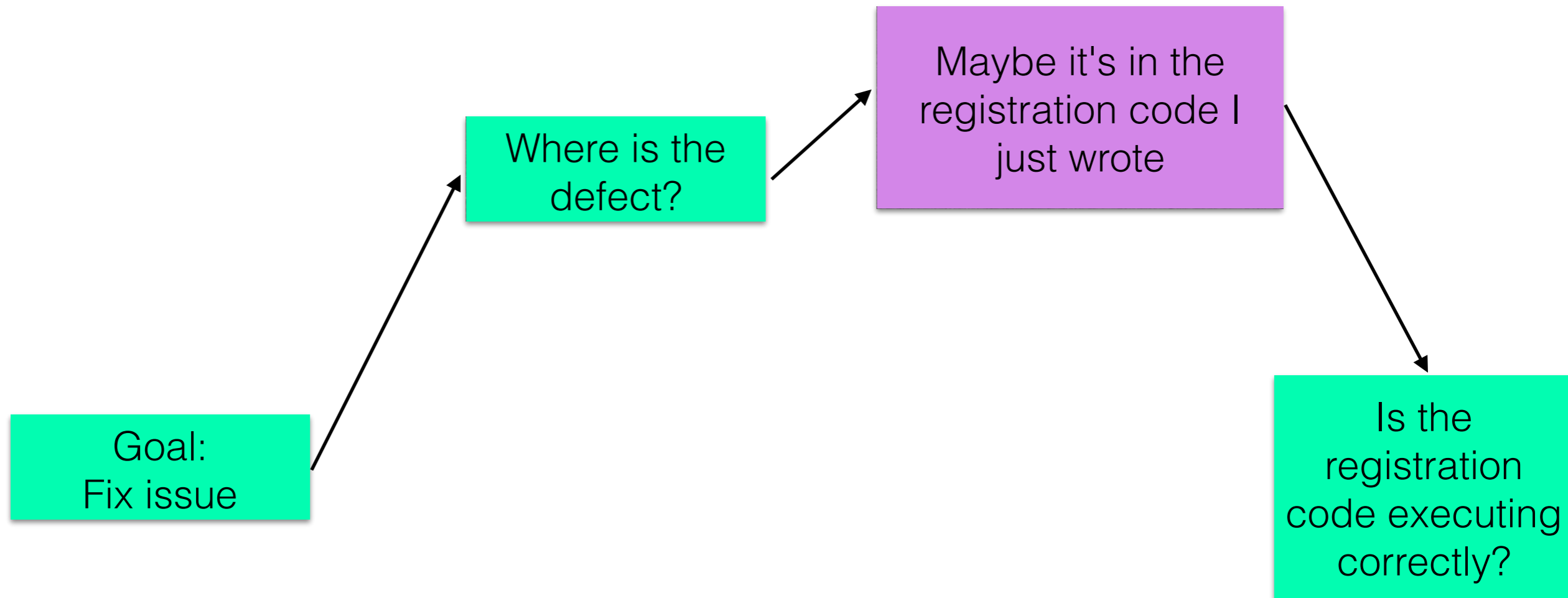
Problem solving involves taking actions to answer questions and follow strategies



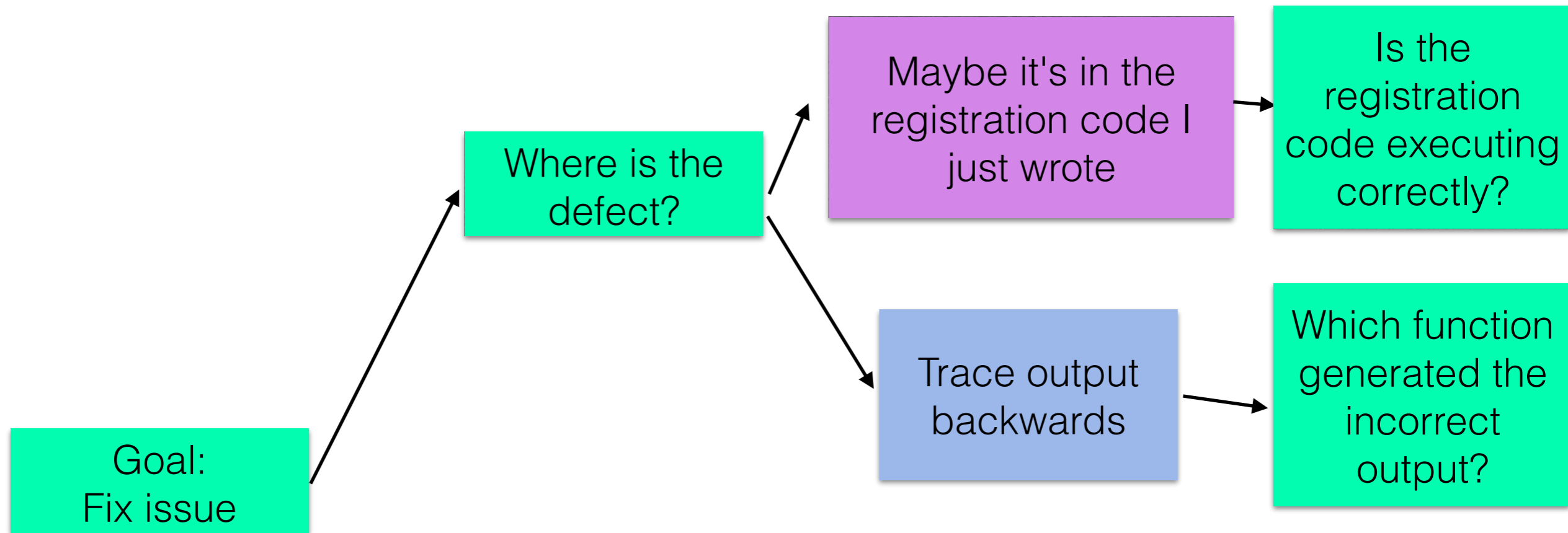
Developers use a variety of techniques for obtaining information and answering questions



Problem solving involves formulating hypotheses



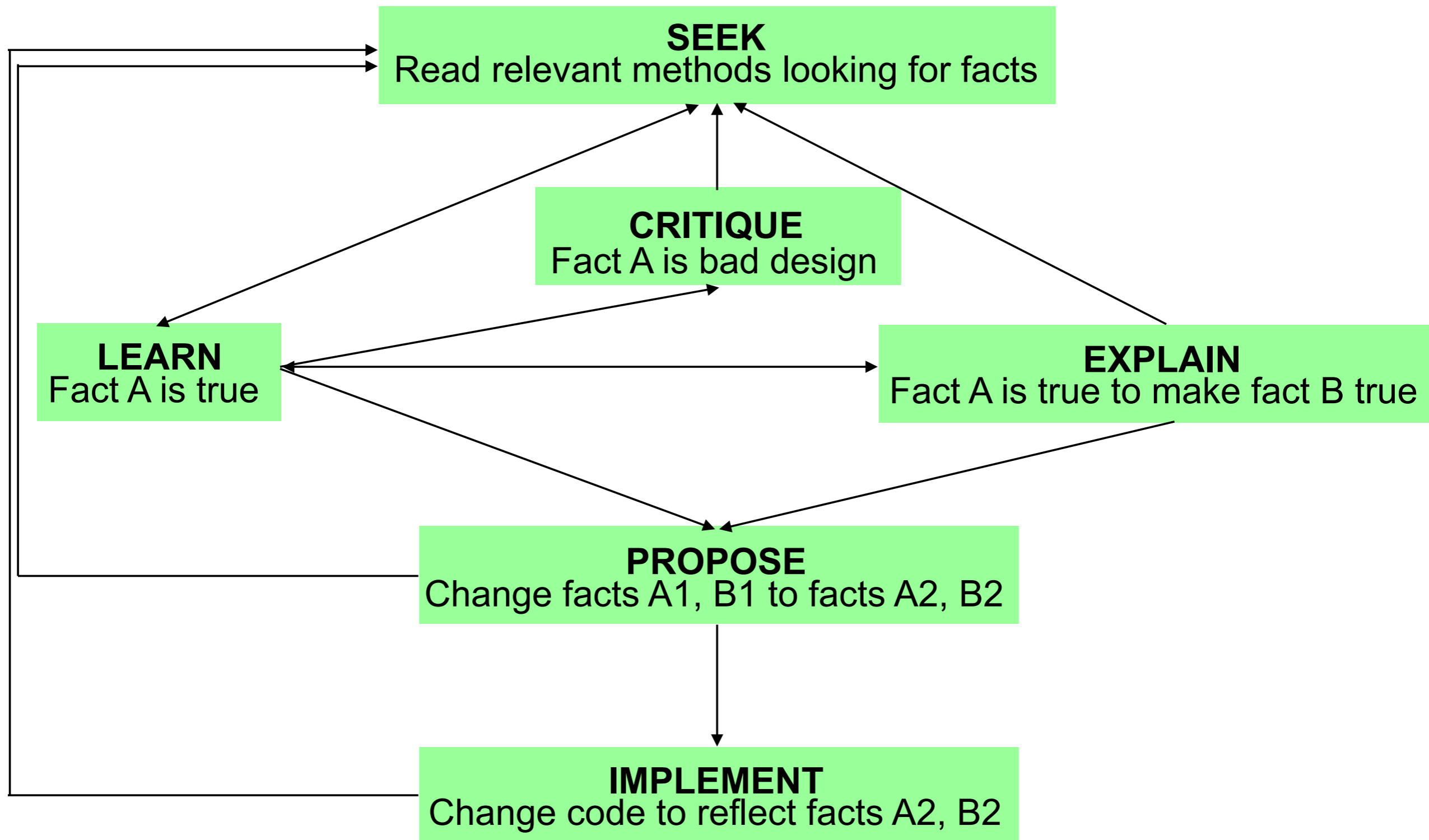
Problem solving involves choices between strategies



