SWE 632 - Design & Development of User Interfaces

George Mason University

Instructor:
Dr. Kevin Moran

Teaching Assistant:
David Gonzalez Samudio

Class will start in:

20:00

SWE 632 - Design & Development of User Interfaces

George Mason University

Instructor:
Dr. Kevin Moran

Teaching Assistant:
David Gonzalez Samudio

Class will start in:

20:00

SWE 632 - Design & Development of User Interfaces

Fall 2020

George Mason University

Dr. Kevin Moran

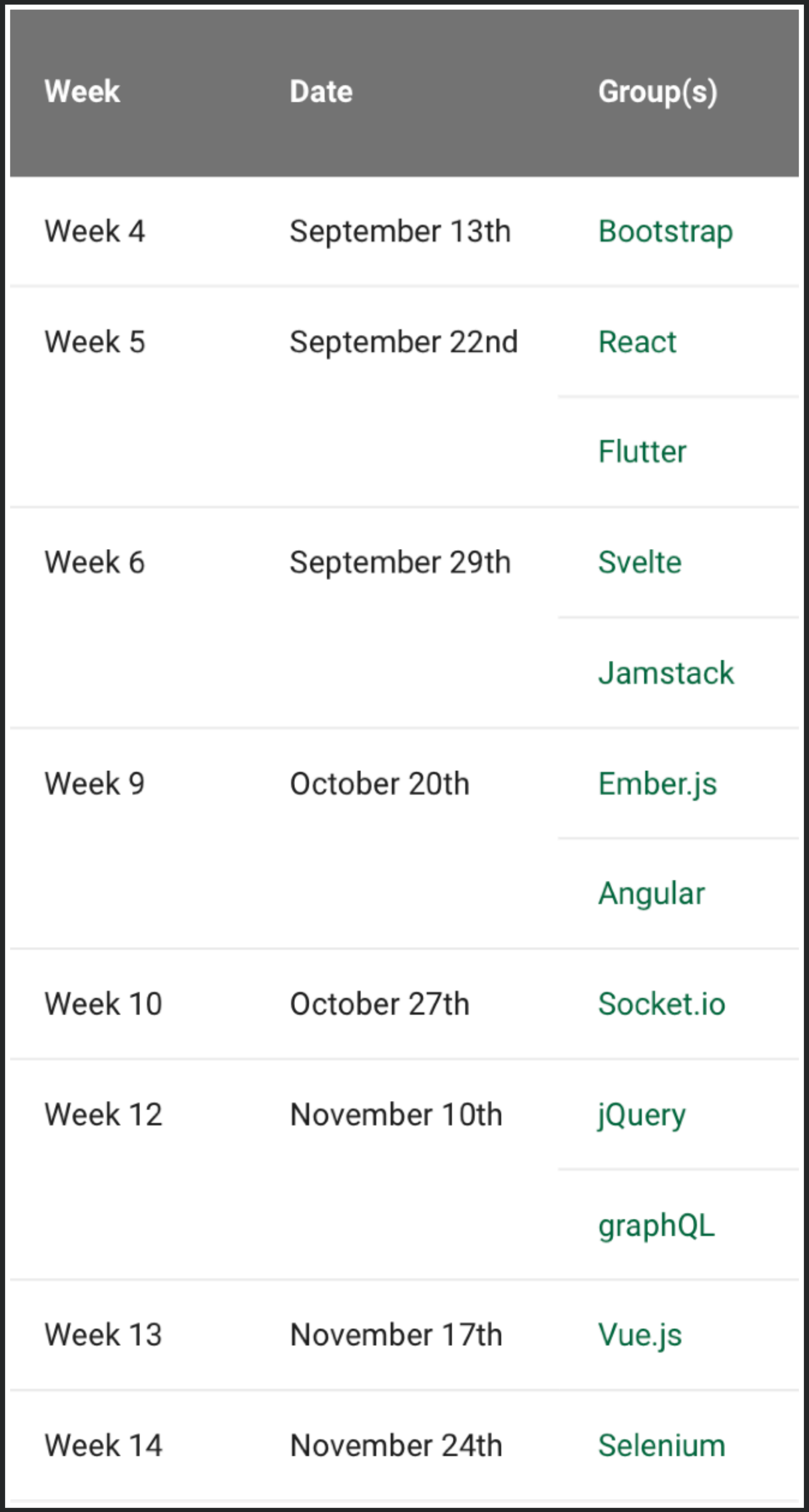
Week 3:

User-Centered Design

Administrivia

- ***Tech Talks:*** Schedule has been posted to the course website!
- ***Project Checkpoint 1:*** Feedback sent out, writeup due before next class.
- ***Discussion Question 3:*** Posted After Class

Tech Talks

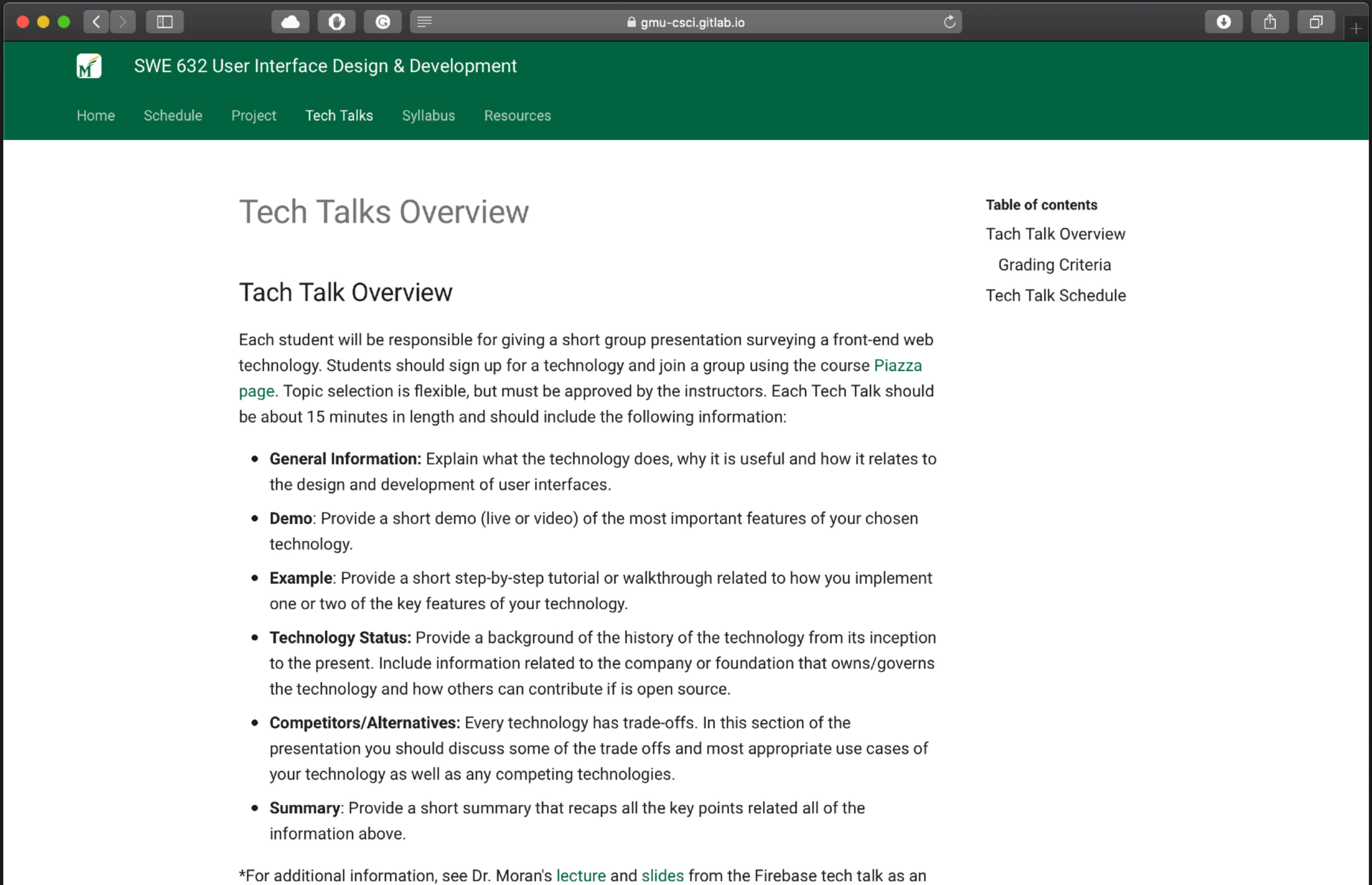


Week	Date	Group(s)
Week 4	September 13th	Bootstrap
Week 5	September 22nd	React Flutter
Week 6	September 29th	Svelte Jamstack
Week 9	October 20th	Ember.js Angular
Week 10	October 27th	Socket.io
Week 12	November 10th	jQuery graphQL
Week 13	November 17th	Vue.js
Week 14	November 24th	Selenium

