User-Centered Design

SWE 632
Spring 2022

GEORGE MASON UNIVERSITY

Administrivia

- *HW1 due next Tuesday before class*

What We Learned & Looking Ahead

- Examined human cognition
- Have 2 ways to identify usability issues (Heuristics & Principles)
- But... is HCI just identifying usability issues?
- What does ***design*** mean?
- How do we learn about user ***needs***?
- How do we build designs?
- How do we evaluate designs?

Overview of User-Centered Design

In Class Discussion

- ***Today's question:***
    - What does *user-centered design* mean to you?

User-centered design

User-centered design

Who are the users?

How does the product fit into the broader context of their lives?

What are the user's needs?

What problems may users encounter w/ current ways of doing things?

What are the user's tasks and goals?

What extreme cases may exist?

Technology-Centered Design

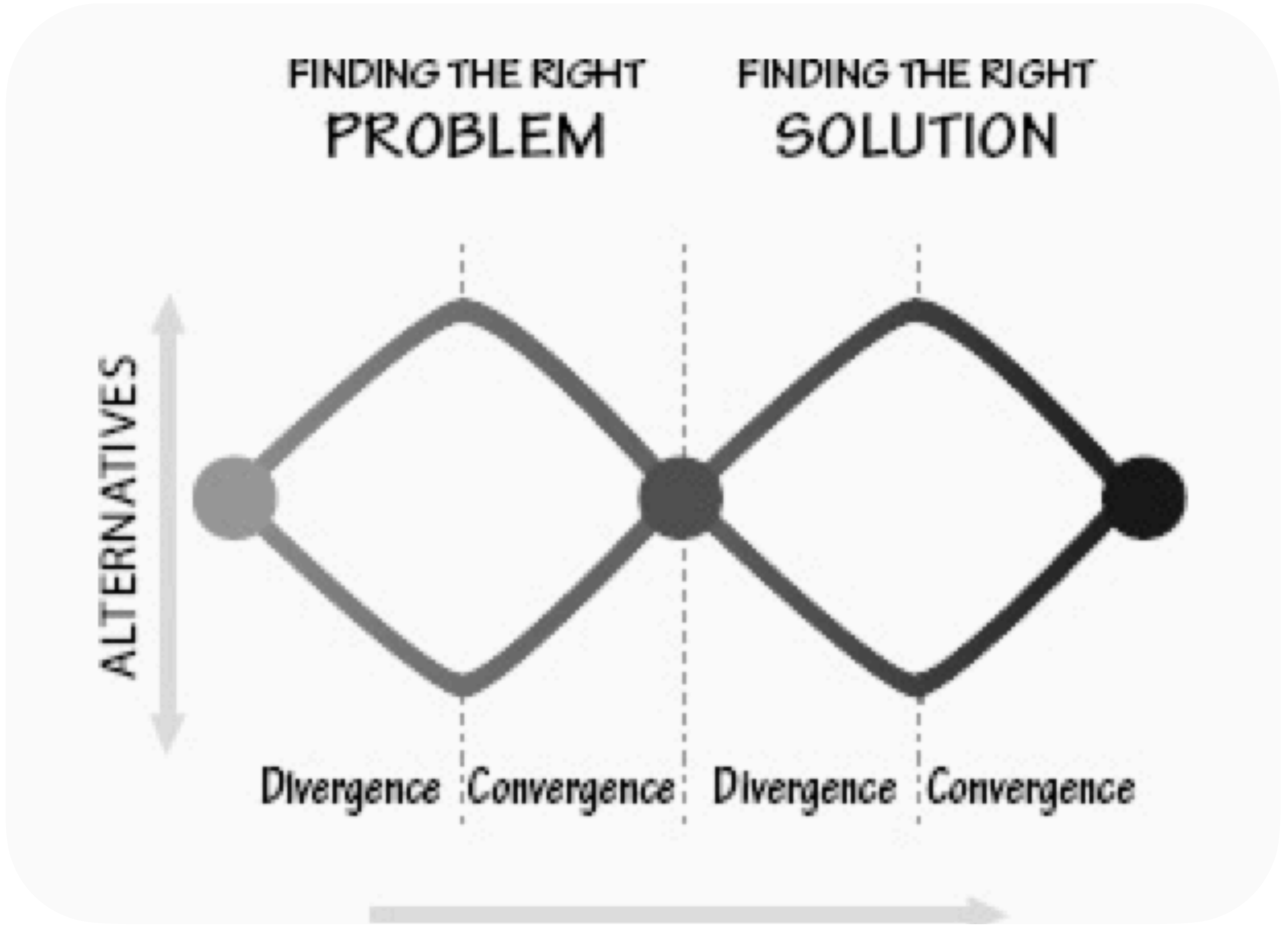
What can this technology do?

How might users use it?

What features does it have?

Double Diamond Model of Design

- Question problem, expand scope, discover fundamental issues
- Converge on problem
- Expand possible solutions
- Converge on solution



Iterative Model of Design

Observation

(Re)Define the Problem

Understand User Needs

Idea Generation

Brainstorm
what to build

Prototype

Build

Test

Evaluate what
you have built

Iteration, Iteration, Iteration

- Repeated study and testing
- Use tests to determine what is working or not working
- Determine what the problem might be, redefining the problem
- Collect more data
- Generate new alternatives

Observation

Needfinding (a.k.a. design research)

- Goal: understand user’s needs
- Use of methods to gather qualitative data
  - behaviors, attitudes, aptitudes of potential and existing users
  - technical, business, and environmental contexts - domain
  - vocabulary and social aspects of domain
  - how existing products used
- Empowers team w/ credibility and authority, helping inform decisions

Needfinding vs. market research

Needfinding

- What users really need
- How they will really use product
- Qualitative methods to study in depth
- Small numbers of participants

Market research

- Who might purchase item
- What factors influence purchasing
- Quantitative studies w/ focus groups, surveys
- Large numbers of participants

Example

- Cooper conducted a user study for entry-level video editing product
- Company built professional software, looking to move into consumer software
  - Help connect those w/ computers and video cameras
- Found strongest desire for video editing was parents
- Found 1/12 had successfully connected camera, using work IT guy

Solving the correct problem

- Practices may sometimes mask deeper problems
- ***Goal:*** uncover layers of practices to understand how problems emerge

Interviews

- May include bother current users and potential users w/ related needs
- Questions
  - context of how product fits into lives or work
  - when, why, how is or will product be used
  - what do users need to know to do jobs?
  - current tasks and activities, including those not currently supported
  - goals and motivations of using product
  - problems and frustrations with current products or systems

Observations

- Most incapable of accurately assessing own behaviors
- May avoid talking about problems to avoid feeling dumb
- Observing yields more accurate data
- Capture behaviors: notes, pictures, video (if possible)

Contextual inquiry

- Method that includes both interviews and observations
- Next week’s lecture

Idea Generation

Creativity

- What's the most creative software app you've seen?
- What made it creative?

Ideation

- Process of generating, developing, communicating new **ideas**
- Guidelines and best practices
  - Generate ***numerous*** ideas
  - Number ideas
  - Avoid premature dismissal of ideas
  - Sharpen the **focus** - pose the right problem
  - Build and jump - build to keep momentum on ideas, jump when theme tapers out

Prototyping

Prototyping - Building Quickly

- Build quick prototype or mock-up of each potential solution
- “Wizard of Oz” Studies
- Mainly performed to ensure the problem is well understood

Testing

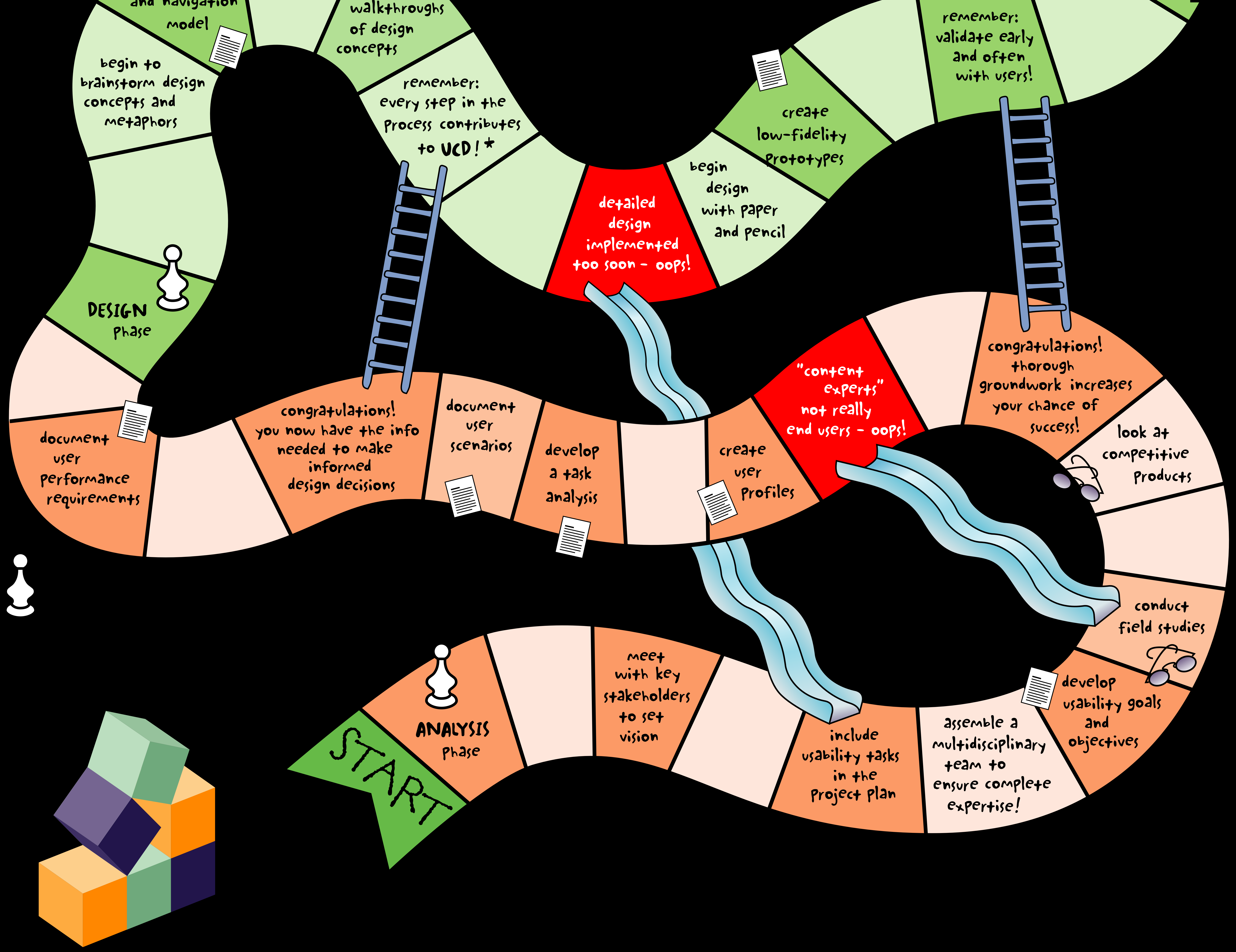
Testing - User Centered Evaluation

- Test with population similar to target population
- Have them use prototypes as close as possible to intended
- If possible, have two people use a prototype, one guiding the other's use.
- More on this in a future lecture…