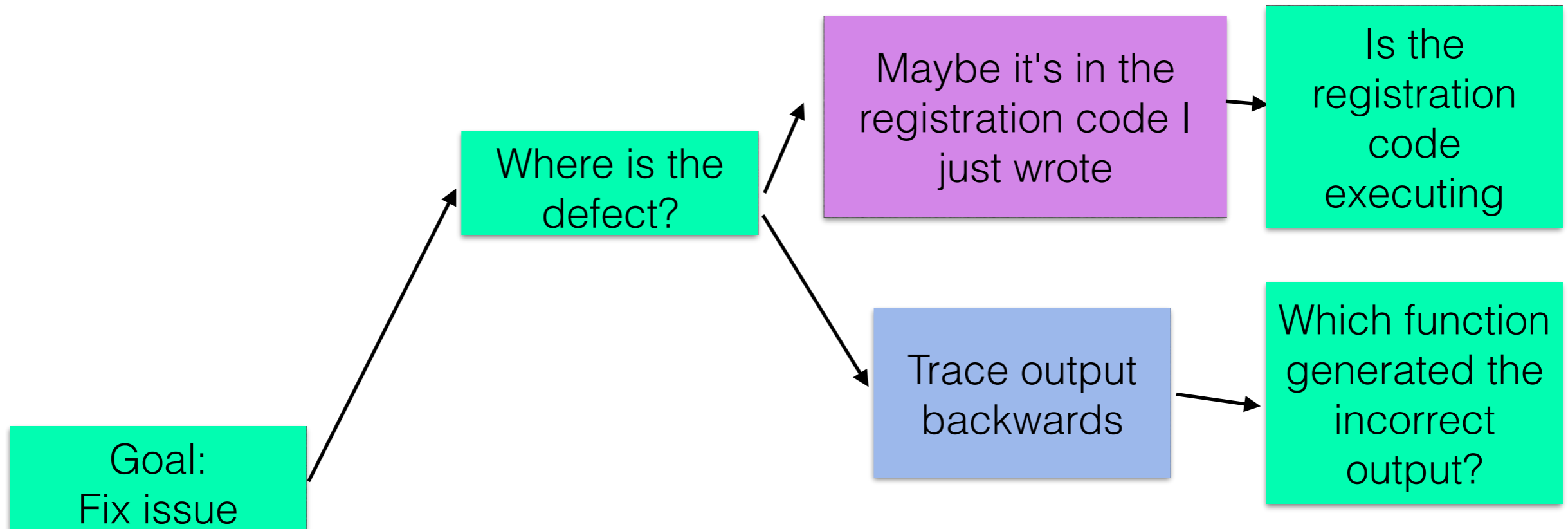
Problem solving in programming

- 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

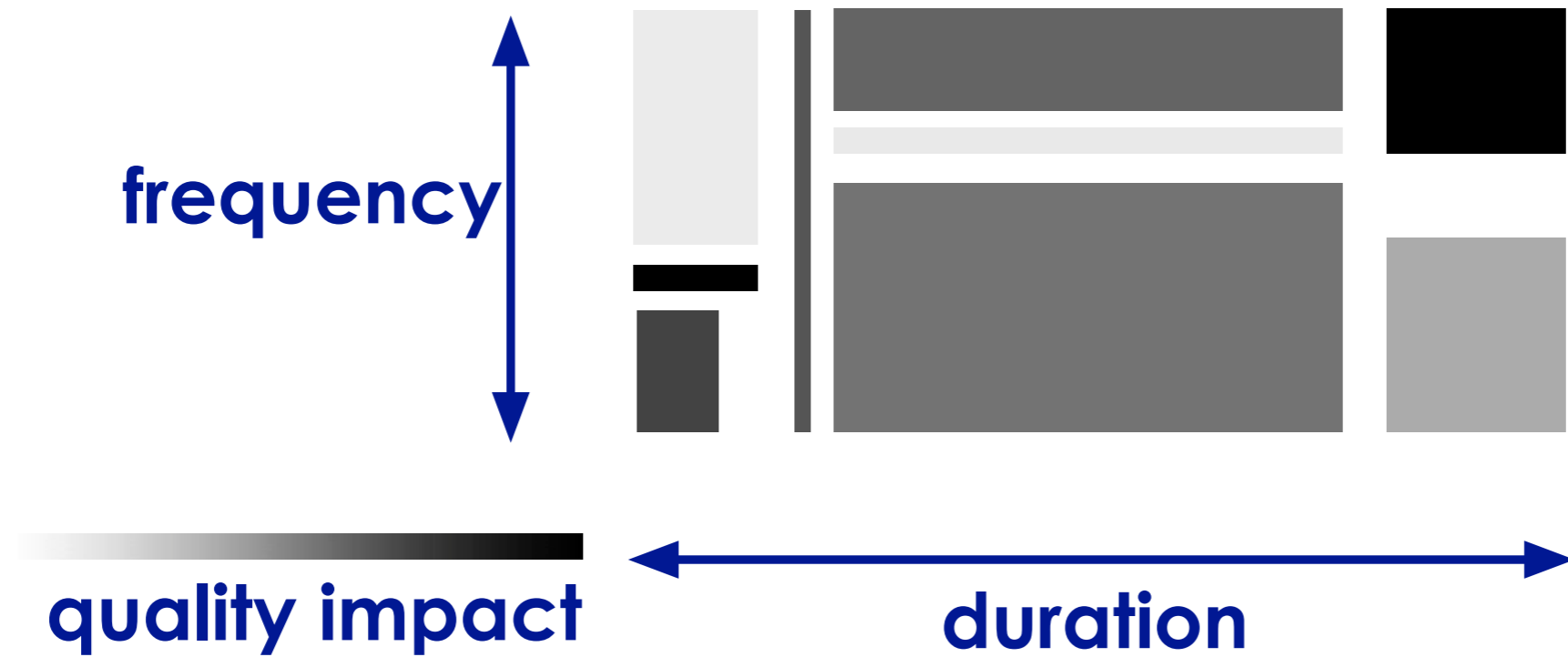


Supporting programming activities

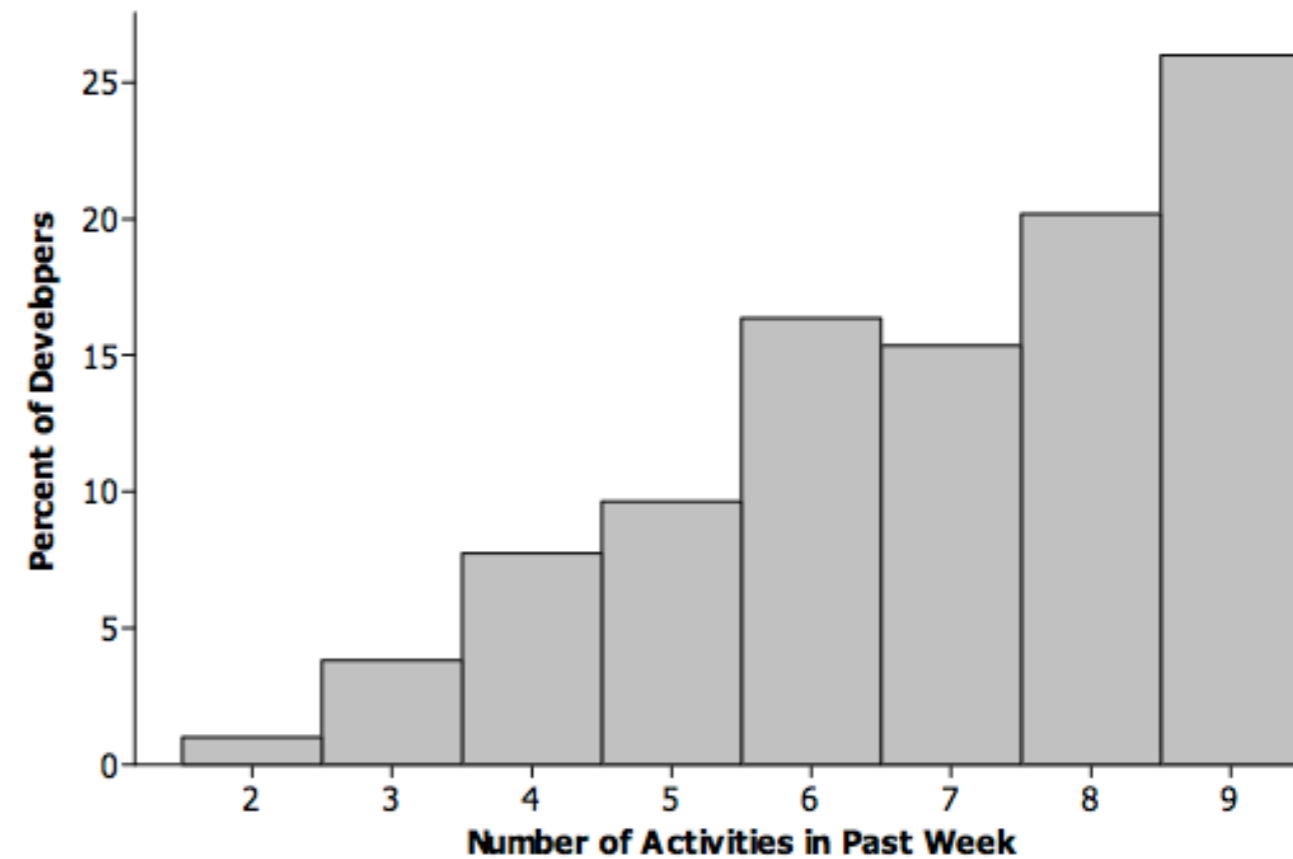
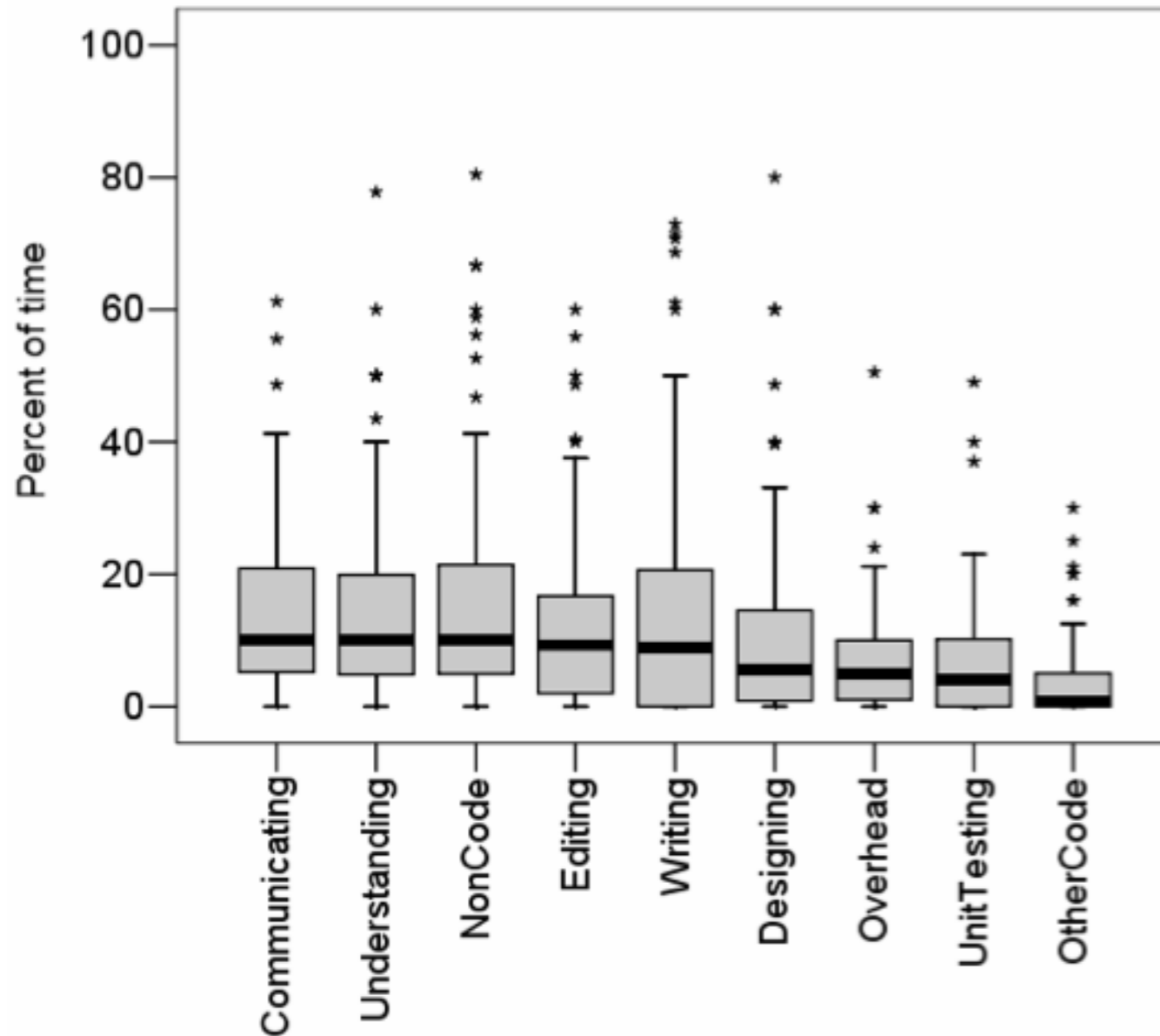