Tech Talks

1. General Information
2. Demo
3. Example
4. Technology Status
5. Competitors/Alternatives
6. Summary

Tech Talks



SWE 632 User Interface Design & Development

Tech Talks Overview

Tach Talk Overview

Each student will be responsible for giving a short group presentation surveying a front-end web technology. Students should sign up for a technology and join a group using the course Piazza page. Topic selection is flexible, but must be approved by the instructors. Each Tech Talk should be about 15 minutes in length and should include the following information:

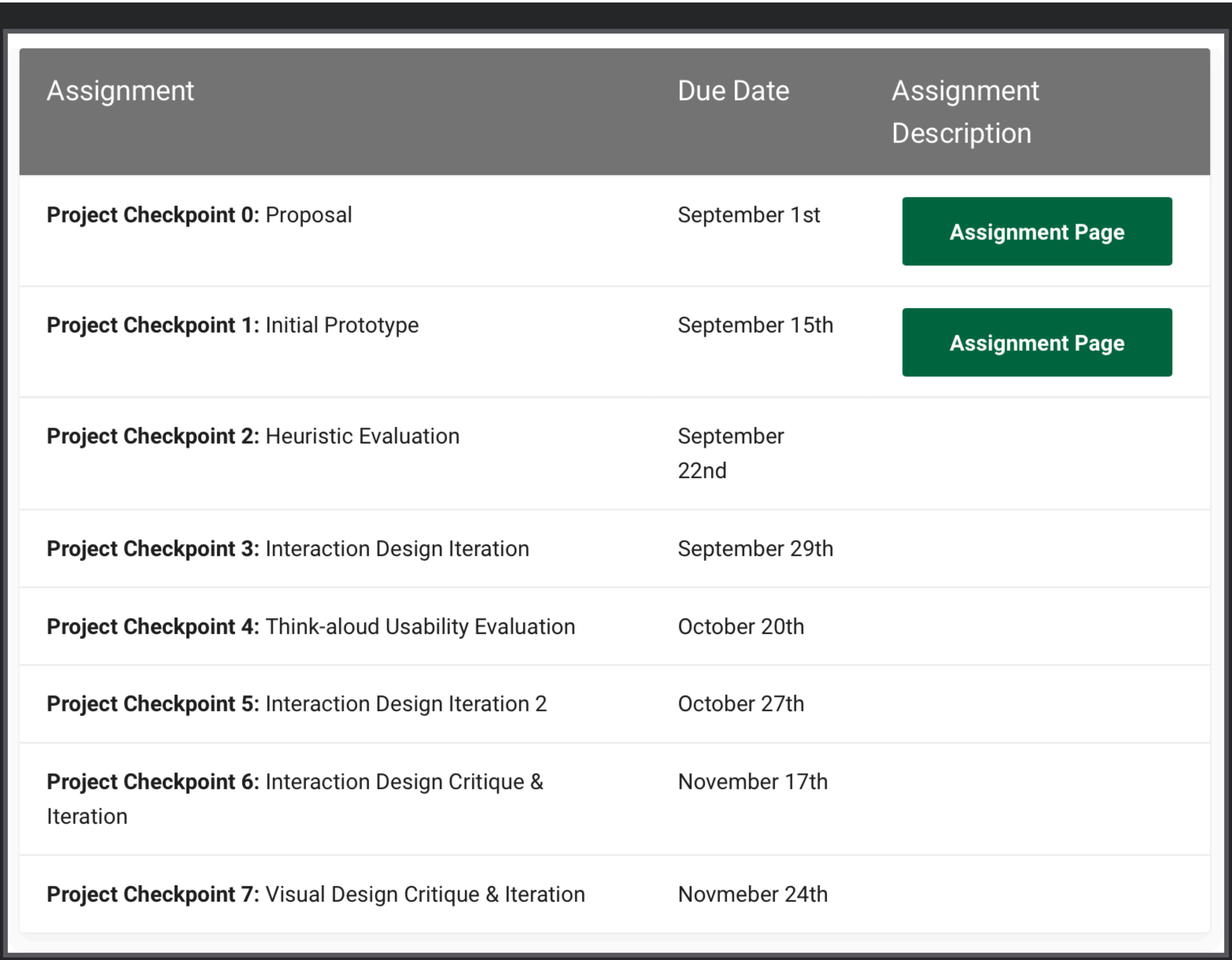
- **General Information:** Explain what the technology does, why it is useful and how it relates to the design and development of user interfaces.
- **Demo**: Provide a short demo (live or video) of the most important features of your chosen technology.
- **Example**: Provide a short step-by-step tutorial or walkthrough related to how you implement one or two of the key features of your technology.
- **Technology Status:** Provide a background of the history of the technology from its inception to the present. Include information related to the company or foundation that owns/governs the technology and how others can contribute if is open source.
- **Competitors/Alternatives:** Every technology has trade-offs. In this section of the presentation you should discuss some of the trade offs and most appropriate use cases of your technology as well as any competing technologies.
- **Summary**: Provide a short summary that recaps all the key points related all of the information above.

*For additional information, see Dr. Moran's lecture and slides from the Firebase tech talk as an

Project Checkpoint I

- Implement as much functionality as you can by this first checkpoint.
- The remainder of the project checkpoints will involve two activities:
  - *Peer Design Evaluations*
  - *Design Iterations*

Project Checkpoints



Assignment	Due Date	Assignment Description
Project Checkpoint 0: Proposal	September 1st	Assignment Page
Project Checkpoint 1: Initial Prototype	September 15th	Assignment Page
Project Checkpoint 2: Heuristic Evaluation	September 22nd	
Project Checkpoint 3: Interaction Design Iteration	September 29th	
Project Checkpoint 4: Think-aloud Usability Evaluation	October 20th	
Project Checkpoint 5: Interaction Design Iteration 2	October 27th	
Project Checkpoint 6: Interaction Design Critique & Iteration	November 17th	
Project Checkpoint 7: Visual Design Critique & Iteration	Novmeber 24th	

Class Overview

Class Overview

- *Part 1 - User-Centered Design:* How do we design for the user?
- *Part 2 - Some User-Centered Design Considerations:* Take Note
- ***Part 3 - Example:*** User Centered Design in Research

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

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

Technology-Centered Design

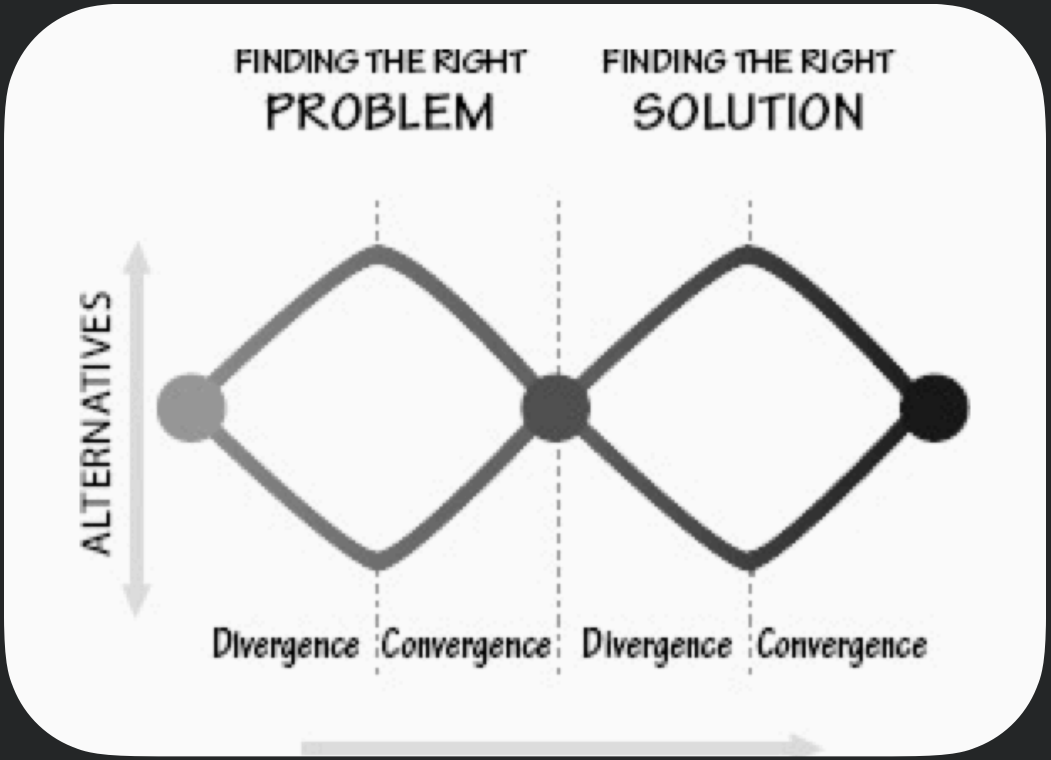
What can this technology do?