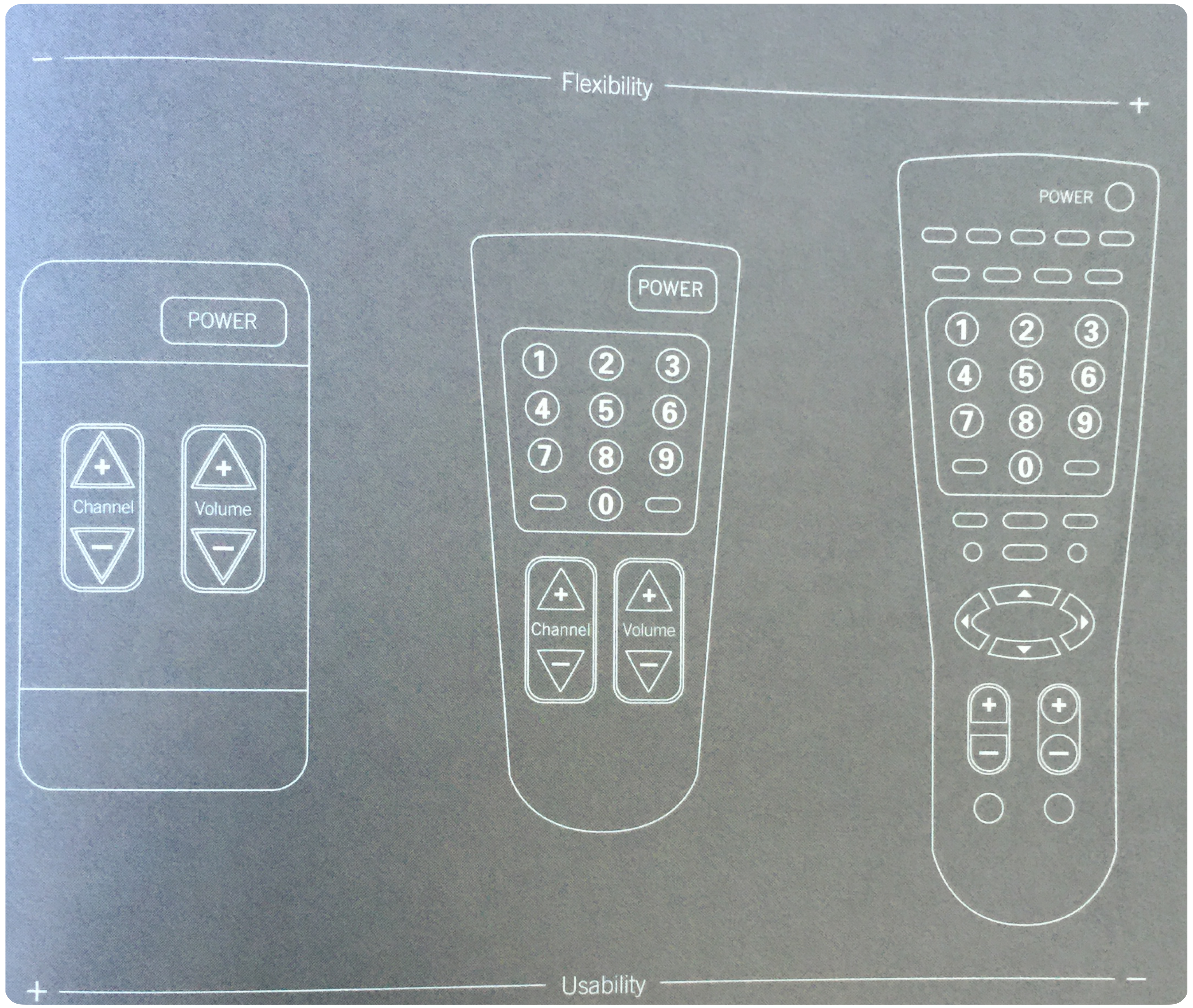
User-Centered Design Considerations

Fail Fast

- *"Fail frequently, fail fast"* David Kelley, founder of Ideo
- Failure is ***learning*** experience
- Crucial to understand correct ***problem*** to solve & ensure solution is appropriate
- Abstract requirements are invariably wrong
- Requirements produced by asking people what they want are wrong


START
ANALYSIS Phase
meet with key stakeholders to set vision
include usability tasks in the project plan
assemble a multidisciplinary team to ensure complete expertise!
develop usability goals and objectives
conduct field studies
look at competitive products
congratulations! thorough groundwork increases your chance of success!
"content experts" not really end users - oops!
create user profiles
develop a task analysis
document user scenarios
congratulations! you now have the info needed to make informed design decisions
document user performance requirements
DESIGN Phase
begin to brainstorm design concepts and metaphors
and navigation model
walkthroughs of design concepts
remember: every step in the process contributes to UCD! *
detailed design implemented too soon - oops!
begin design with paper and pencil
create low-fidelity prototypes
remember: validate early and often with users!

Flexibility-usability tradeoff

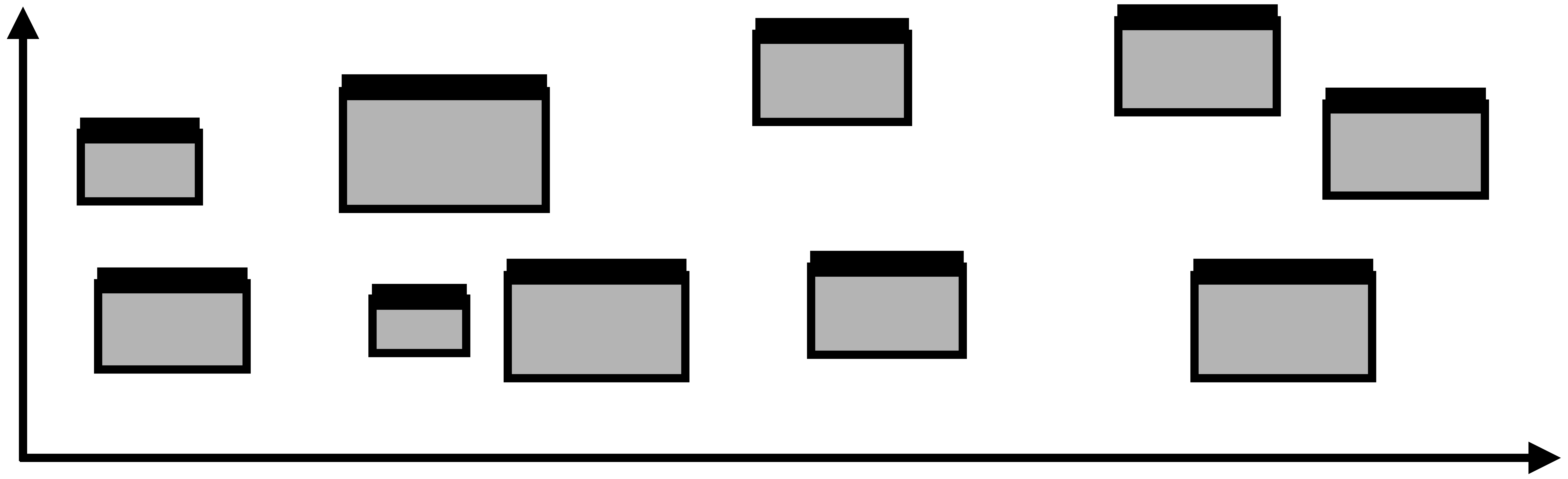


Flexibility-Usability Tradeoff

- Jack of all trades, master of none
- Better understanding needs enables specialization and ***optimization*** for common cases
- System evolution over time:
    - flexibility —> specialization

Examples of flexibility / usability tradeoff?

Navigating Design Space



- What are key decisions in interaction design?
- What alternatives are possible?
- What are tradeoffs between these alternatives?

Hierarchy of Design Decisions

- What are you (*re*)designing?
  - The width of the text input
  - The maximum length of a valid username
  - When in the signup process users enter their username
  - If the user must create a username when signing up
  - Whether users are anonymous or have a login
  - If users can interact with other users in your application

Picking the Right Level of Redesign

- Where are the user's pain points
- What are the underlying causes
- What would be the value to the user of addressing issue
- What do you have time to build (or change)

Example - iPod

- Supports entire activity of listening to music
  - discovering music
  - purchasing music
  - getting it into music player
  - developing playlists
  - sharing playlists
  - listening to music
  - ecosystem of external speakers and accessories

Example of a Design Process

- How do you get from let's make listening to music better to designing an iPod??
- Iterative design...
  - But what does that actually look like more concretely?
  - What insights into activity help inspire design?
  - How does watching users help lead to these insights?
  - How do insights translate into an actual real design?
  - How do know the new design is actually better?