- Many potential points of intervention, supporting subgoals / strategies / question answering / testing hypotheses

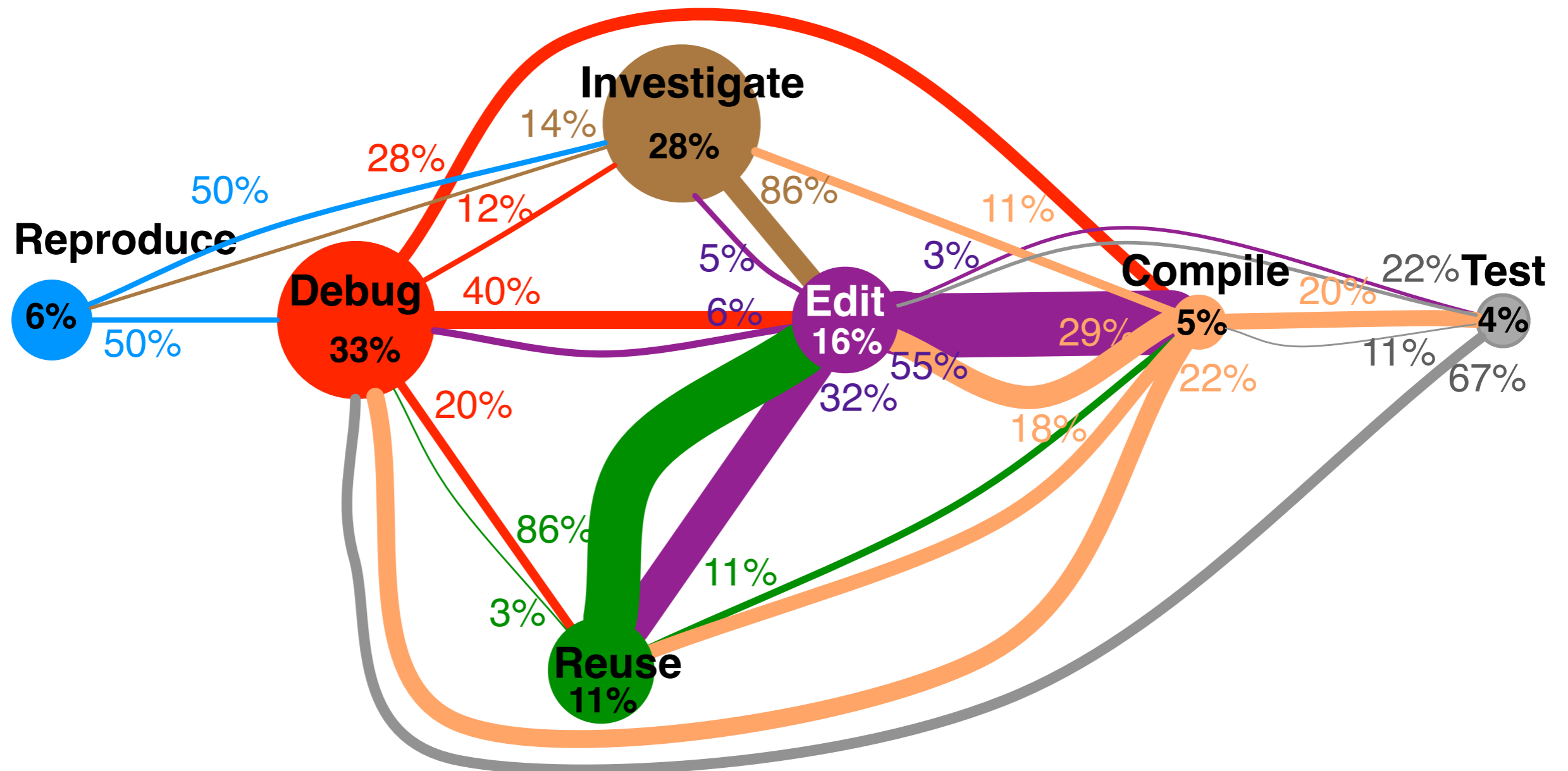
Useful interventions solve important problems



What percentage of the last week have you spent...



Example: Activities in fixing a defect



Circle size: % of time

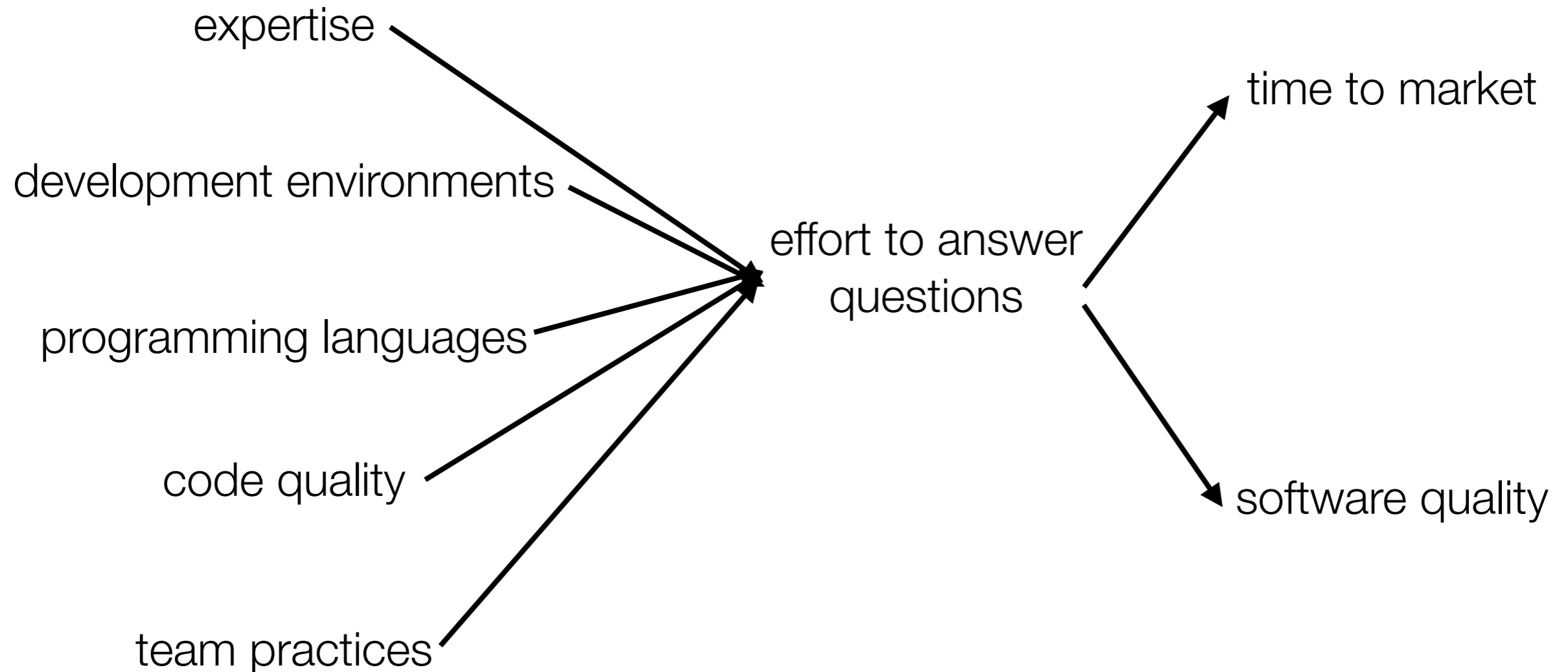
Edge thickness: % of transitions observed