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

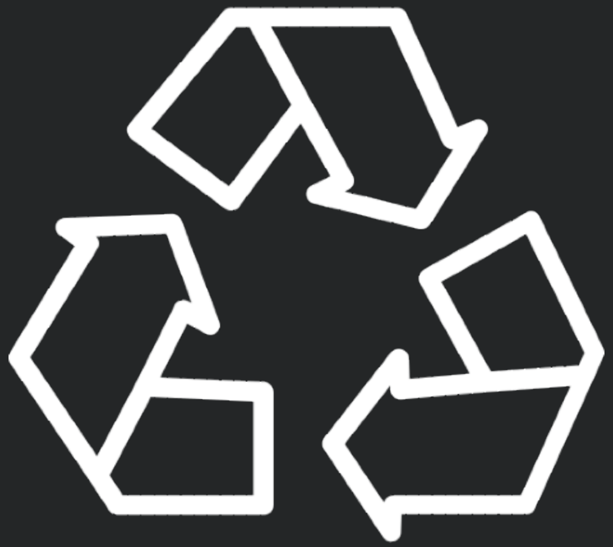


Iterative Model of Design

Observation

(Re)Define the Problem

Understand User Needs



Test

Evaluate what you have built

Idea Generation

Brainstorm what to build

Prototype

Build

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

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

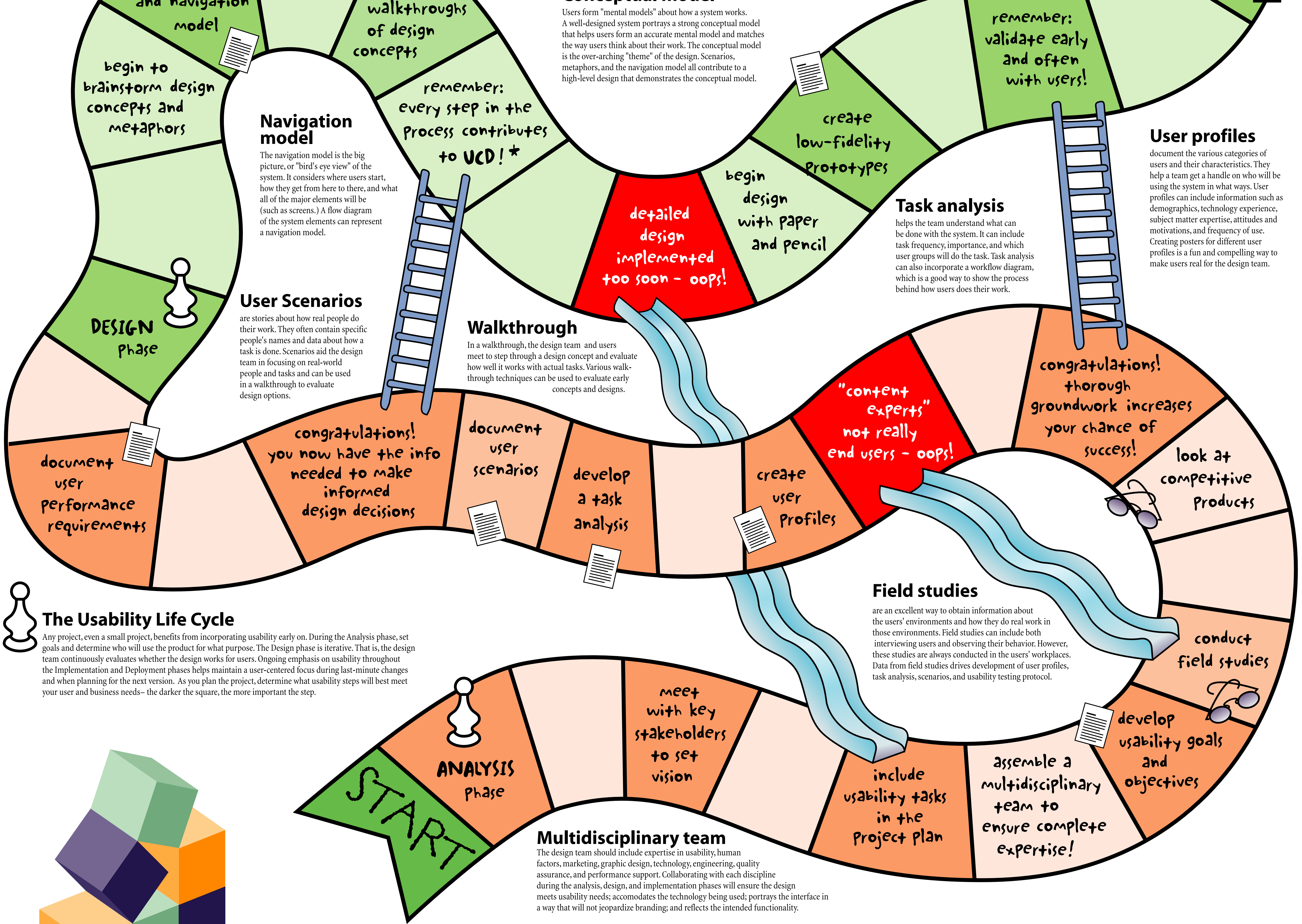
Testing - User Centered Evaluation

- 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!

Conceptual model

Users form "mental models" about how a system works. A well-designed system portrays a strong conceptual model that helps users form an accurate mental model and matches the way users think about their work. The conceptual model is the over-arching "theme" of the design. Scenarios, metaphors, and the navigation model all contribute to a high-level design that demonstrates the conceptual model.

Navigation model

The navigation model is the big picture, or "bird's eye view" of the system. It considers where users start, how they get from here to there, and what all of the major elements will be (such as screens.) A flow diagram of the system elements can represent a navigation model.

User Scenarios

are stories about how real people do their work. They often contain specific people's names and data about how a task is done. Scenarios aid the design team in focusing on real-world people and tasks and can be used in a walkthrough to evaluate design options.

Walkthrough

In a walkthrough, the design team and users meet to step through a design concept and evaluate how well it works with actual tasks. Various walkthrough techniques can be used to evaluate early concepts and designs.

Task analysis

helps the team understand what can be done with the system. It can include task frequency, importance, and which user groups will do the task. Task analysis can also incorporate a workflow diagram, which is a good way to show the process behind how users does their work.

User profiles

document the various categories of users and their characteristics. They help a team get a handle on who will be using the system in what ways. User profiles can include information such as demographics, technology experience, subject matter expertise, attitudes and motivations, and frequency of use. Creating posters for different user profiles is a fun and compelling way to make users real for the design team.

Field studies

are an excellent way to obtain information about the users' environments and how they do real work in those environments. Field studies can include both interviewing users and observing their behavior. However, these studies are always conducted in the users' workplaces. Data from field studies drives development of user profiles, task analysis, scenarios, and usability testing protocol.

The Usability Life Cycle

Any project, even a small project, benefits from incorporating usability early on. During the Analysis phase, set goals and determine who will use the product for what purpose. The Design phase is iterative. That is, the design team continuously evaluates whether the design works for users. Ongoing emphasis on usability throughout the Implementation and Deployment phases helps maintain a user-centered focus during last-minute changes and when planning for the next version. As you plan the project, determine what usability steps will best meet your user and business needs– the darker the square, the more important the step.

Multidisciplinary team

The design team should include expertise in usability, human factors, marketing, graphic design, technology, engineering, quality assurance, and performance support. Collaborating with each discipline during the analysis, design, and implementation phases will ensure the design meets usability needs; accomodates the technology being used; portrays the interface in a way that will not jeopardize branding; and reflects the intended functionality.