In-Class Activity

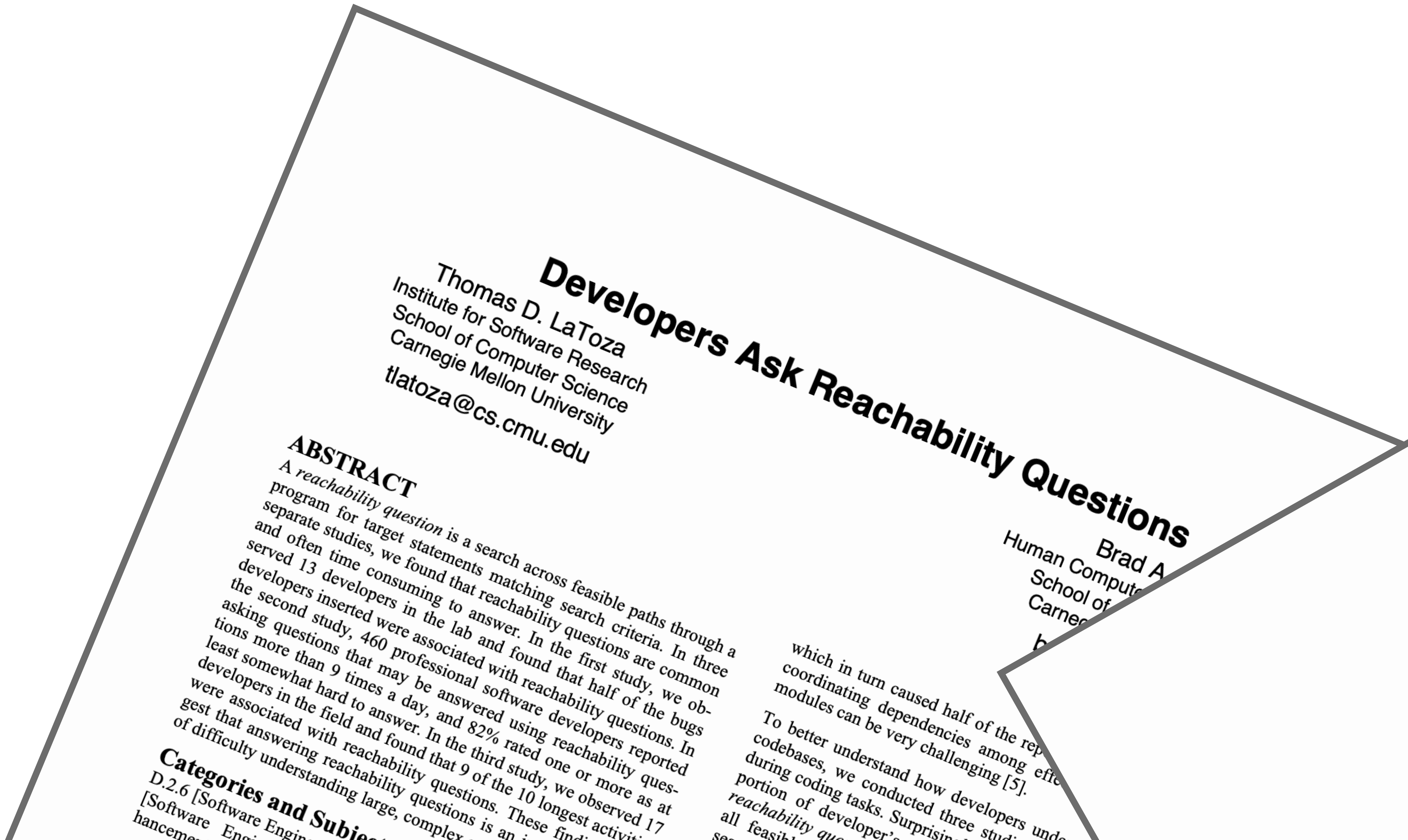
- Redesign PatriotWeb
- Consider: at what level are you redesigning it? What's the problem (at this level)? How are you making it better?

Example

Domain: Debugging

- ***Design goal:*** how do we better support activity of debugging in large, complex codebases?
- Build a better debugging tool (?)
    - What should it do? How would it help?
        - Design a better watch window? Support new types of breakpoints?
    - What's really the key steps in debugging that lead users to struggle the most?

Domain: Debugging



Developers Ask Reachability Questions

Thomas D. LaToza
Institute for Software Research
School of Computer Science
Carnegie Mellon University
tlatoza@cs.cmu.edu

Brad A
Human Compute
School of
Carneg
b

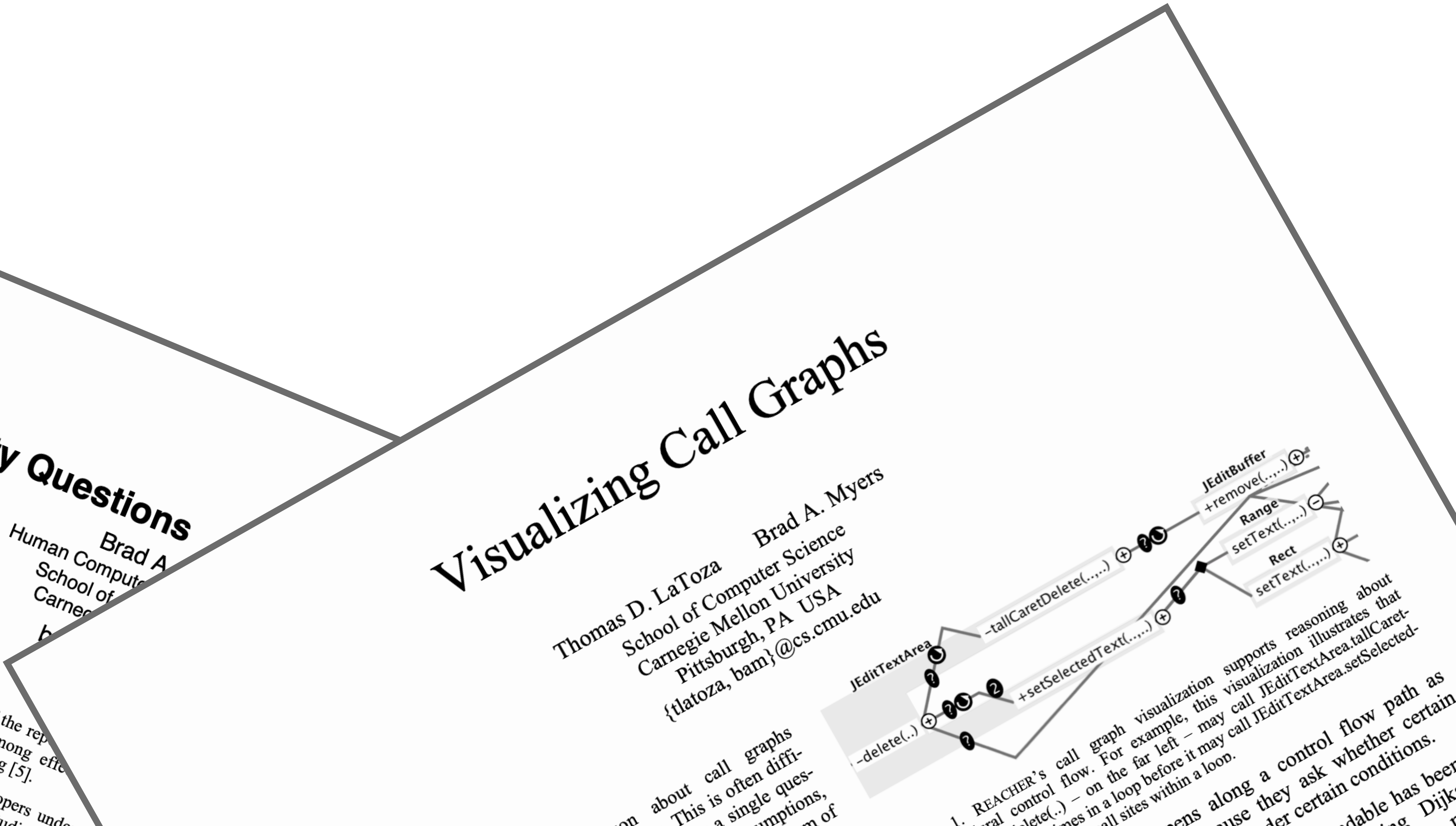
ABSTRACT

A *reachability question* is a search across feasible paths through a program for target statements matching search criteria. In three separate studies, we found that reachability questions are common and often time consuming to answer. In the first study, we observed 13 developers in the lab and found that half of the bugs developers inserted were associated with reachability questions. In the second study, 460 professional software developers reported asking questions that may be answered using reachability questions more than 9 times a day, and 82% rated one or more as at least somewhat hard to answer. In the third study, we observed 17 developers in the field and found that 9 of the 10 longest activities were associated with reachability questions. These findi gest that answering reachability questions is an i of difficulty understanding large, complex

Categories and Subjec
D.2.6 [Software Engi
[Software Engi
hanceme

which in turn caused half of the rep coordinating dependencies among effe modules can be very challenging [5].