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

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



- 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

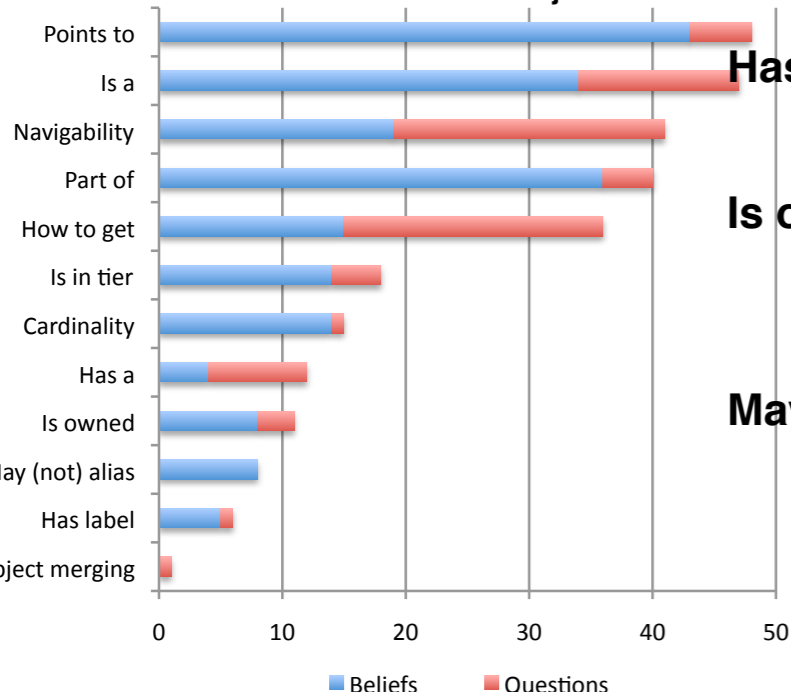
- Tools help developers be more productive by reducing the time to answer questions, increasing likelihood of success
- This requires
 - understanding **precisely** the information required and context available to developers
 - insight into a **mechanism** to make a question easier to answer

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?

Has a	Maybe I would start with the Drawing object and that should have a list of listeners
Is owned	[...] 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 alias	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...

Questions and beliefs about object structure



Marwan Abi-Antoun, Nariman Ammar, and Thomas LaToza. 2010. Questions about object structure during coding activities. Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '10). ACM, New York, NY, USA, 64-71. DOI=<http://dx.doi.org/10.1145/1833310.1833321>

Example: Programming questions

information type	search times			% agreed info is...			frequency and outcome of searches				frequency of sources
	min	mid	max	import.	unavail.	inacc.	acquired	deferred	gave up	beyond obs.	br = bug report, dbug = debugger
s1 Did I make any mistakes in my new code?	0	1	6	59	7	12				dbug 10 compile 26 intuition 6 unit test 4
a2 What have my coworkers been doing?	0	1	11	17	10	10				coworker 20 email 13 tool 4 bug alert 4 im 2
u3 What code caused this program state?	0	2	21	90	49	32				dbug 16 br 3 intuition 3 log 3 tools 3 code 2 coworker 1
r2 In what situations does this failure occur?	0	2	49	80	32	20				br 8 coworker 8 inference 5 tools 3 dbug 2 comment 1
d2 What is the program supposed to do?	0	1	21	93	29	29				spec 13 coworker 9 docs 5 email 1
a1 How have resources I depend on changed?	0	1	9	41	15	15				tools 12 coworker 6 email 4 br 2 code 1
u1 What code could have caused this behavior?	0	2	17	73	20	22				coworker 5 intuition 4 log 4 br 4 dbug 2 im 1 code 1 spec 1
c2 How do I use this data structure or function?	0	1	14	71	20	29				docs 11 code 5 coworker 4 spec 1
d3 Why was this code implemented this way?	0	2	21	61	37	39				code 4 intuition 4 history 3 coworker 2 dbug 2 tools 2 comment 1 br 1
b3 Is this problem worth fixing?	0	2	6	44	10	20				coworker 12 email 2 br 1 intuition 1
d4 What are the implications of this change?	0	2	9	85	44	49				coworker 13 log 1
d1 What is the purpose of this code?	1	1	5	56	24	29				intuition 5 code 2 dbug 2 tools 2 spec 1 docs 1
u2 What's statically related to this code?	0	1	7	66	27	27				tools 8 intuition 2 email 1
b1 Is this a legitimate problem?	0	1	2	49	17	34				br 5 coworker 1 log 1
s2 Did I follow my team's conventions?	0	7	25	41	10	15				docs 2 tools 2 memory 1
r1 What does the failure look like?	0	0	2	88	24	23				br 3 screenshot 2
s3 Which changes are part of this submission?	0	2	3	61	7	5				tools 2 memory 2
c3 How I can coordinate this with this other code?	1	1	4	75	28	30				docs 2 code 1 coworker 1
b2 How difficult will this problem be to fix?	2	2	4	41	15	32				code 1 coworker 1 screenshot 1
c1 What can be used to implement this behavior?	2	2	2	61	27	22				memory 1 docs 1
a3 What information was relevant to my task?	1	1	1	59	15	13				memory 2

This is a serious problem for me	% agree
Code Understanding	
Understanding the rationale behind a piece of code	66%
Understanding code that someone else wrote	56%
Understanding the history of a piece of code	51%
Understanding code that I wrote a while ago	17%
Task Switching	
Having to switch tasks often because of requests from my teammates or manager	62%
Having to switch tasks because my current task gets blocked	50%
Modularity	
Being aware of changes to code elsewhere that impact my code	61%
Understanding the impact of changes I make on code elsewhere	55%
Links between Artifacts	
Finding all the places code has been duplicated	59%
Understanding who “owns” a piece of code	50%

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(7)? (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 pre(3)/post(2)conditions can be made?
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)? (2) [24]
What 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 implemented? (5) [24]
Is this functionality already implemented? (5) [15]
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)
What 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 **x** claims to make debugging easier!
- Does tool **x** help?
- Depends...
 - Does tool **x** apply in the situations that make debugging challenging?
 - Do developers have the context they need to invoke tool **x**
 - Does tool **x** reliably produce the information required
 - Are the interactions for using tool **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)