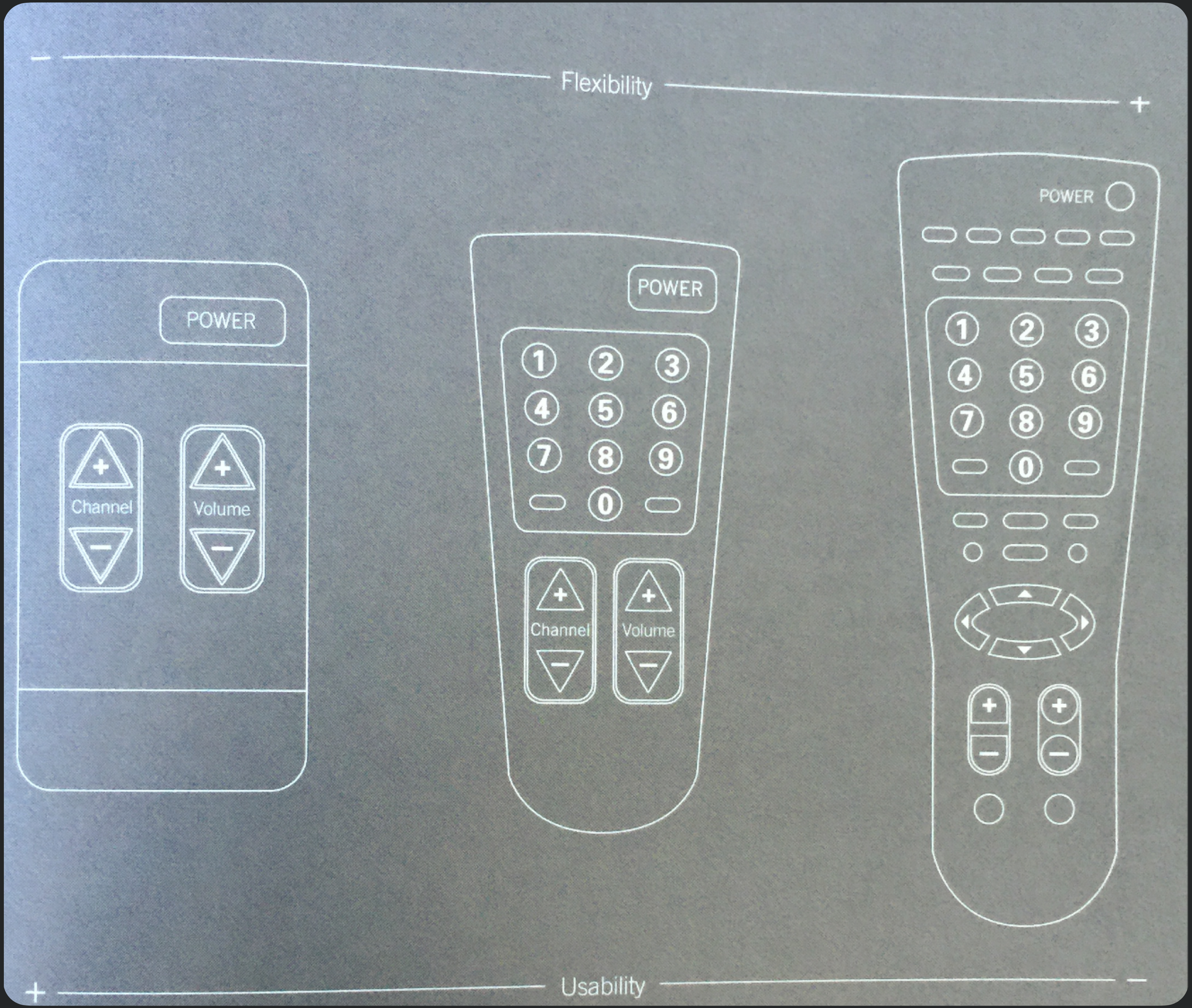
upa

usability
professionals'
association
www.upassoc.org

Acknowledgements:

Meg Ross - digitalMeg
Julie Nowicki - Optavia Corporation
Dara Solomon & **Larry Yarbrough** - iXL, Inc.
Charlotte Schwendeman - Consultant

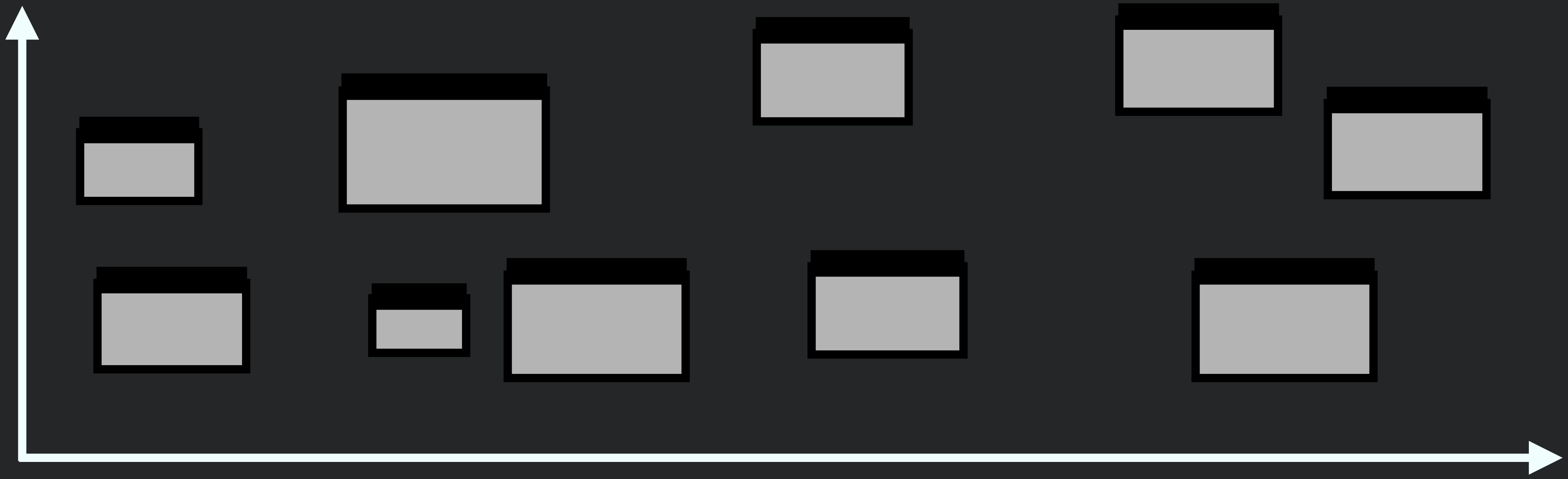
Flexibility-usability tradeoff



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

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)

Activities and Tasks

- ***Activity*** - set of tasks performed together for a common goal
  - Go shopping
- ***Task*** - component of an activity, organized cohesive set of operations towards a single low-level goal
  - Drive to market
  - Find shopping basket
  - Find item in store
  - Pay for items

Activities and Tasks

- Activities are ***hierarchical***
- High-level activities spawn other activities, spawn tasks
- Software supports tasks and activities
- Important to design for ***activities***, not just tasks
    - Support whole activity seamlessly
    - Ensure interactions between tasks do not interfere

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?

7 Minute Break

SWE 632 - Design & Development of User Interfaces

George Mason
University

Instructor:
Dr. Kevin Moran

Teaching Assistant:
David Gonzalez Samudio

Class will start in:

07:00

In the Chat, list what ***user-centered design*** means to you.

SWE 632 - Design & Development of User Interfaces

George Mason University

Instructor:
Dr. Kevin Moran

Teaching Assistant:
David Gonzalez Samudio

Class will start in:

07:00

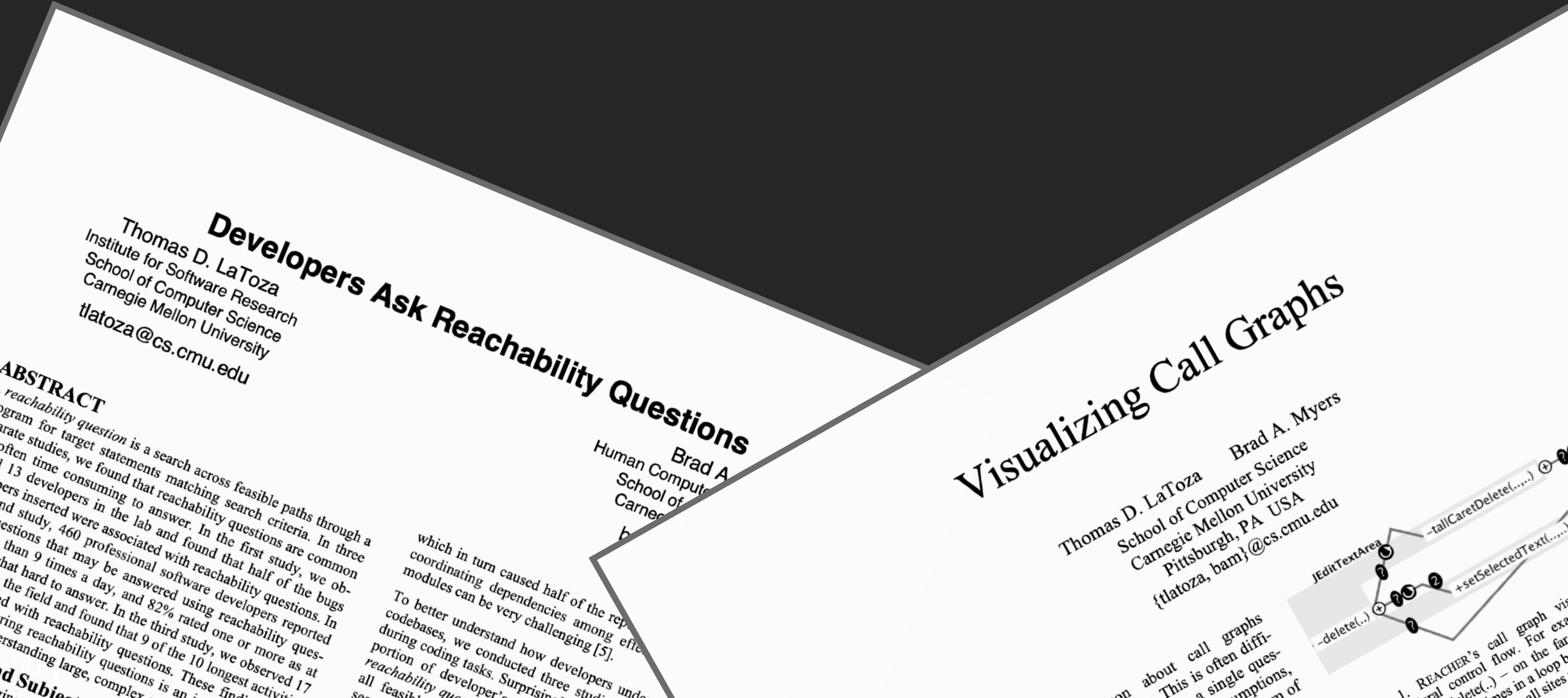
In the Chat, list what ***user-centered design*** means to you.

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 Comput
School of
Carneg
b

ABSTRACT

reachability question is a search across feasible paths through a
ogram for target statements matching search criteria. In three
arate studies, we found that reachability questions are common
often time consuming to answer. In the first study, we ob-
l 13 developers in the lab and found that half of the bugs
pers inserted were associated with reachability questions. In
nd study, 460 professional software developers reported
estions that may be answered using reachability ques-
than 9 times a day, and 82% rated one or more as at
hat hard to answer. In the third study, we observed 17
the field and found that 9 of the 10 longest activiti
d with reachability questions. These findi
ring reachability questions is an i
erstanding large, complex

d Subjec

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 qu
all feasib

Visualizing Call Graphs

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

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

1. REACHER's call graph vi
ral control flow. For exa
lete(..) – on the far
mes in a loop b
all sites

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.

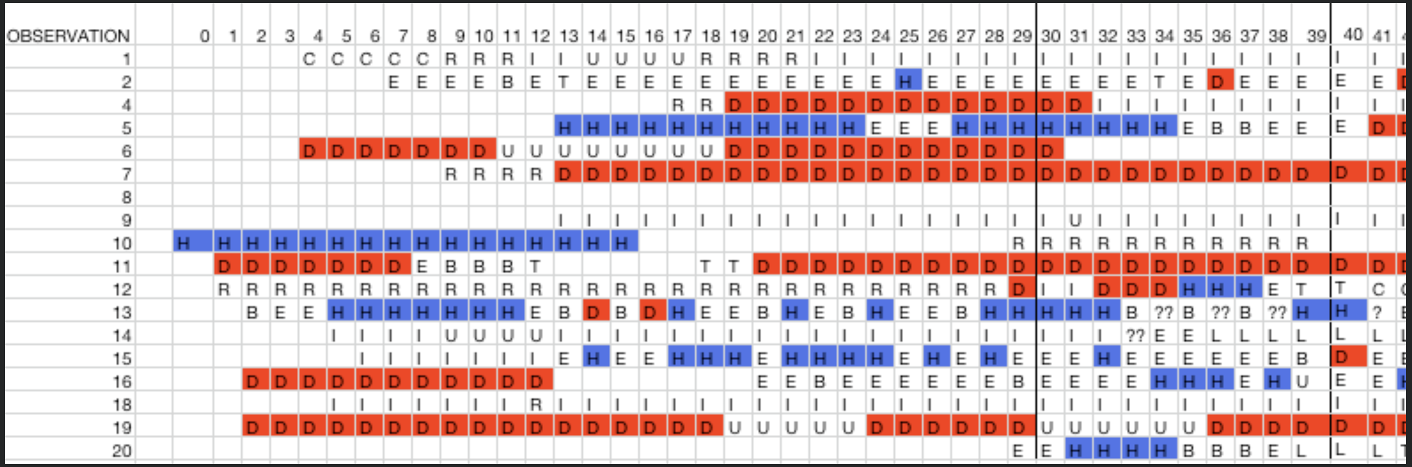
So the change seemed to have changed some of the way these things are registered,

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)

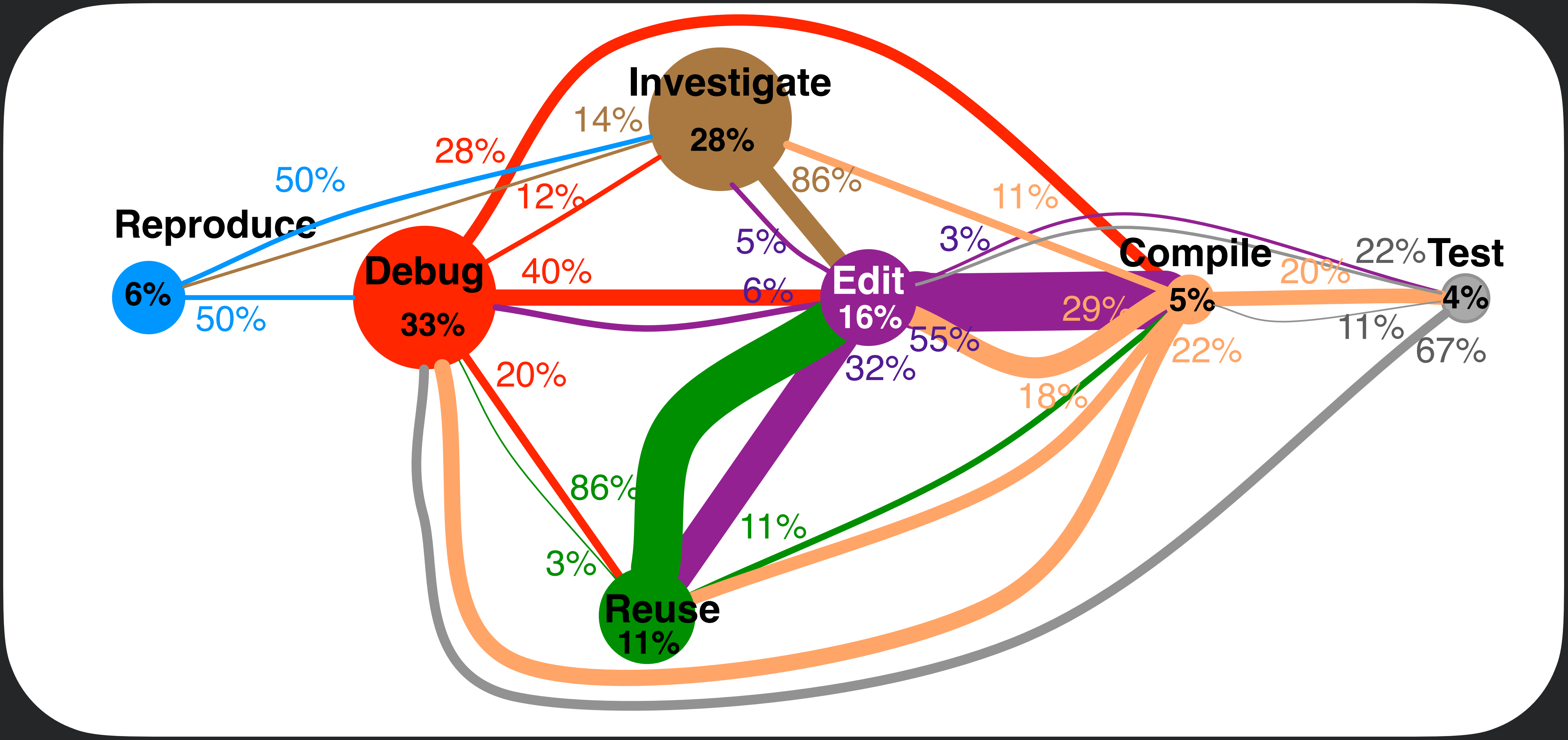
(386 pages)