To better understand how developers unde codebases, we conducted three studi during coding tasks. Surprisin portion of developer's *reachability que* all feasib se



Visualizing Call Graphs

Thomas D. LaToza Brad A. Myers
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA USA
{tlatoza, bam}@cs.cmu.edu

JEditBuffer
+remove(..,..)
Range
setText(..,..)
Rect
setText(..,..)
-tallCaretDelete(..,..)
JEditTextArea
+setSelectedText(..,..)
-delete(..)

on about call graphs
This is often diffi-
a single ques-
umptions,
m of

1. REACHER's call graph visualization supports reasoning about
eral control flow. For example, this visualization illustrates that
delete(..) – on the far left – may call JEditTextArea.tallCaret-
imes in a loop before it may call JEditTextArea.setSelected-
all sites within a loop.

pens along a control flow path as
use they ask whether certain
der certain conditions.
dable has been
Dijk

Observing Developers

Participants

17 professional developers

Tasks

~90 minutes
picked one of ***their*** own coding tasks involving unfamiliar code

Transcripts

Interesting. This looks like, this looks like the code is approximately the same but it's refactored. But the other code is.

Changed what flags it's ???

He added a new flag that I don't care about. He just renamed a couple things.

Well.

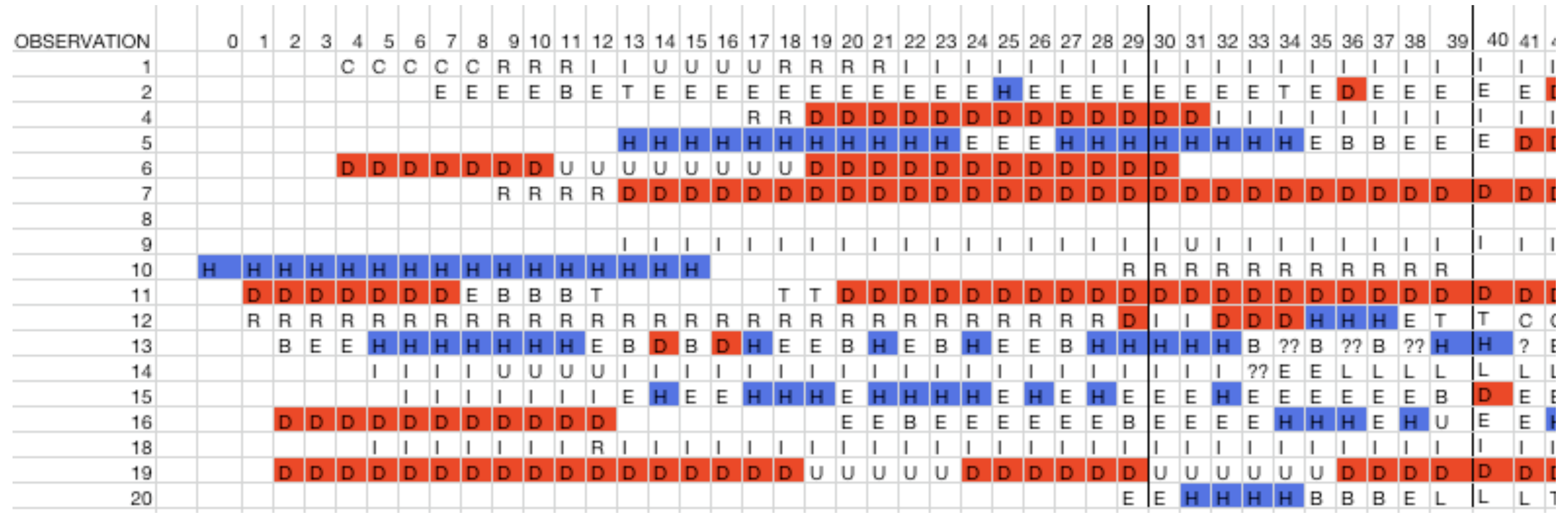
So the change seemed to have changed some of the way these things are registered,

but I didn't see anything that talked at all about whether the app is running or whether the app is booted. So it seems like, this was useless to me.

(annotated with observer notes about goals and actions)

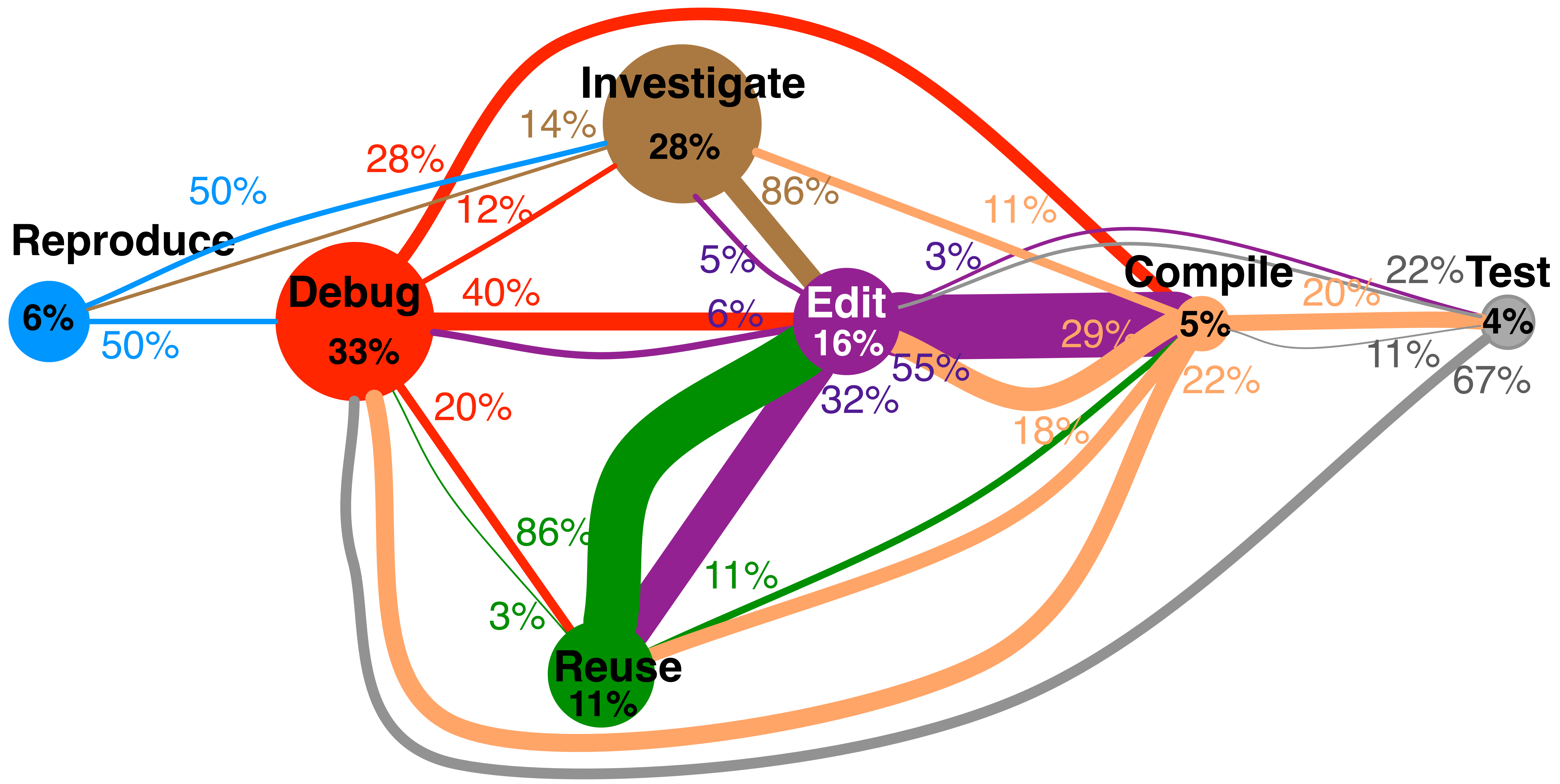
(386 pages)

Activities



OBSERVATION	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	4
1					C	C	C	C	C	R	R	R	I	I	U	U	U	U	R	R	R	R	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
2								E	E	E	E	B	E	T	E	E	E	E	E	E	E	E	E	E	E	H	E	E	E	E	E	E	E	E	T	E	D	E	E	E	E	E	D
4																		R	R	D	D	D	D	D	D	D	D	D	D	D	D	D	I	I	I	I	I	I	I	I	I	I	I
5														H	H	H	H	H	H	H	H	H	H	H	E	E	E	H	H	H	H	H	H	H	H	E	B	B	E	E	E	D	D
6					D	D	D	D	D	D	D	U	U	U	U	U	U	U	U	D	D	D	D	D	D	D	D	D	D	D	D												
7										R	R	R	R	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
8																																											
9														I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	U	I	I	I	I	I	I	I	I	I	I	I
10	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H														R	R	R	R	R	R	R	R	R	R	R			
11		D	D	D	D	D	D	D	E	B	B	B	T						T	T	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
12		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	D	I	I	D	D	D	H	H	H	E	T	T	C	C
13			B	E	E	H	H	H	H	H	H	H	E	B	D	B	D	H	E	E	B	H	E	B	H	E	E	B	H	H	H	H	H	B	??	B	??	B	??	H	H	?	E
14						I	I	I	I	U	U	U	U	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	??	E	E	L	L	L	L	L	L	L
15							I	I	I	I	I	I	I	E	H	E	E	H	H	H	E	H	H	H	H	E	H	E	H	E	E	E	H	E	E	E	E	E	E	B	D	E	E
16			D	D	D	D	D	D	D	D	D	D	D								E	E	B	E	E	E	E	E	E	B	E	E	E	E	H	H	H	E	H	U	E	E	H
18						I	I	I	I	I	I	I	R	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
19			D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	U	U	U	U	U	D	D	D	D	D	D	U	U	U	U	U	U	D	D	D	D	D	D	D
20																														E	E	H	H	H	H	B	B	B	E	L	L	L	T

Coding Activities



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

Longest Activities: Control Flow

4 out of the 5 longest investigation activities

Primary question	**Time** (m)	**Related control flow question**
How is this data structure being mutated in this code?	83	Search downstream for **writes** to data structure
"Where [is] the code assuming that the tables are already there?"	53	**Compare** behaviors when tables are or are not loaded
How [does] application state change when *m* is called denoting startup completion?	50	Find field **writes** caused by m
"Is [there] another reason why *status* could be non-zero?"	11	Find statements through which values **flow** into status