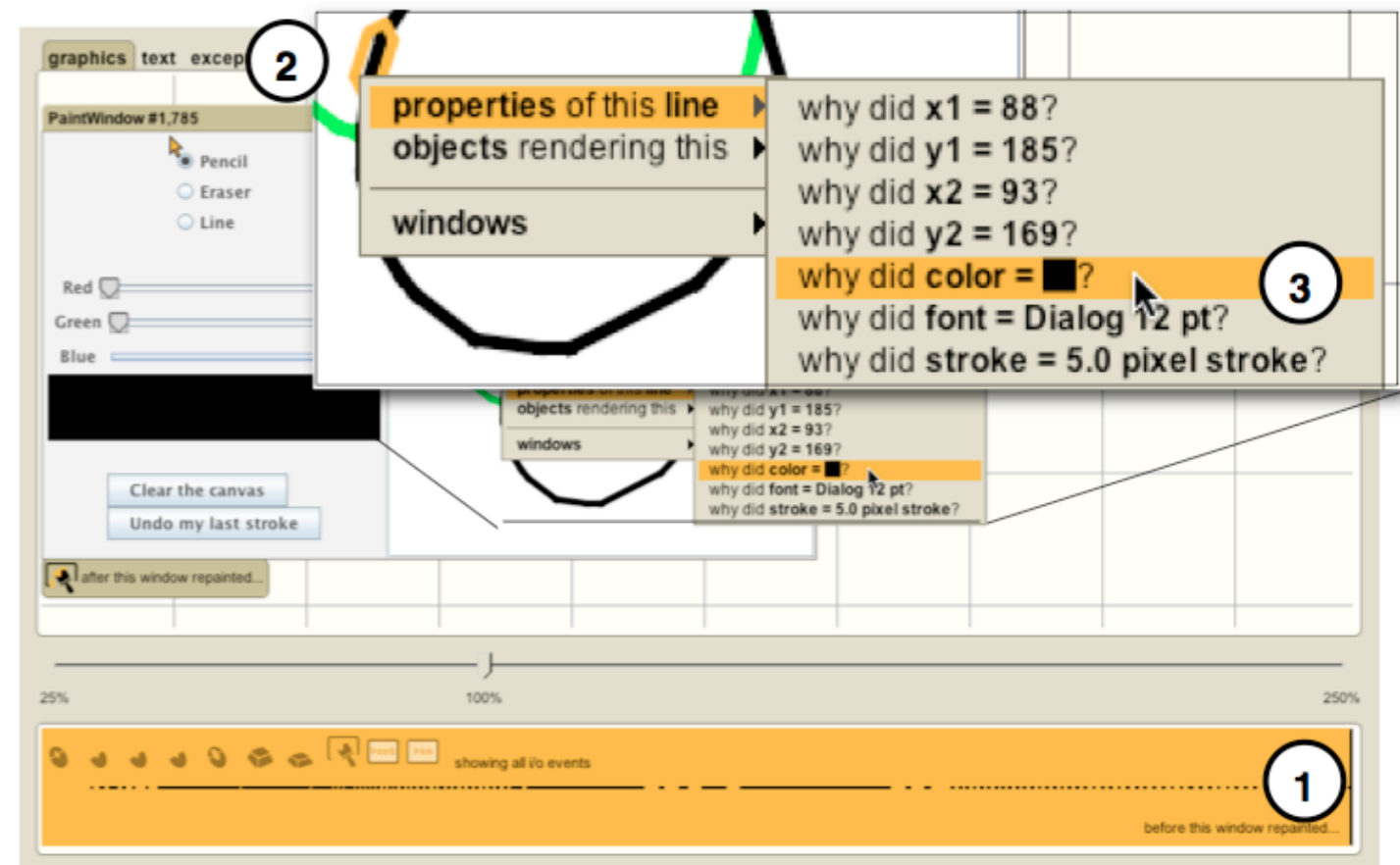
omniscient debuggers

Record execution history

Provide interactions for browsing or searching

WhyLine [1]

directly supports all 3 questions in some situations



[1] Ko, A.J., and Myers, B.A. (2008). Debugging reinvented: asking and answering why and why not questions about program behavior. In *Proc. of the Int'l Conf. on Soft. Eng. (ICSE)*.

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

[1] Liblit, B., Aiken, A., Zheng, A. X., and Jordan, M. I. 2003. Bug isolation via remote program sampling. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*.

Debugging (26)

✘ *How is this object **different** from that object? (1)*

✘ *Which **team's** component caused this bug? (1)*

Which team should I assign this bug to?

✘ *What runtime state **changed** when this executed? (2)*

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

clone

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

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

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

[1] Murphy-Hill, E. and Black, A. P. (2008). Seven habits of a highly effective smell detector. In *Proc of Recommendation Systems for Software Engineering at FSE*.

[2] Kamiya, T., Kusumoto, S., and Inoue, K. (2002). CCFinder: a multi-linguistic token-based code clone detection system for large scale source code. In *TSE*, 28(7).

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?

159 votes

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....

✓ 15 answers

160k views

javascript reactjs ecmascript-6

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

125 votes

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 understa...

✓ 7 answers

58k views

javascript reactjs

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

2096 votes

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 ...

✓ 38 answers

1.3m views

reactjs react-router

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

586 votes

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 })...

✓ 34 answers

428k views

javascript reactjs ecmascript-6 setstate

 Alex Yong **7,025** asked Mar 27, 2017 at 7:51

Related Tags

javascript × 9735

react-native × 1909

redux × 1434

react-router × 1353

react-hooks × 1335

typescript × 1177

node.js × 1169

material-ui × 1073

react-redux × 897

webpack × 848

[more related tags](#)

Hot Network Questions

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

How to access the correct `this` inside a callback

Asked 8 years, 9 months ago Modified 10 months ago Viewed 573k times

I have a constructor function which registers an event handler:

```
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);
```

Run code snippet

Expand snippet

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:

```
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.

13 Answers

Sorted by: Highest score (default) 
Trending sort available 

What you should know about `this`

2196

`this` (aka "the context") is a special keyword inside each function and its value only depends on *how* the function was called, not *how/when/where* it was defined. It is not affected by lexical scopes like other variables (except for arrow functions, see below). Here are some examples:

```
function foo() {
  console.log(this);
}

// normal function call
foo(); // `this` will refer to `window`

// as object method
var obj = {bar: foo};
obj.bar(); // `this` will refer to `obj`

// as constructor function
new foo(); // `this` will refer to an object that inherits from `foo.prototype`
```

To learn more about `this`, have a look at the [MDN documentation](#).

How to refer to the correct `this`

Use [arrow functions](#)

ECMAScript 6 introduced *arrow functions*, which can be thought of as lambda functions. They don't have their own `this` binding. Instead, `this` is looked up in scope just like a normal variable. That means you don't have to call `.bind`. That's not the only special behavior they have, please refer to the MDN documentation for more information.

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

Don't use `this`

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....:"

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

GB1 Unidentified Setter
visual property change → code fragment to mutate property

51% CALLBACK IDIOMS

29% 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

25% 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

23% 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:**
framework configuration fragments & desired behavior → correct framework code

42% GRAPHICAL IDIOMS

37% GRAPHICAL SETTERS UPDATING GRAPHICAL PROPERTIES OF THE LAYOUT VIA API (DOM ACCESS METHODS, CSS SELECTORS)

- GB1 Unidentified Setter:**
visual property change → code fragment to mutate property
- GB2 Unobservable Setter:**
setterA & visual property change → setterB to mutate property
- GB3 Indirect Setter:**
setterA → elements which inherit properties from setterA or occlude mutations
- GB4 Overwritten Setter:**
setterA → setterB overwriting setterA & code fragments with alternative fixes

21% 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 → queryB matching those elements
- GB6 Outdated Query:**
queryA → changes to query result set over time & code fragment fixing it
- GB7 Overwritten Query:**
queryA → queryB intersecting queryA's mutations & code fragment fixing queryA

8% GRAPHICAL GETTERS OBTAINING GRAPHICAL PROPERTIES OF THE LAYOUT VIA API METHODS

- GB8 Unidentified Getter:**
visual property → getter code fragment to retrieve it

40% OBJECT-INTERACTION IDIOMS

21% 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

16% BACK-END REQUESTS SENDING STRUCTURED DATA TO A SERVER, OR HANDLING SERVER RESPONSES

- 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

8% SCOPE CONTEXTS IDENTIFYING THE CONTEXT GIVEN TO THE KEYWORD **this** WITHIN A CODE BLOCK, OR A VARIABLE'S VISIBILITY

- OB12 Unclear Scope:** **this** statement → owner scope of **this**

20% COLLECTIONS AND FORMATS CREATING OR MANIPULATING A COLLECTION, OR FORMATTING DATA FOR USE IN A FRAMEWORK OR LIBRARY

- 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

8% METHOD CHAINS DETERMINING THE EFFECTS OF A METHOD INVOCATION WITHIN A SEQUENCE OF CONSECUTIVE CALLS

- OB9 Incomplete Sequence:**
`o.m1(...).m2(...)...mn(...)` → `o.m1(...).m2(...)...mk(...)...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

51% CALLBACK IDIOMS

29% **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

25% **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

23% **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:

framework configuration fragments & desired behavior → correct framework code

40% OBJECT-INTERACTION IDIOMS

21% **VALID REFERENCES** DETERMINING DEFINED STANDARD, OR FRAMEWORK IDENTIFIERS AT COMPILE TIME OR RUNTIME

20% **COLLECTIONS AND**
OR FORMATTING DA

Callbacks

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

```
x.on("event", ..., function callback(arg) {/**/})
```

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

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 → API 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

42% GRAPHICAL IDIOMS

37% GRAPHICAL SETTERS UPDATING GRAPHICAL PROPERTIES OF THE LAYOUT VIA API (DOM ACCESS METHODS, CSS SELECTORS)

GB1 Unidentified Setter:

visual property change → code fragment to mutate property

GB2 Unobservable Setter:

setterA & visual property change → setterB to mutate property

GB3 Indirect Setter:

setterA → elements which inherit properties from setterA or occlude mutations

GB4 Overwritten Setter:

setterA → setterB overwriting setterA & code fragments with alternative fixes

21% 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 → queryB matching those elements

GB6 Outdated Query:

queryA → changes to query result set over time & code fragment fixing it

GB7 Overwritten Query:

queryA → queryB intersecting queryA's mutations & code fragment fixing queryA

8% GRAPHICAL GETTERS OBTAINING GRAPHICAL PROPERTIES OF THE LAYOUT VIA API METHODS

GB8 Unidentified Getter:

visual property → getter code fragment to retrieve it

40% OBJECT-INTERACTION IDIOMS

, OR FRAMEWORK

20% COLLECTIONS AND FORMATS CREATING OR MANIPULATING A COLLECTION, OR FORMATTING DATA FOR USE IN A FRAMEWORK OR LIBRARY

Graphical idioms

graphical query

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

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

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 → code fragment to mutate property
- **GB2 Unobservable Setter**
setterA & visual property change → setterB to mutate property
- **GB3 Indirect Setter**
setterA → elements which inherit properties from setterA or occlude mutations
- **GB4 Overwritten Setter**
setterA → setterB overwriting setterA & code fragments with alternative fixes

CB6 Missed Callbacks:

callback code fragment → API rationale & state required for callback to occur

23%

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

CB7 Incorrect Bind Parameters:

GB6 Outdated Query:

queryA → changes to query result set over time & code fragment fixing it

GB7 Overwritten Query:

queryA → queryB intersecting queryA's mutations & code fragment fixing queryA

9%

GRAPHICAL GETTERS OBTAINING GRAPHICAL PROPERTIES OF THE LAYOUT

40%

OBJECT-INTERACTION IDIOMS

21%

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

16%

BACK-END REQUESTS SENDING STRUCTURED DATA TO A SERVER, OR HANDLING SERVER RESPONSES

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

8%

SCOPE CONTEXTS IDENTIFYING THE CONTEXT GIVEN TO THE KEYWORD `this` WITHIN A CODE BLOCK, OR A VARIABLE'S VISIBILITY

OB12 Unclear Scope: `this` statement → owner scope of `this`

20%

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

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

8%

METHOD CHAINS DETERMINING THE EFFECTS OF A METHOD INVOCATION WITHIN A SEQUENCE OF CONSECUTIVE CALLS

OB9 Incomplete Sequence:

`o.m1(...).m2(...)...mn(...)` → `o.m1(...).m2(...)...mk(...)...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 t
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

vs.

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