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
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
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
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
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
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
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
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
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
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	?
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
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
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
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
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
20																															E	E	H	H	H	H	B	B	B	E	L	L	L

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

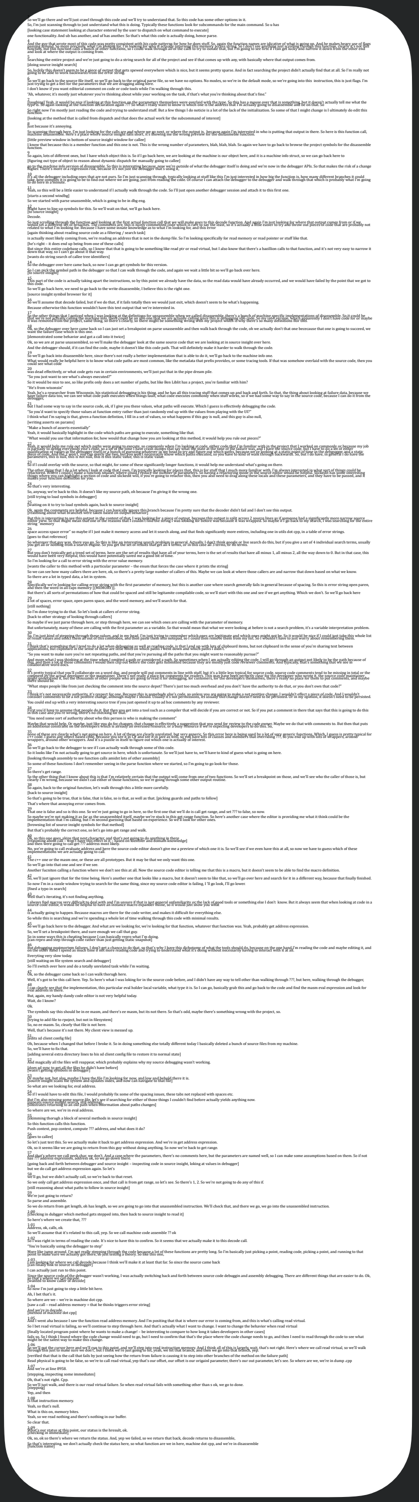
Static Call Traversal

Statically ***traversed*** calls to explore

Debugger

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

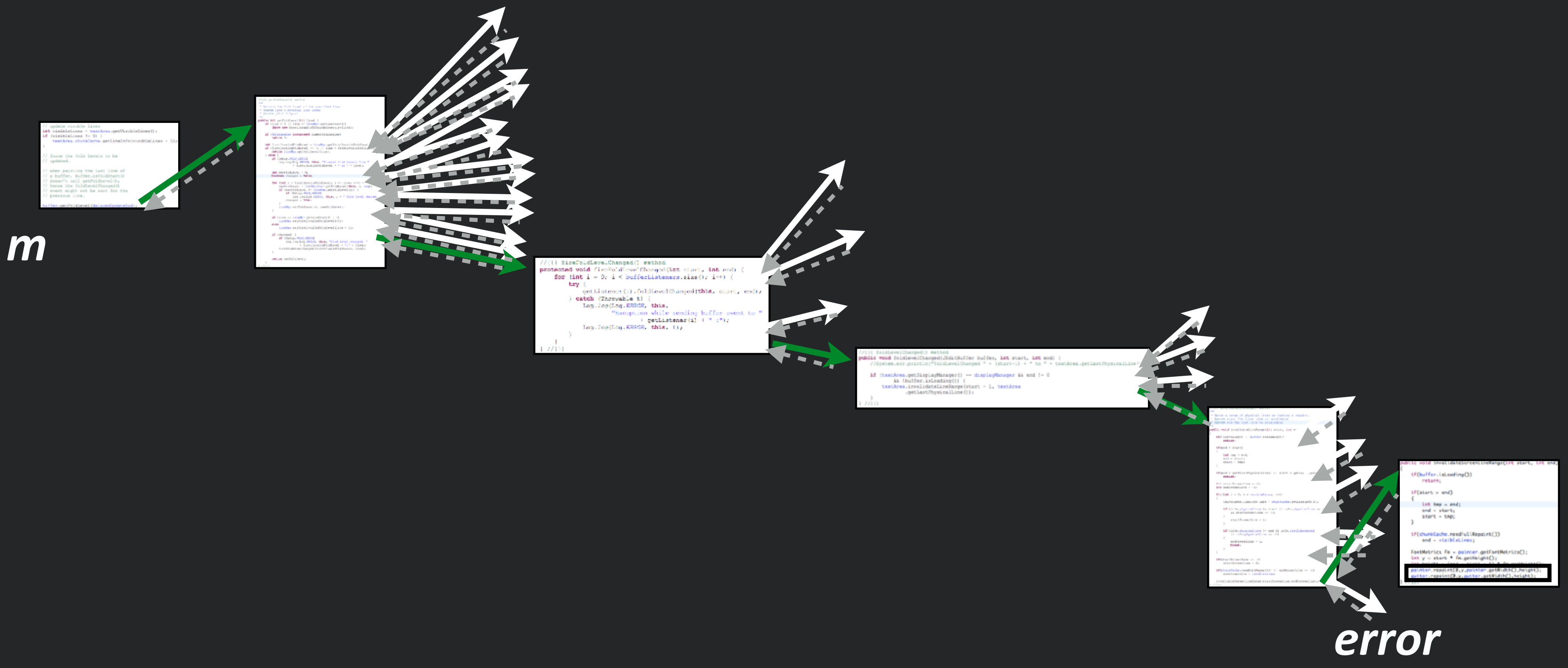
(66 minutes)



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?



```
//{{{ 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);
        }
    }
} //}}}
```

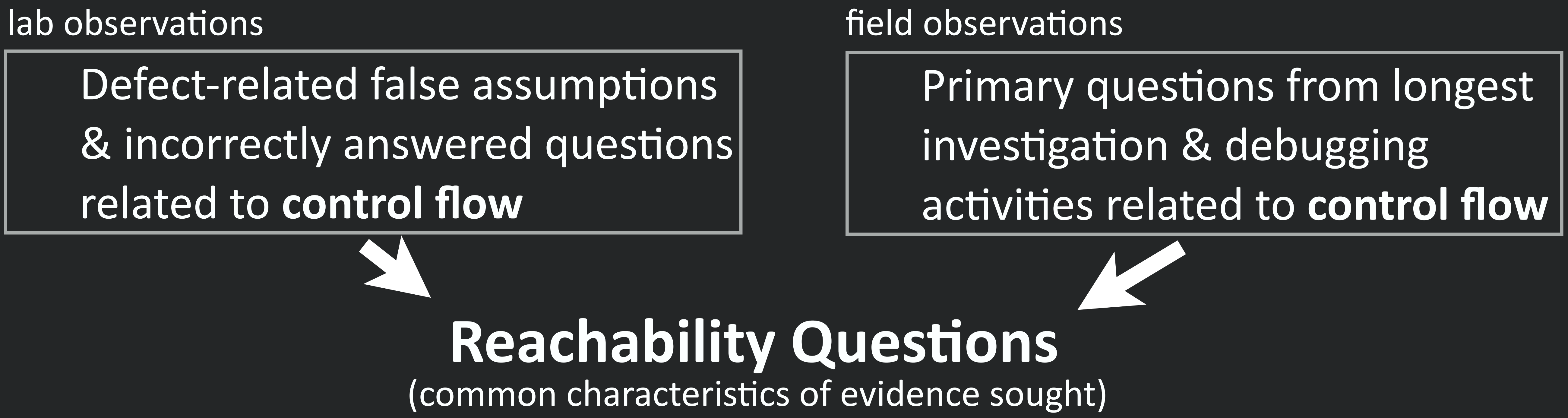
```
//{{{ foldLevelChanged() method
public void foldLevelChanged(JEditBuffer buffer, int start, int end) {
    //System.err.println("foldLevelChanged " + (start-1) + " to " + textArea.getLastPhysicalLine()

    if (textArea.getDisplayManager() == displayManager && end != 0
        && !buffer.isLoading()) {
        textArea.invalidateLineRange(start - 1, textArea
            .getLastPhysicalLine());
    }
} //}}}
```