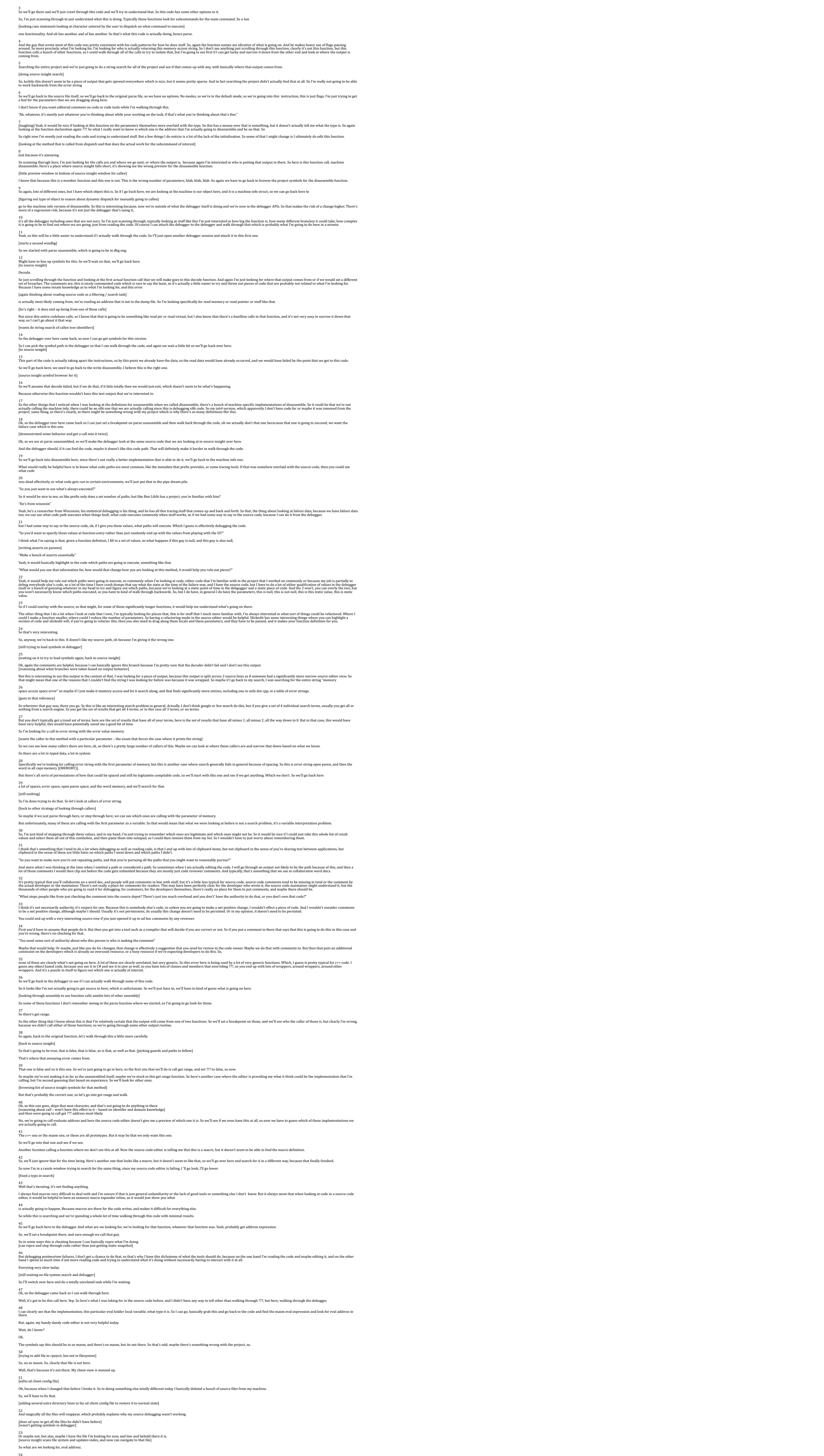
5 out of the 5 longest debugging activities

Where is method *m* generating an error?	66	Search downstream from *m* for **error** text
What resources are being acquired to cause this deadlock?	51	Search downstream for **acquire** method calls
"When they have this attribute, they must use it somewhere to generate the content, so where is it?"	35	Search downstream for **reads** of attribute
"What [is] the test doing which is different from what my app is doing?"	30	**Compare** test traces to app traces
How are these thread pools interacting?	19	Search downstream for **calls** into thread pools

Longest Debugging Activities

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

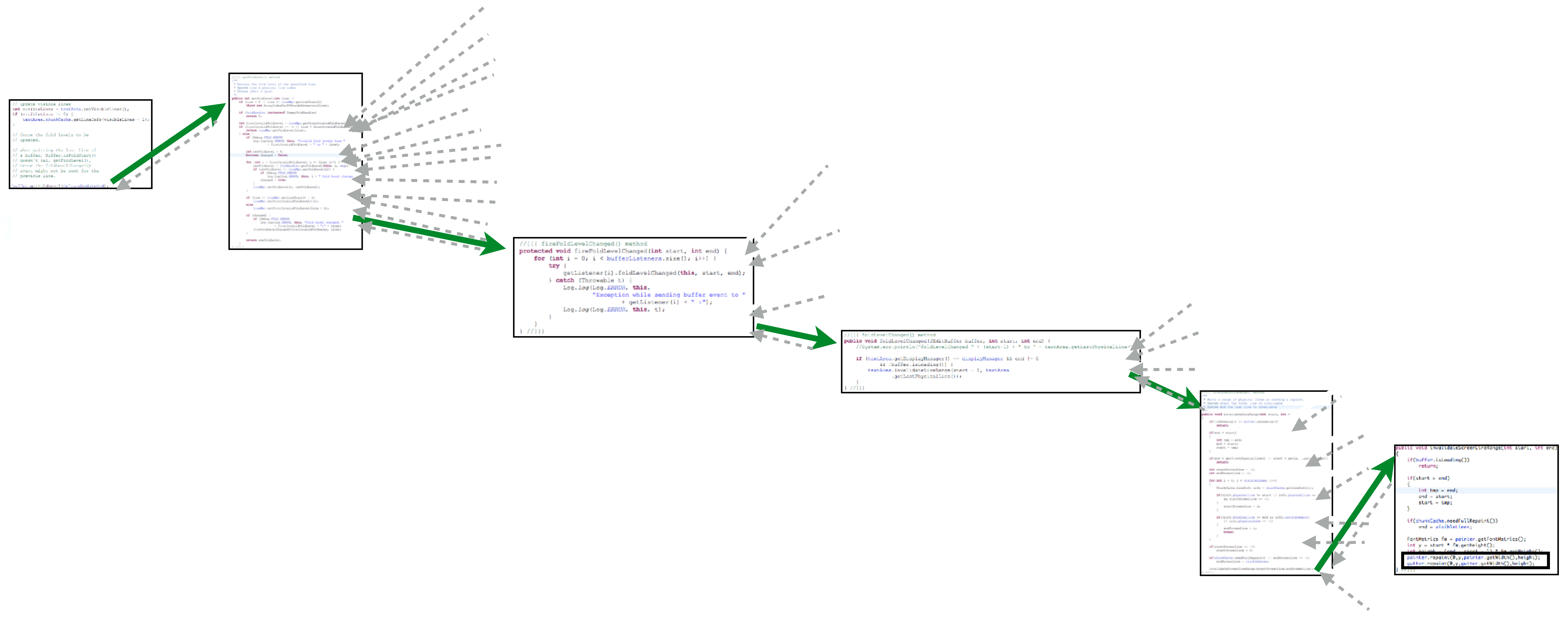
(66 minutes)



Why was this Hard to Answer?

Hard to pick the ***control flow path*** that leads from starting point to target

Guess and check: which path leads to the target?



```
[illegible]
```

```
[illegible]
```

```
//{{{ fireFoldLevelChanged() method
protected void fireFoldLevelChanged(int start, int end) {
    for (int i = 0; i < bufferListeners.size(); i++) {
        try {
            getListener(i).foldLevelChanged(this, start, end);
        } catch (Throwable t) {
            Log.log(Log.ERROR, this,
                    "Exception while sending buffer event to "
                        + getListener(i) + " :");
            Log.log(Log.ERROR, this, t);
        }
    }
} //}}}
```

```
//{{{ foldLevelChanged() method
public void foldLevelChanged(JEditBuffer buffer, int start, int end) {
    //System.err.println("foldLevelChanged " + (start-1) + " to " + textArea.getLastPhysicalLine
    
    if (textArea.getDisplayManager() == displayManager && end != 0
            && !buffer.isLoading()) {
        textArea.invalidateLineRange(start - 1, textArea
                .getLastPhysicalLine());
    }
} //}}}
```

```
[illegible]
```