```


Many factors influence the effectiveness of a strategy in a situation

Influencing factor	Strategy: <i>Implement & test</i> vs.	Strategy: <i>Understand</i>
Work style [Clarke+04]	Opportunistic	Systematic
Development process	Test-driven development	Few unit tests
Cost of bugs	Low	High
Time to implement	Easy to implement	Hard to implement
Difficulty of testing	An easily tested property (e.g., performance)	Non-functional property (e.g., testing usability)
Test execution time	Short-running test suites	Long-running test suites

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

static call traversal

Statically traversed calls looking for something that would generate error

debugger

Tried debugger

grep

Did string **search** for error, found it, but many callers

debugger

Stepped in debugger to find something relevant

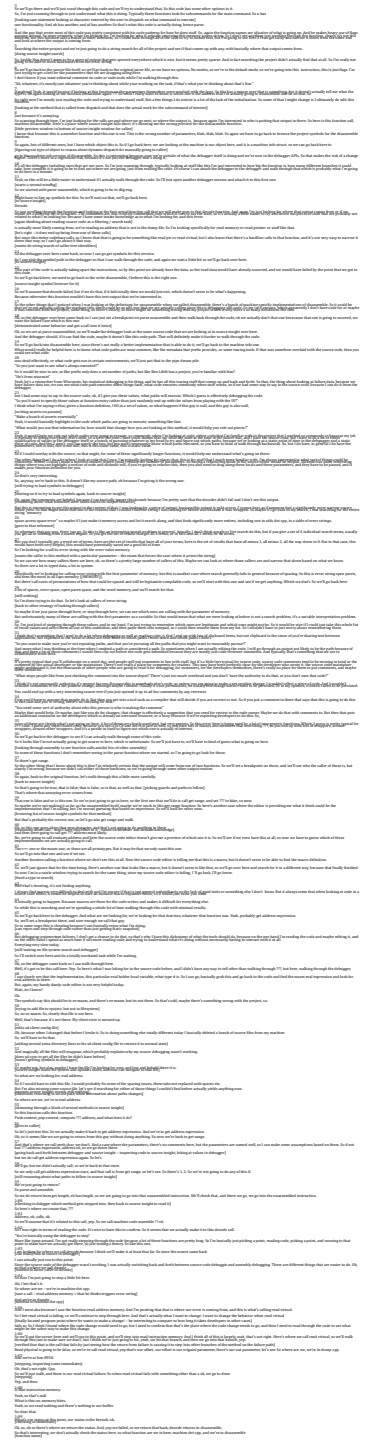
static call traversal

Statically **traversed** calls to explore

debugger

Went back to **stepping** debugger to inspect values
Found the answer

(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?*

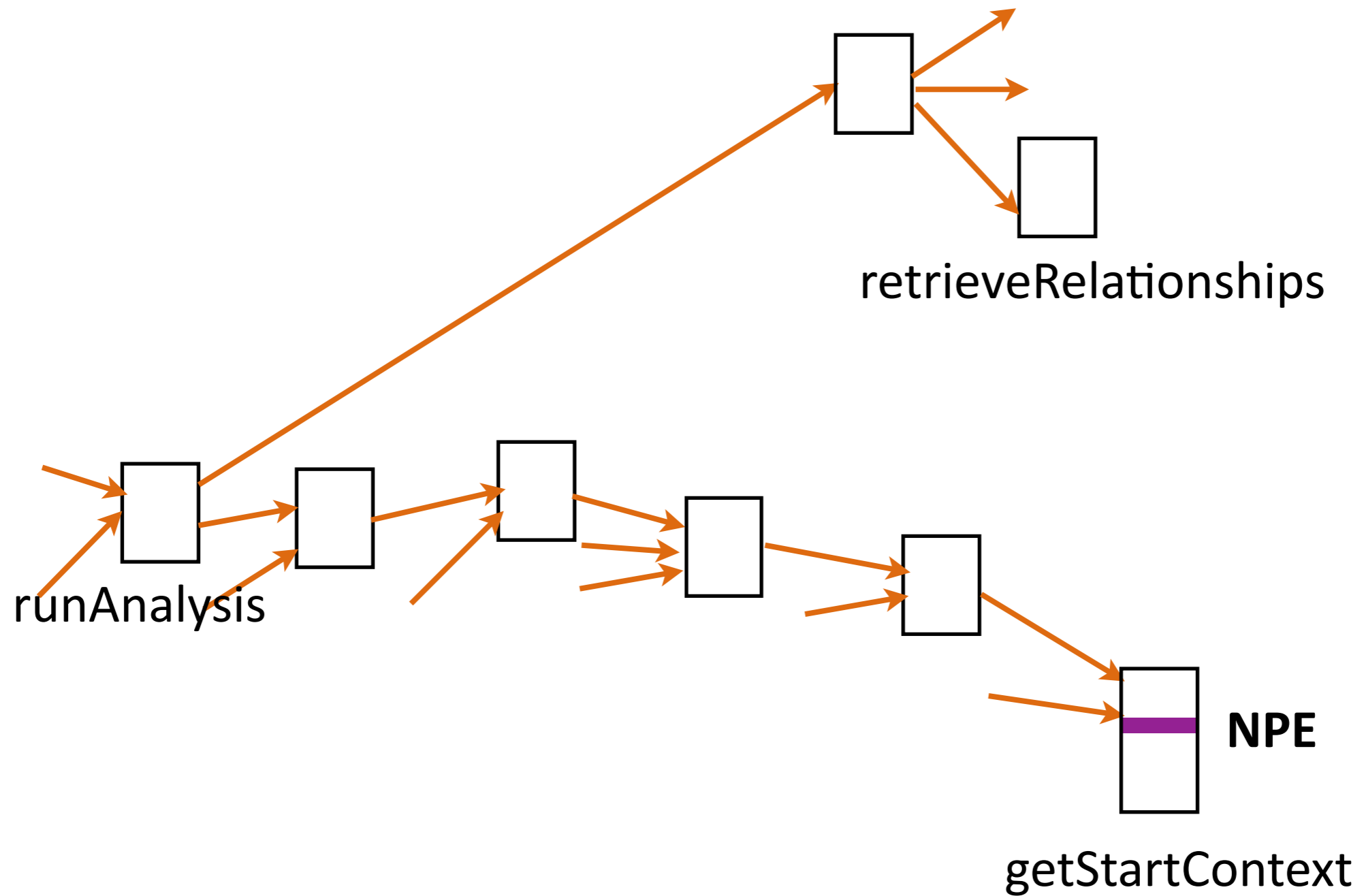
Strategy 3. **Implement & test.**

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

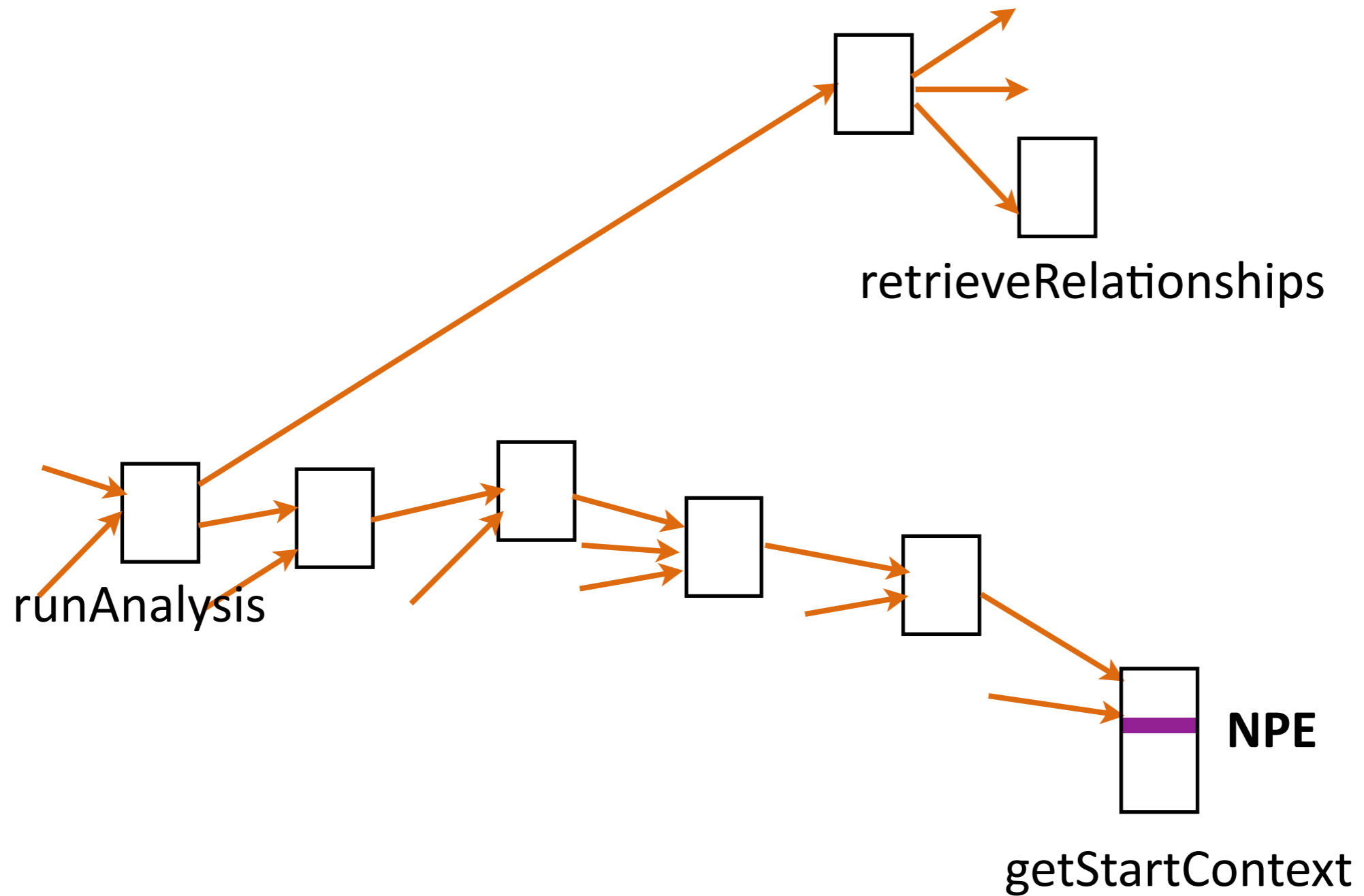
Strategy 4. **Ask my teammates.**

— *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

Thomas D. LaToza, Maryam Arab, Dastyni Loksa, and Amy J. Ko. (2020). Explicit programming strategies. *Empirical Software Engineering (ESE)*, 25, 2416–2449.