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**

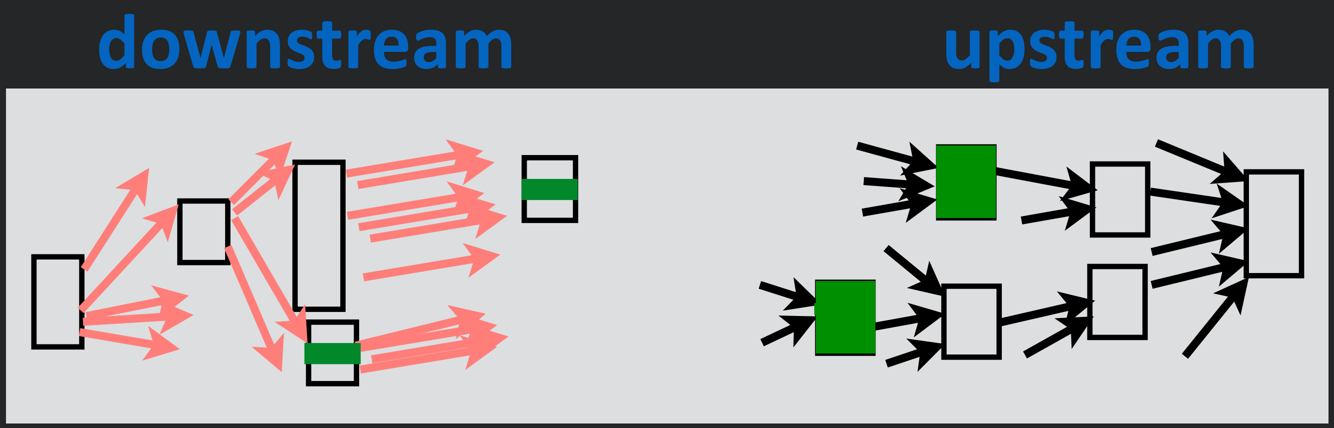
Reachability Questions

(common characteristics of evidence sought)

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

feasible paths

filter compare



search criteria

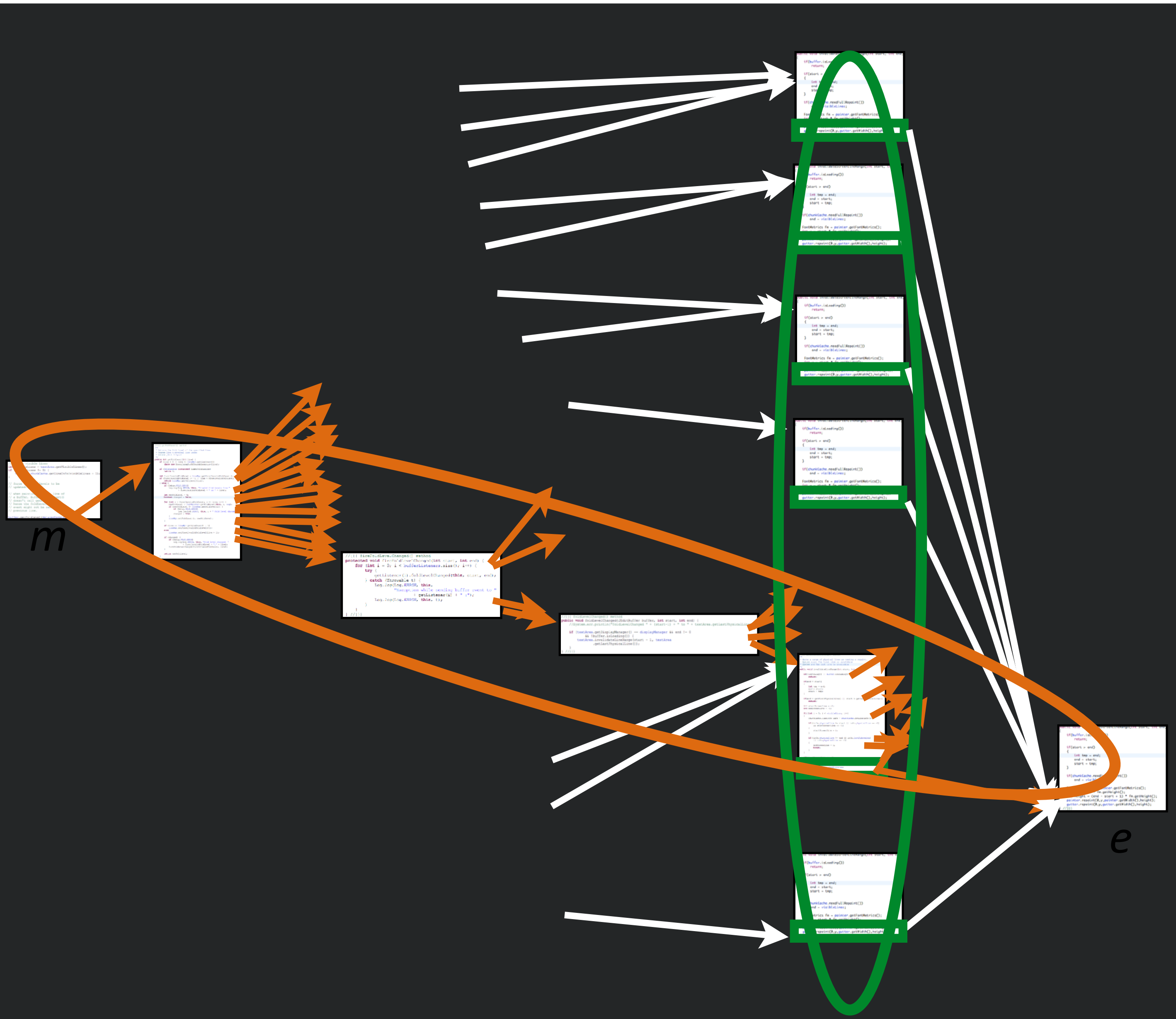
identifier

statement type (field write/read, library call)