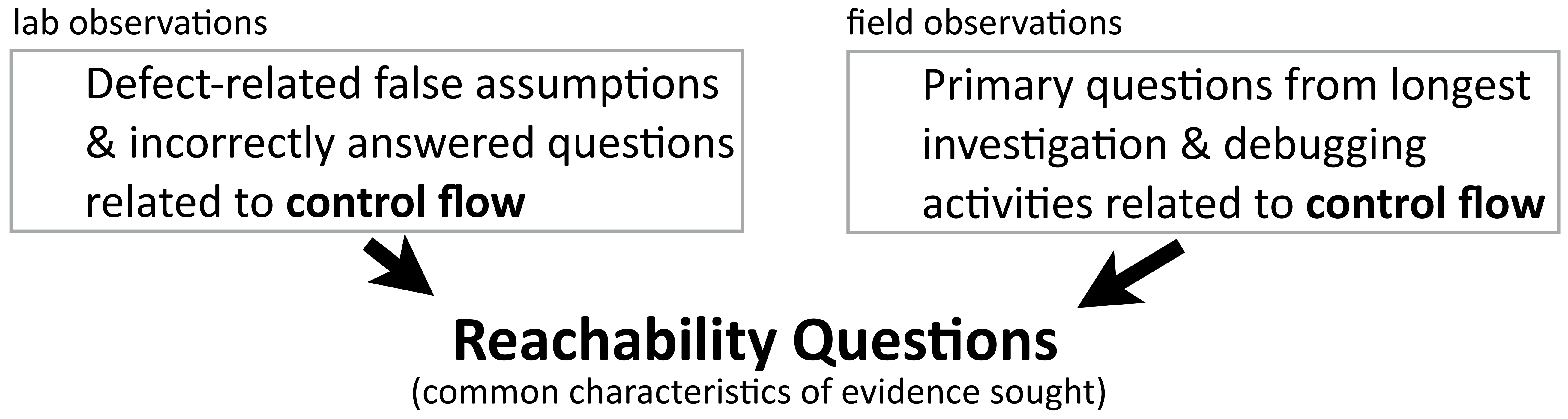
```
public void invalidateScreenLineRange(int start, int end)
{
    if(buffer.isLoading())
        return;

    if(start > end)
    {
        int tmp = end;
        end = start;
        start = tmp;
    }

    if(chunkCache.needFullRepaint())
        end = visibleLines;

    FontMetrics fm = painter.getFontMetrics();
    int y = start * fm.getHeight();
    [illegible]
    painter.repaint(0,y,painter.getWidth(),height);
    gutter.repaint(0,y,gutter.getWidth(),height);
} //}}}
```

Why are Control Flow Questions Common?

Helps answer questions about:

Causality	What does this do? What causes this to happen?
Ordering	Does A happen before B?
Choice	Does x always occur? In which situations does x occur?

When scattered across a codebase, finding statements to answer these questions can be hard.


lab observations
Defect-related false assumptions & incorrectly answered questions related to **control flow**
field observations
Primary questions from longest investigation & debugging activities related to **control flow**
Reachability Questions
(common characteristics of evidence sought)

lab observations

Defect-related false assumptions & incorrectly answered questions related to **control flow**

field observations

Primary questions from longest investigation & debugging activities related to **control flow**

Reachability Questions

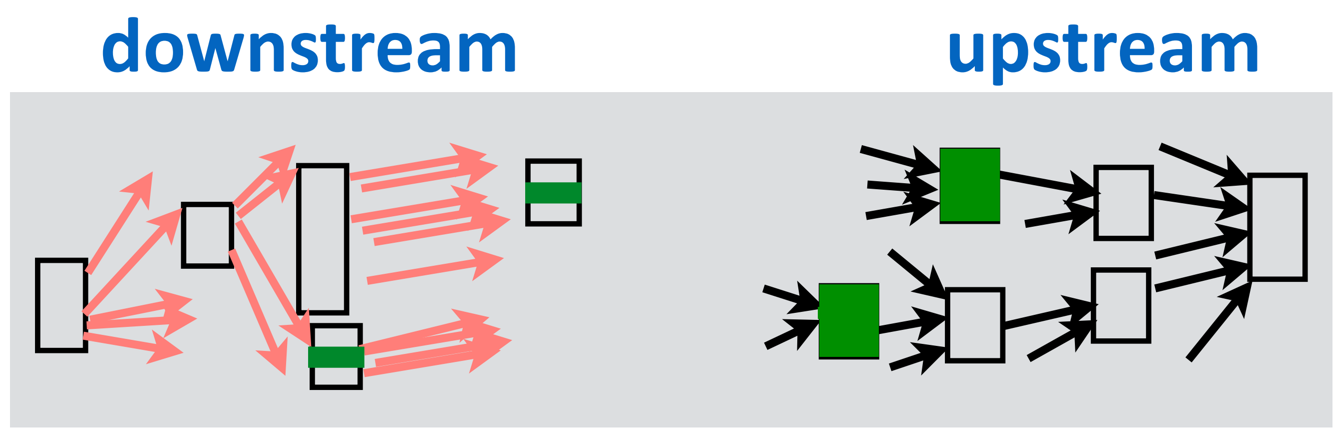
(common characteristics of evidence sought)

A search along **feasible paths** **downstream** or **upstream** from a statement for **target statements** matching **search criteria**

feasible paths $\cap$ statements matching search criteria

feasible paths

filter compare

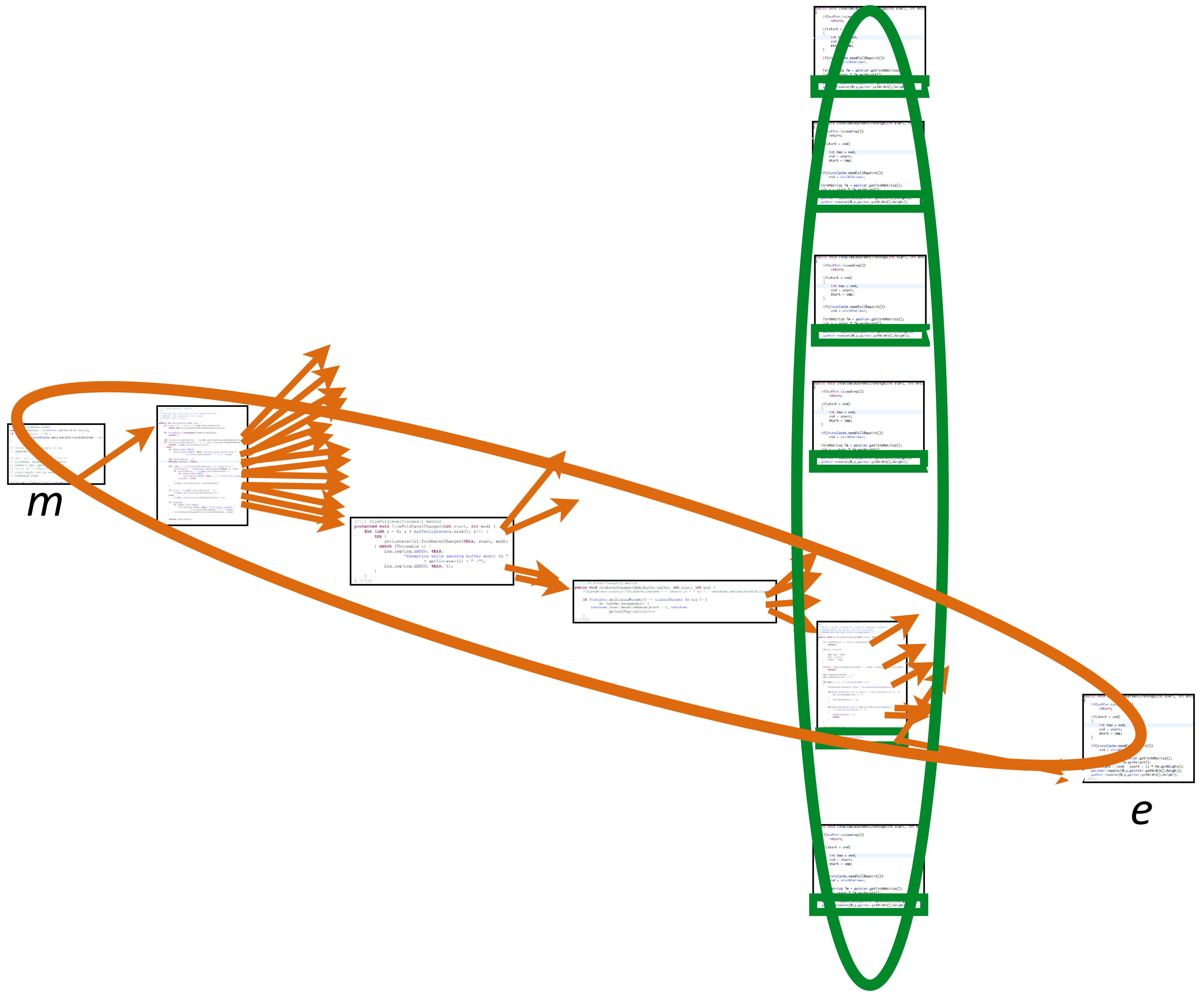


search criteria

identifier
statement type (field write/read, library call)

Reachability Question Example

A search along **feasible paths downstream** or **upstream** from a statement (*m)* for **target statements** matching **search criteria (calls to method e)**



feasible paths ∩ statements matching search criteria

Longest Activities: Control Flow

4 out of the 5 longest investigation activities

Primary question	**Time** (m)	**Related control flow question**
How is this data structure being mutated in this code?	83	Search downstream for **writes** to data structure
"Where [is] the code assuming that the tables are already there?"	53	**Compare** behaviors when tables are or are not loaded
How [does] application state change when *m* is called denoting startup completion?	50	Find field **writes** caused by m
"Is [there] another reason why *status* could be non-zero?"	11	Find statements through which values **flow** into status