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.”

David J. Gilmore. Expert programming knowledge: A strategic approach. In *Psychology of Programming*. Elsevier, 223–234.

Amy J. Ko, Thomas D. LaToza, Stephen Hull, Ellen A. Ko, William Kwok, Jane Quichocho, Harshitha Akkaraju, and Rishin Pandit. 2019. Teaching Explicit Programming Strategies to Adolescents. In *Technical Symposium on Computer Science Education (SIGCSE '19)*, 469–475.

Thomas D. LaToza, Maryam Arab, Dastyni Loksa, and Amy J. Ko. (2020). Explicit programming strategies. *Empirical Software Engineering (ESE)*, 25, 2416–2449.



better CSS
debugging strategy

Q: Was function *F*'s implementation the ideal design, a hack, or accidental?

Strategy for answering:

1. Begin procedure *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 *RetrieveRationaleFromDevelopers*
2. Begin procedure *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 *InferRationaleFromCode*.
3. Begin procedure *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.



Debugging Strategy x +

Not Secure | programmingstrategies.org/StrategyTracker/Stra... ☆

Apps Elms payment NIW LeetCode Lexer and Parser ESEC/FSE 2018 MIT MIT Algorithms | »

Please select your Strategy

student-platform [~/Documents/Projects/StrategySharingPlatform/student-platform] - .../src/Components/Strategy/StrategyHandler/Strat

student-platform > src > Components > Strategy > StrategyHandler > StrategyHandler.js

```
302         this.setState( state: {
303             strategyId: response.data,
304             status: "saved"
305         })
306         localStorage.clear();
307     }).catch(error => {
308         console.log(error);
309     });
310 }
311 }
312
313 updateKnowledge = (array) => {
314     let uniqueItems = new Set();
315     array.forEach((item) => {
316         uniqueItems.add(item.toLowerCase())
317     });
318
319     this.setState( state: {
320         requiredKnowledge: Array.from(uniqueItems)
321     })
322     localStorage.setItem("new_requiredKnowledge", JSON.stringify(Array.from(uniqueItems)));
323 }
324
325 updateTools = async (array) => {
326     let uniqueItems = new Set();
327     let formattedAllTools = {}
328
329     Object.values(this.state.allTools).forEach((item) => {
330         formattedAllTools[item.toLowerCase()] = item;
331     })
332
333     for (let i = 0; i < array.length; i++) {
334         let item = array[i]
335         let key = item.toLowerCase()
336
337         if (formattedAllTools[key]) {
338             uniqueItems.add(formattedAllTools[key])
339         } else {
340             let success = true
341             try {
342                 await axios.post( url: "/dataManagement/technologies" data: {name: item})
```

StrategyHandler > retrieveFromLocalStorage() > updates > requiredKnowledge

6: TODO Version Control TypeScript 3.7.2 Terminal Event Log

WebStorm 2019.3.5 available: // Update... (yesterday 10:07 PM) 17 chars 670:81 LF UTF-8 4 spaces Git: branch1

STRATEGY :: strategy IDENTIFIER (IDENTIFIER+) STATEMENTS

STATEMENTS :: STATEMENT+

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

ACTION :: (word | 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 | IDENTIFIER | CALL)+

IDENTIFIER :: ' identifier '

ASSIGNMENT :: set IDENTIFIER to QUERY

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

Variables
Please separate multiple inputs with a comma

conflictedFiles

Layout.js

FOREACH :: for each IDENTIFIER in identifier STATEMENTS

FOR EACH 'file' IN '**conflictedFiles**'

```

1 # If you've spent a lot of time debugging unfamiliar code, the way that you probably debug is
2 # to first look at the failure, then look at the code to understand how it's architected, and
3 # then look for possible reasons for why the program failed. Once you have a guess, you
4 # probably then check it with things like breakpoints and logging. This strategy often works
5 # if you can have a lot of prior experience with debugging and inspecting program state. But
6 # if you don't have that experience, or you happen to guess wrong, this approach can lead to
7 # a lot of dead ends.
8 #
9 # The strategy you're about to use is different. Instead of guessing and checking, this
10 # strategy involves systematically working backwards from the code that directly caused the
11 # failed output to all of the code that caused that failed output to occur. As you work
12 # backwards, you'll check each statement for defects. If you work backwards like this,
13 # following the chain of causality from failure to cause, you will almost certainly find the
14 # bug.

```

15 **STRATEGY** debug()

```

16 # This first step will give you enough familiarity to find lines in the program that create
17 # the program's output. Read the names of all of the functions and variables in the program
18 # Some programs produce command line output with print statements.
19 # Is the faulty output you're investigating printed to a command line?
20 IF the faulty output is logged to a command line
21     # To find print statements, try searching for keywords related to 'log' or 'print'
22     SET outputLines TO the line numbers of calls to console logging functions
23 # Graphical output includes things like colored lines and rectangles
24 IF the faulty output is graphical output
25     # To find these lines, try searching for keywords related to graphical output, like '
26     # draw' or 'fill'. Focus on lines that directly render something, not on higher-level
27     # functions that indirectly call rendering functions.
28     SET outputLines TO the line numbers of function calls that directly render graphics to
29     the screen
30 # Now that you have some lines that could have directly produced the faulty output, you're
31 # going to check each line, see if it executed, and then find the cause of it executing. If
32 # you're lucky, you only have one output line to check.
33 FOR EACH 'line' IN 'outputLines'
34     # First, let's make sure the line executed. You want to be sure that this is actually the
35     # source of the wrong output. You can check this by inserting a logging statement, or
36     # setting a breakpoint on the line.
37     IF the program executed 'line'
38         Analyze the line to determine its role in the overall behavior of the program
39         # Check for errors such as the wrong function being called, the wrong argument being
40         # passed to a function, the wrong variable being referenced, or a wrong operator being

```

Strategy: Design task

		Self-guided	Guided
Template	Found and used example code as a template for implementation.	4/14 (29%)	0/14 (0%)
Decompose	Analyzed functional requirements for sub-problems, implementing each independently	9/14 (64%)	0/14 (0%)
TDD	Translated functional requirements into test cases, identifying sub-problems from test case requirements.	2/14 (14%)	11/14 (79%)

Strategy: Debugging task

		Self-guided	Guided
Guess & check	Participants found suspicious lines of code, modifying them and checking the effects of their modification.	4/14 (29%)	0/14 (0%)
Forward search	Participants identified where the program began processing input, following its execution forward	9/14 (64%)	0/14 (0%)
Backward search	Participants identified faulty output and worked backwards through control and data flow dependencies	2/14 (14%)	11/14 (79%)

Thomas D. LaToza, Maryam Arab, Dastyni Loksa, and Amy J. Ko. (2020). Explicit programming strategies. *Empirical Software Engineering (ESE)*, 25, 2416–2449.

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

$p < 0.004^*$

Thomas D. LaToza, Maryam Arab, Dastyni Loksa, and Amy J. Ko. (2020).
Explicit programming strategies. *Empirical Software Engineering (ESE)*,
25, 2416–2449.

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 property 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, pseudo
class selector | type selector and pseudo element
For easy explanation: use this url: <http://qanimate.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

Maryam Arab, Thomas D. LaToza, Jenny Liang, Amy J. Ko. (2022). An exploratory study of sharing strategic programming knowledge. Conference on Human Factors in Computing Systems (CHI), 1-15.

- 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

Maryam Arab, Thomas D. LaToza, Jenny Liang, Amy J. Ko. (2022). An exploratory study of sharing strategic programming knowledge. Conference on Human Factors in Computing Systems (CHI), 1-15.



code interacting with framework

search online forum

create diagrams

likelihood
(odds
ratio)

3.84

(3.84x more
likely)

0.51

(0.51x less
likely)

Cassandra Bailey. The Impact of Affect, Scenario and Task Characteristics on Developer Decision-Making. (2020). Masters Thesis, George Mason University.

feeling stressed / nervous (LVHA)

add print statements

read surrounding code

likelihood
(odds
ratio)

2.42

(2.42x more
likely)

0.17

(0.17x less
likely)

Cassandra Bailey. The Impact of Affect, Scenario and Task Characteristics on Developer Decision-Making. (2020). Masters Thesis, George Mason University.

feeling sad / depressed (LVLA)

experiment with edits

likelihood
(odds
ratio)

0.09
(0.09x less
likely)

Cassandra Bailey. The Impact of Affect, Scenario and Task Characteristics on Developer Decision-Making. (2020). Masters Thesis, George Mason University.

feeling excited / enthusiastic (HVHA)

ask for help from a colleague

likelihood
(odds
ratio)

2.13

(2.13x more
likely)

Cassandra Bailey. The Impact of Affect, Scenario and Task Characteristics on Developer Decision-Making. (2020). Masters Thesis, George Mason University.