5 out of the 5 longest debugging activities

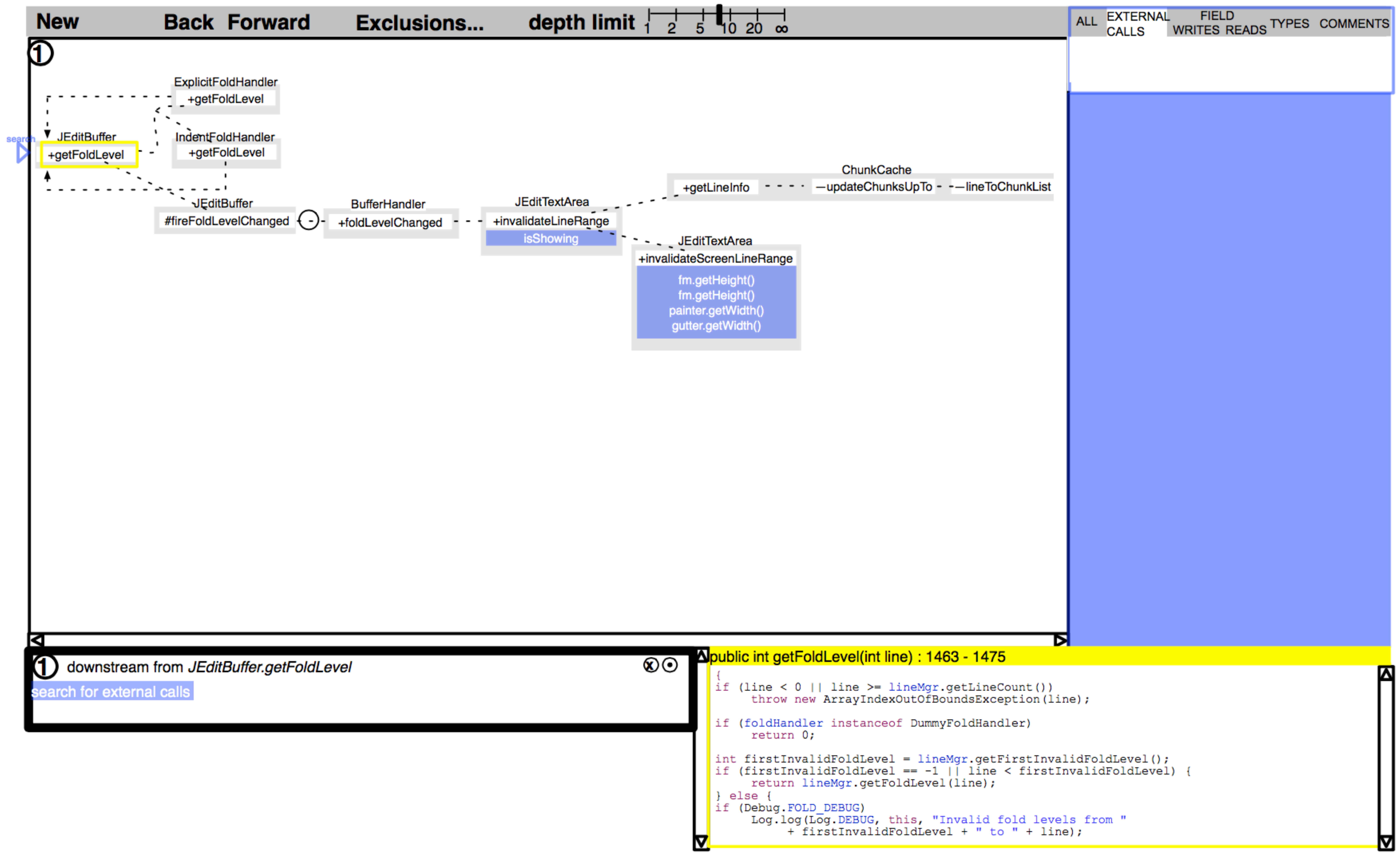
Where is method *m* generating an error?	66	Search downstream from *m* for **error** text
What resources are being acquired to cause this deadlock?	51	Search downstream for **acquire** method calls
"When they have this attribute, they must use it somewhere to generate the content, so where is it?"	35	Search downstream for **reads** of attribute
"What [is] the test doing which is different from what my app is doing?"	30	**Compare** test traces to app traces
How are these thread pools interacting?	19	Search downstream for **calls** into thread pools

Insights

- Developers can construct ***incorrect*** mental models of control flow, leading them to insert **defects**

- The ***longest*** investigation & debugging activities involved a single primary question about control flow

- Found evidence for an underlying cause of these difficulties
  Challenges answering ***reachability questions***



1 downstream from *JEditBuffer.getFoldLevel*

search for external calls

public int getFoldLevel(int line) : 1463 - 1475

```
{
if (line < 0 || line >= lineMgr.getLineCount())
     throw new ArrayIndexOutOfBoundsException(line);

if (foldHandler instanceof DummyFoldHandler)
     return 0;

int firstInvalidFoldLevel = lineMgr.getFirstInvalidFoldLevel();
if (firstInvalidFoldLevel == -1 || line < firstInvalidFoldLevel) {
     return lineMgr.getFoldLevel(line);
} else {
if (Debug.FOLD_DEBUG)
     Log.log(Log.DEBUG, this, "Invalid fold levels from "
          + firstInvalidFoldLevel + " to " + line);
```

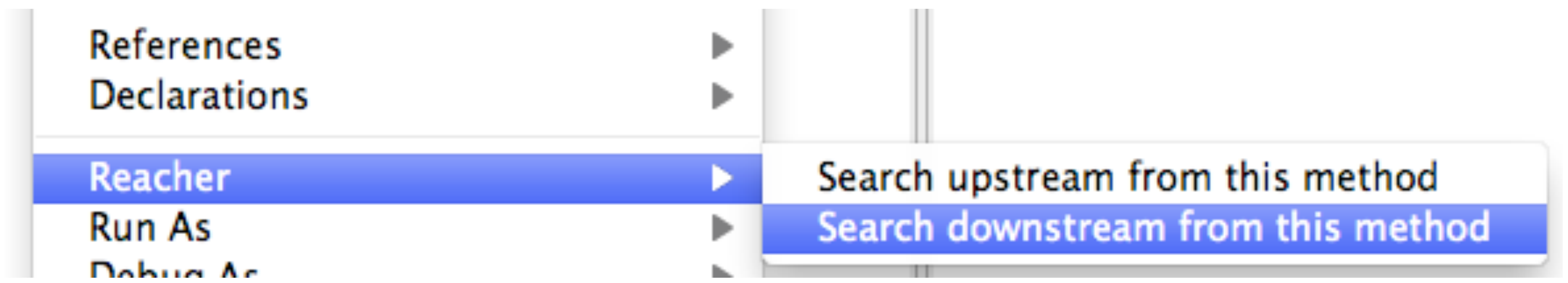
Paper Prototype Study

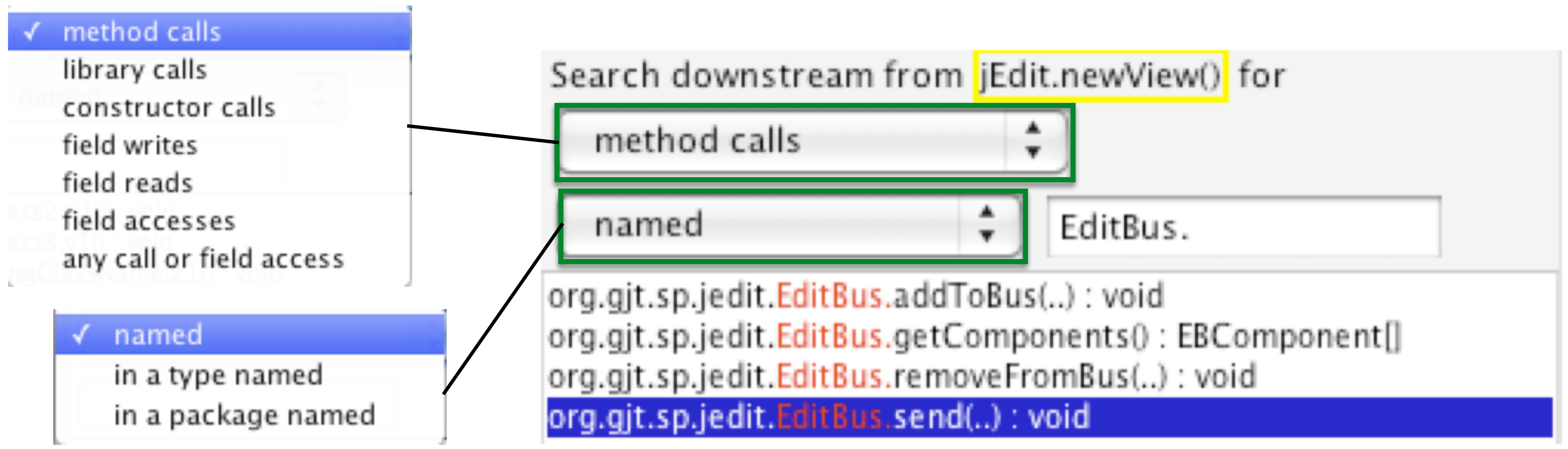
- Built mockups of interface for task from lab study
- Asked 1 participant to complete lab study task with Eclipse & mockup of *Reacher*
  - Paper overlay of *Reacher* commands on monitor
  - Experimenter opened appropriate view
- Asked to think aloud, screen capture + audio recording

Study results

- Used *Reacher* to explore code, unable to complete task
- Barriers discovered
  - Wanted to see methods before or after, not on path to origin or destination
  - Switching between downstream and upstream confusing, particularly search cursor
  - Found horizontal orientation confusing, as unlike debugger call stacks
  - Wanted to know when a path might execute

Find Statements Matching Search Criteria


References
Declarations
Reacher
Run As
Search upstream from this method
Search downstream from this method



Search downstream from jEdit.newView() for
method calls
named EditBus.

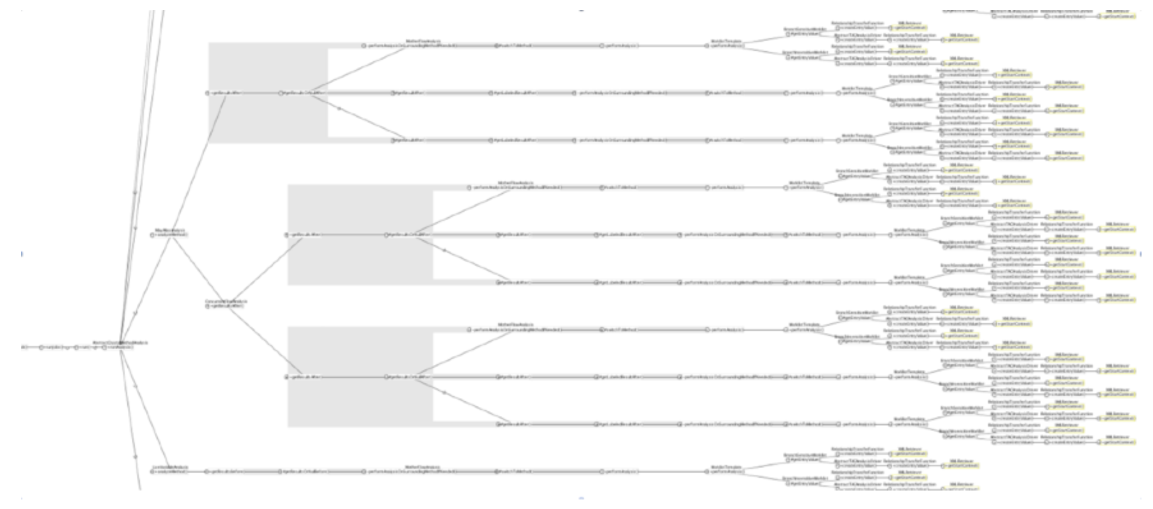
```
org.gjt.sp.jedit.EditBus.addToBus(..) : void
org.gjt.sp.jedit.EditBus.getComponents() : EBComponent[]
org.gjt.sp.jedit.EditBus.removeFromBus(..) : void
org.gjt.sp.jedit.EditBus.send(..) : void
```

Examples of observed reachability questions Reacher supports	**Steps to use Reacher**
What resources are being acquired to cause this deadlock?	Search downstream for each method which might acquire a resource, pinning results to keep them visible
When they have this attribute, they must use it somewhere to generate the content, so where is it?	Search downstream for a field read of the attribute
How are these thread pools interacting?	Search downstream for the thread pool class
How is data structure *struct* being mutated in this code (between *o* and *d*)?	Search downstream for *struct* class, scoping search to matching type names and searching for field writes.
How [does] application state change when *m* is called denoting startup completion?	Search downstream from *m* for all field writes

Help Developers Understand Paths

Goal: help developers reason about control flow by summarizing statements along paths in **compact** visualization

Challenges:
control flow paths can be



complex

long

repetitive

developers get lost and disoriented navigating code

Approach:

visually encode properties of path

hide paths by default

coalesce similar paths

use visualization to support navigation

Example

EditBus
+send(..)
View
+handleMessage(..)
-handleEditPaneUpdate(..)
StatusBar
+updateCaretStatus()
JEditTextArea
+setBuffer(..)
+selectNone()
+setSelection(..)
-finishCaretUpdate(..,..,..)
_finishCaretUpdate()
+scrollToCaret(..)
+scrollTo(..,..
CaretHandle
-fireCaretEvent()
+caretUpdate
EditPane
+setBuffer(..)
JEditTextArea
+setCaretPosition(..)
+moveCaretPosition(..,..)
+moveCaretPositi
VFSManager
+runInAWTThread(..)
WorkThreadPool
+addWorkRequest(..,..
+run()
EditPane
+loadCaretInfo()
JEditTextArea
+setSelection(..)
JEditTextArea
+setFirstPhysicalLine(..)

Evaluation

Does Reacher enable developers to answer reachability questions faster or more successfully?

Method

12 developers 15 minutes to answer **reachability** question x **6**

Eclipse only on 3 tasks Eclipse w/ Reacher on 3 tasks

Tasks

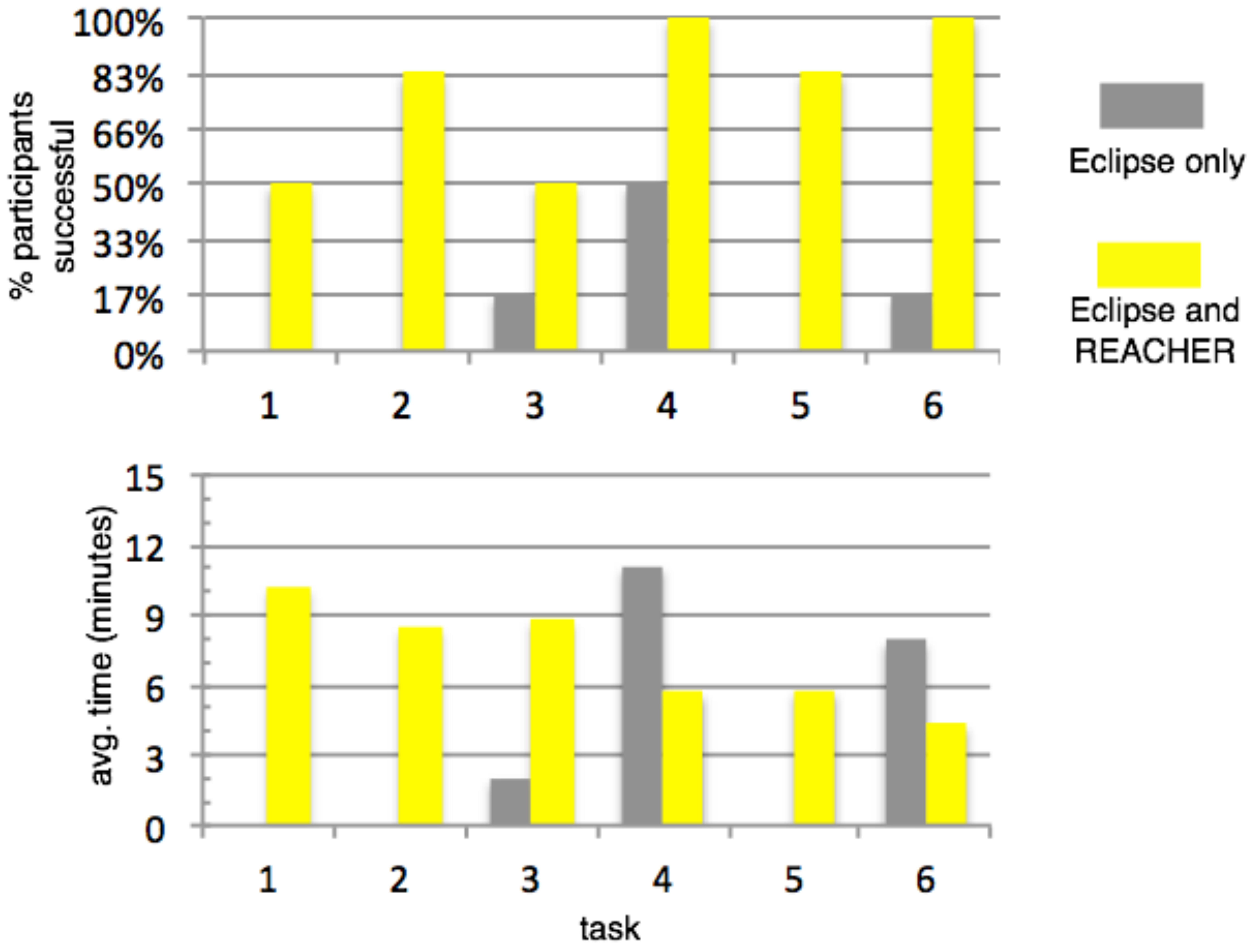
Based on developer questions in lab study.

Example:

When a new view is created in jEdit.newView(View), what messages, in what order, may be sent on the EditBus (EditBus.send())?

Results

Developers with REACHER were **5.6** times more **successful** than those working with Eclipse only.



Task time includes only participants that succeeded.

More Results

Participants with **Reacher** used it to jump between methods.

"It seems pretty cool if you can navigate your way around a complex graph."

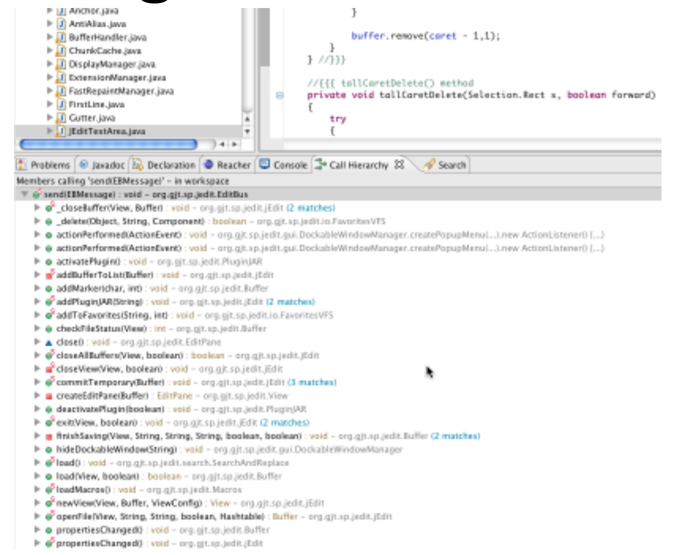
When **not** using Reacher, participants often reported being lost and confused.

"Where am I? I'm so lost."

"These call stacks are horrible."

"There was a call to it here somewhere, but I don't remember the path."

"I'm just too lost."



Participants reported that they liked working with Reacher.

"I like it a lot. It seems like an easy way to navigate the code. And the view maps to more of how I think of the call hierarchy."

"Reacher was my hero. … It's a lot more fun to use and look at."

"You don't have to think as much."

Reflection on Design Process

- Started with a goal: make debugging in large, complex codebases better
- Observed users to build ***insight*** into what key challenge was
- Rather than address usability challenges of existing debugging tools, designed new way to debug
- Gathered evidence that it worked better

10 Minute Break

In-Class Activity

- Form groups of 2 or 3
- A venture capitalist just gave your group $5M to build a new consumer software product (mobile, web, desktop, etc.)
- Brainstorm an idea: what's the product? how will it help?
- Deliverables. Answer the following questions:
  - What do you know now
    - Who are the users? What are their tasks and goals? What problems do they encounter? How will your tool help?
  - What would like to learn through needfinding that you don't already know?
  - How would you use an interview / survey / observations / or other method to answer these questions?