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

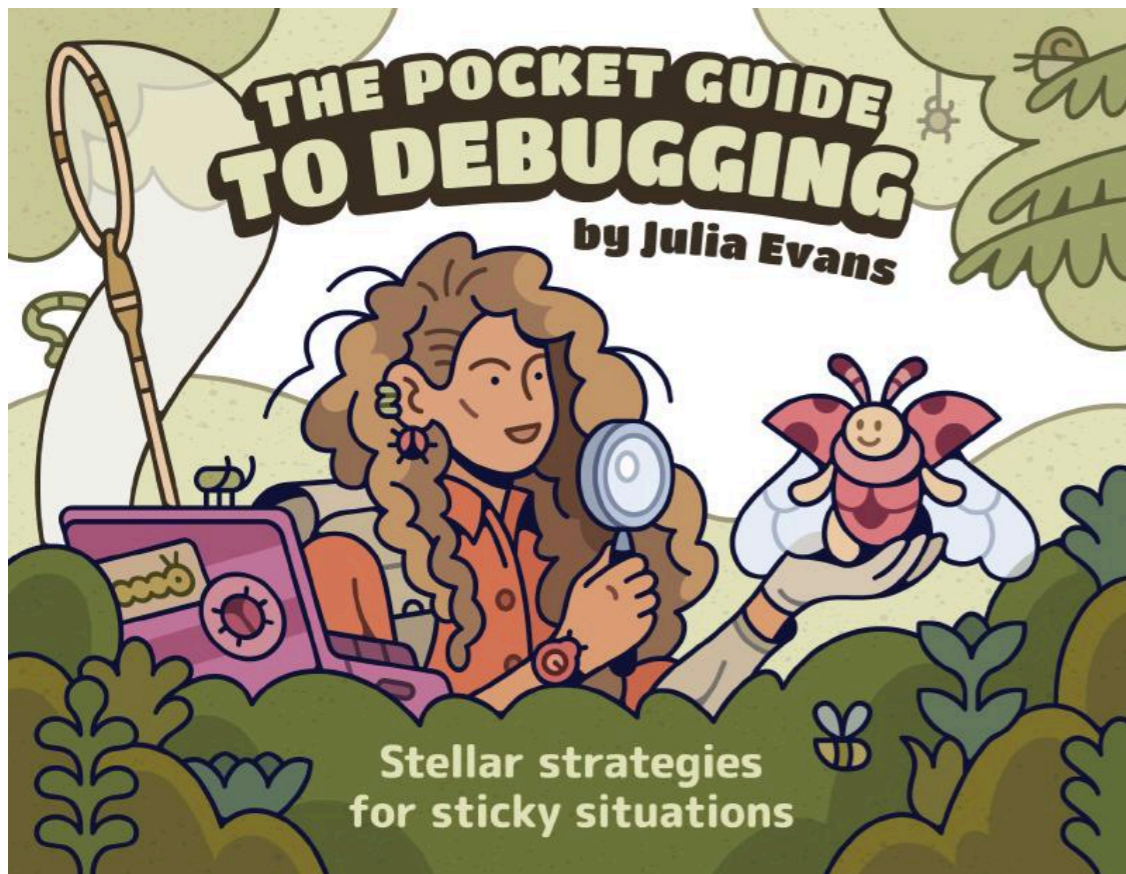
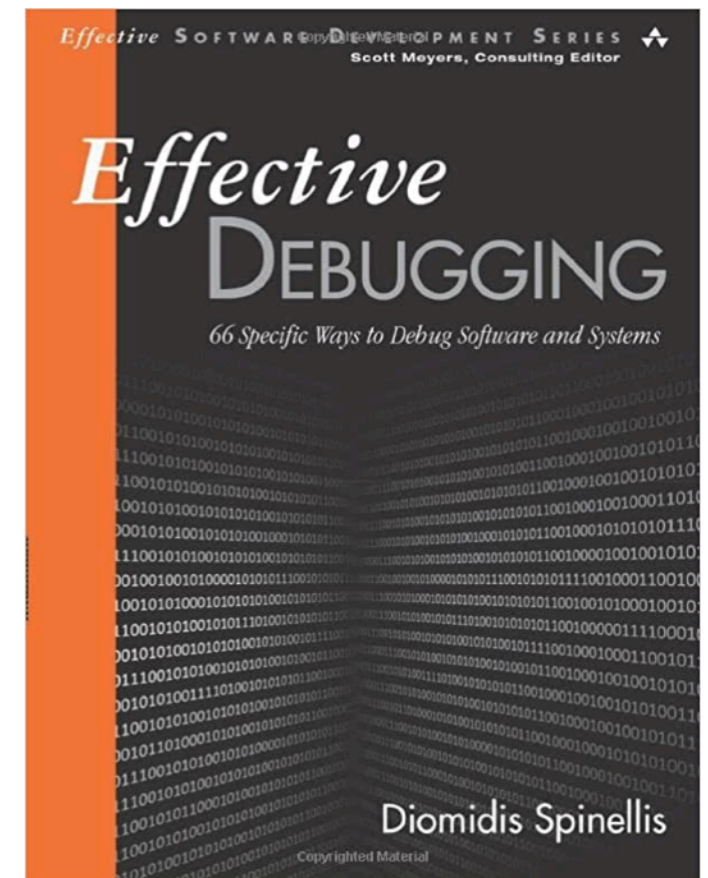


table of contents

manifesto.....	6-7	⑤ simplify 📄➔	
① first steps 🐛➔		write a tiny program.....	38
preserve the crime scene.....	9	one thing at a time.....	39
read the error message.....	11	tidy up your code.....	40
reread the error message.....	11	delete the buggy code.....	41
reproduce the bug.....	12	reduce randomness.....	42
inspect unreproducible bugs.....	13		
identify one small question.....	14	⑥ get unstuck 🔄➔	
retrace the code's steps.....	15	take a break.....	44
write a failing test.....	16	investigate the bug together.....	45
		timebox your investigation.....	46
② get organized 📁		write a message asking for help... 47	
brainstorm some suspects.....	18	explain the bug out loud.....	48
rule things out.....	19	make sure your code is running... 49	
keep a log book.....	20	do the annoying thing.....	50
draw a diagram.....	21		
		⑦ improve your toolkit 📦	
③ investigate 🔍		try out a new tool.....	52
add lots of print statements.....	23	types of debugging tools.....	53
use a debugger.....	24	shorten your feedback loop.....	54
jump into a REPL.....	25	add pretty printing.....	55
find a version that works.....	26	colours, graphs, and sounds.....	56
look at recent changes.....	27		
add assertions everywhere.....	28	⑧ after it's fixed 🏆	
comment out code.....	29	do a victory lap.....	58
analyze the logs.....	30	tell a friend what you learned.....	59
		find related bugs.....	60
④ research 📖		add a comment.....	61
read the docs.....	32	document your quest.....	62
find the type of bug.....	33		
learn one small thing.....	34		
read the library's code.....	35		
find a new source of info.....	36		



be more effective with sharing strategies

Home Search Request Strategy New Strategy Roboto Tutorial About Us Report a problem

Strategies for How do I debug frontend web UI code?

I would like to debug code bugs that occur on web browser UIs. [Add strategy](#)

- Debugging HTML in Chrome**
written by Amy J. Ko
The main approach to this strategy is to use the Chrome inspector to try to understand how the browser interpreted your HTML, then use that to infer the problem in your HTML.
Technologies: HTML, Chrome Inspector
Experiences: fast
- How to debug CSS to fix a visual defect**
written by Maryam Arab
This strategy helps developers fix the issue of an element with an undesired visual/position style.
Technologies: CSS
Experiences: fast, Clear
- Debug HTML**
written by Rob Thompson
Isolate where and why there might be an issue in your HTML code.
Technologies: Web browser, Code editor
Experiences: fast
- Debug CSS**
written by Rob Thompson
Technologies: Code editor, Web browser
Experiences: fast
- Debugging JavaScript**
written by Rob Thompson
Identify the problem and isolate it using print commands and debugging tools.
Technologies: JavaScript, Web browser, Code editor, Browser developer tools
Experiences: fast, Efficient

Home Search Request Strategy New Strategy Roboto Tutorial About Us Report a problem

Debugging HTML in Chrome

By Amy J. Ko

The main approach to this strategy is to use the Chrome inspector to try to understand how the browser interpreted your HTML, then use that to infer the problem in your HTML.

🔔 [Request](#) for your desired strategy at any point if you have difficulties performing some actions described in the steps. Our team is ready to help you with your desired problem solutions in the form of strategies.

Strategy Category: **debug**
Technology: HTML, Chrome Inspector

💡 Click 'Start Strategy' button to try using the strategy.

[Start Strategy>>](#)

💡 Read the strategy, and click on lines to keep track of where you're at. Use Keyboard **↑** or **↓** to move up and down each line.

💡 Add comments or questions for the statements that you don't understand, have questions about, need more description, and any other kinds of comments by **🗨** on each line.

Strategy DebugHTMLWithChrome

- SET** expectation **TO** what you expected to see **🗨**
Write down what you expected
- SET** reality **TO** what you see in the browser
- Right click on the element on the web page you did not expect
- Choose the inspect menu item
- Inspect the structure of the selected element
- Inspect the tag that contains the selected element
- Inspect the tags before and after the selected element
- IF** the structure you see don't match the structure you wrote
Look for an HTML syntax error such as a missing or malformed angle bracket
- RETURN** Ask a peer or TA or instructor to see if they notice a problem that explains the problem

[Add General Comment](#)

Take Notes:
Please use this area to take note. (Unfortunately, this note will not be stored)

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