feasible paths ∩ statements matching search criteria

Reachability Question Example

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



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

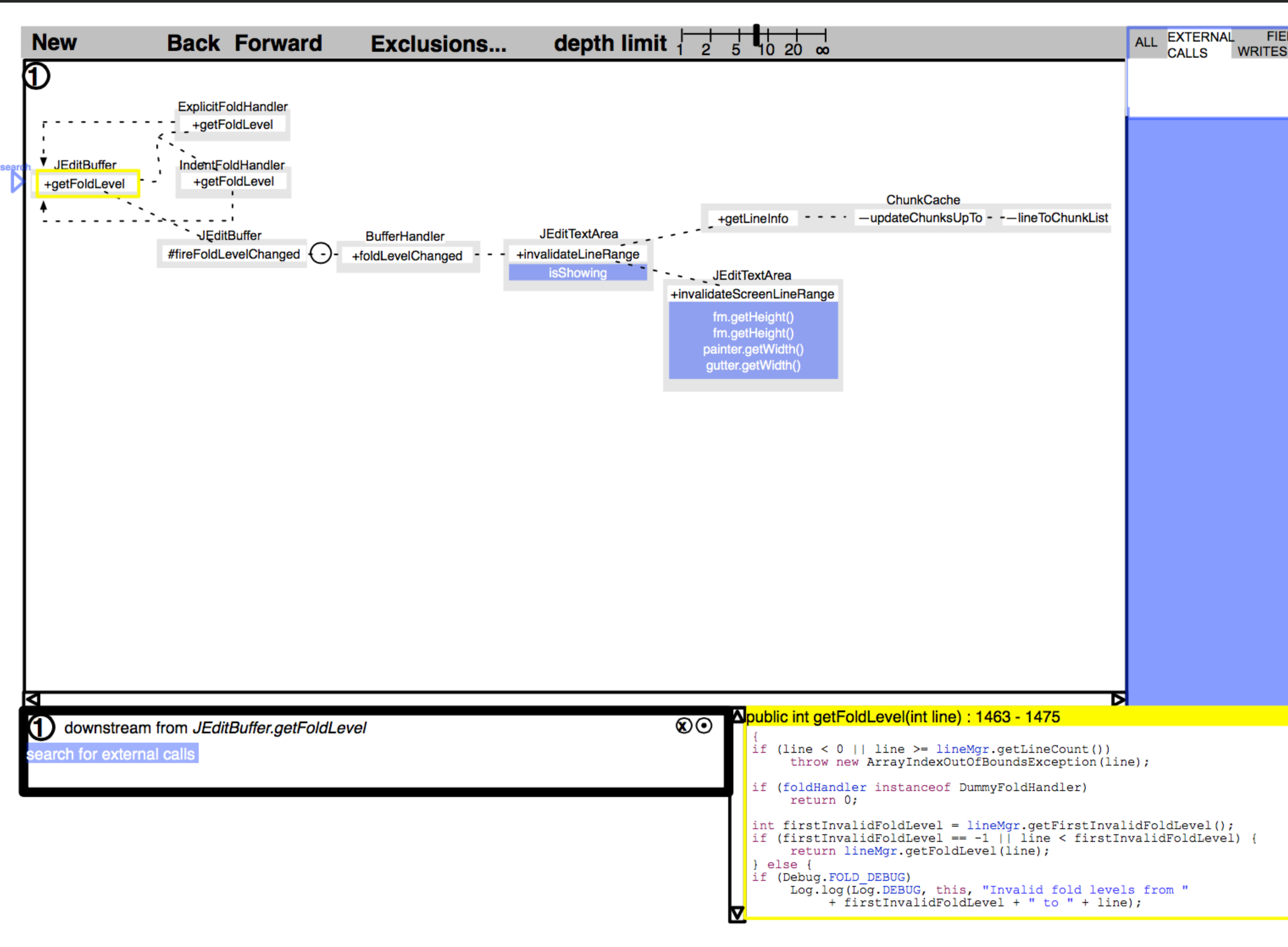
Insights

- 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*

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);
```

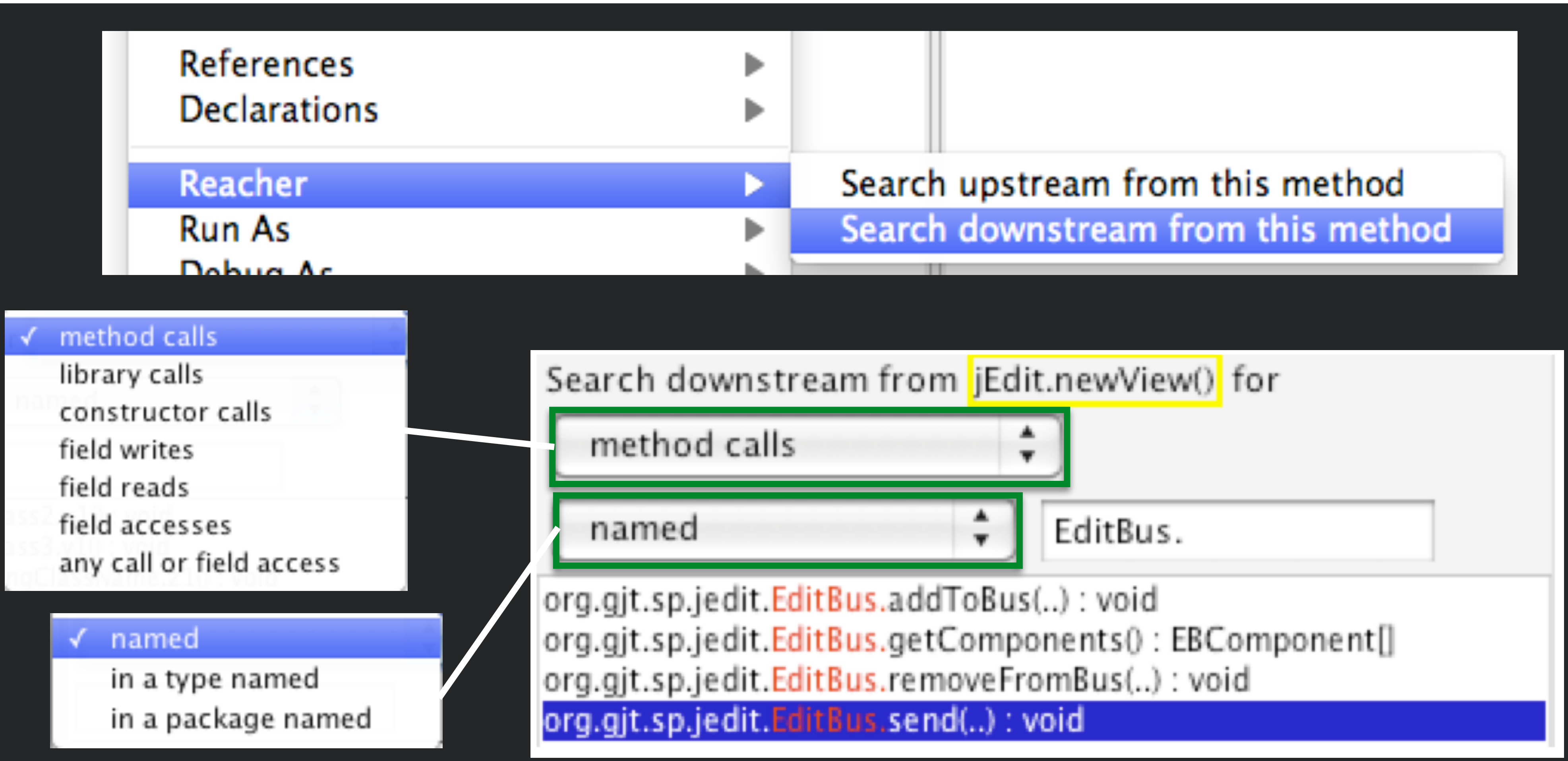
Paper Prototype Study

- 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



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
```

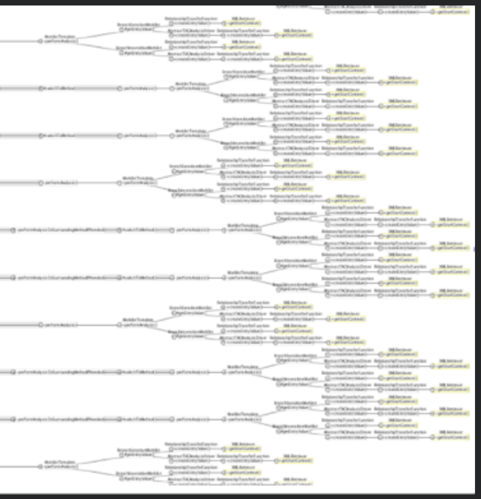
Examples of observed reachability questions Reacher supports	Steps to use Reacher
What resources are being acquired to cause this deadlock?	Search downstream for each method which might acquire a resource, pinning results to keep them visible
When they have this attribute, they must use it somewhere 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:

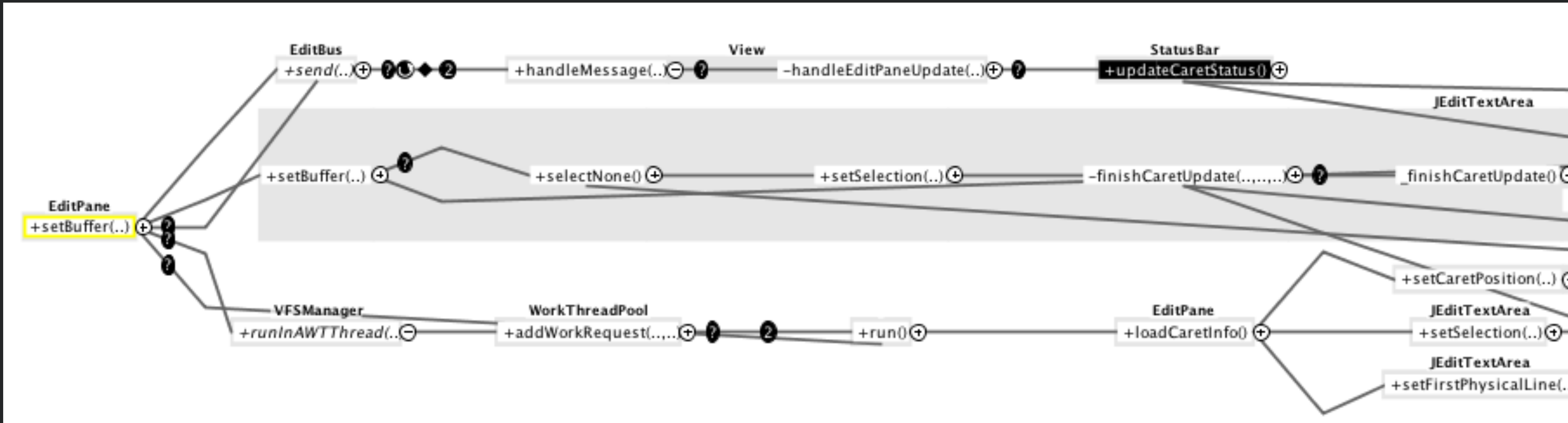
visually encode properties of path

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()
EditPane
+setBuffer(..)
+setCaretPosition(..)
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

(order counterbalanced)

Tasks

Based on developer questions in lab study.

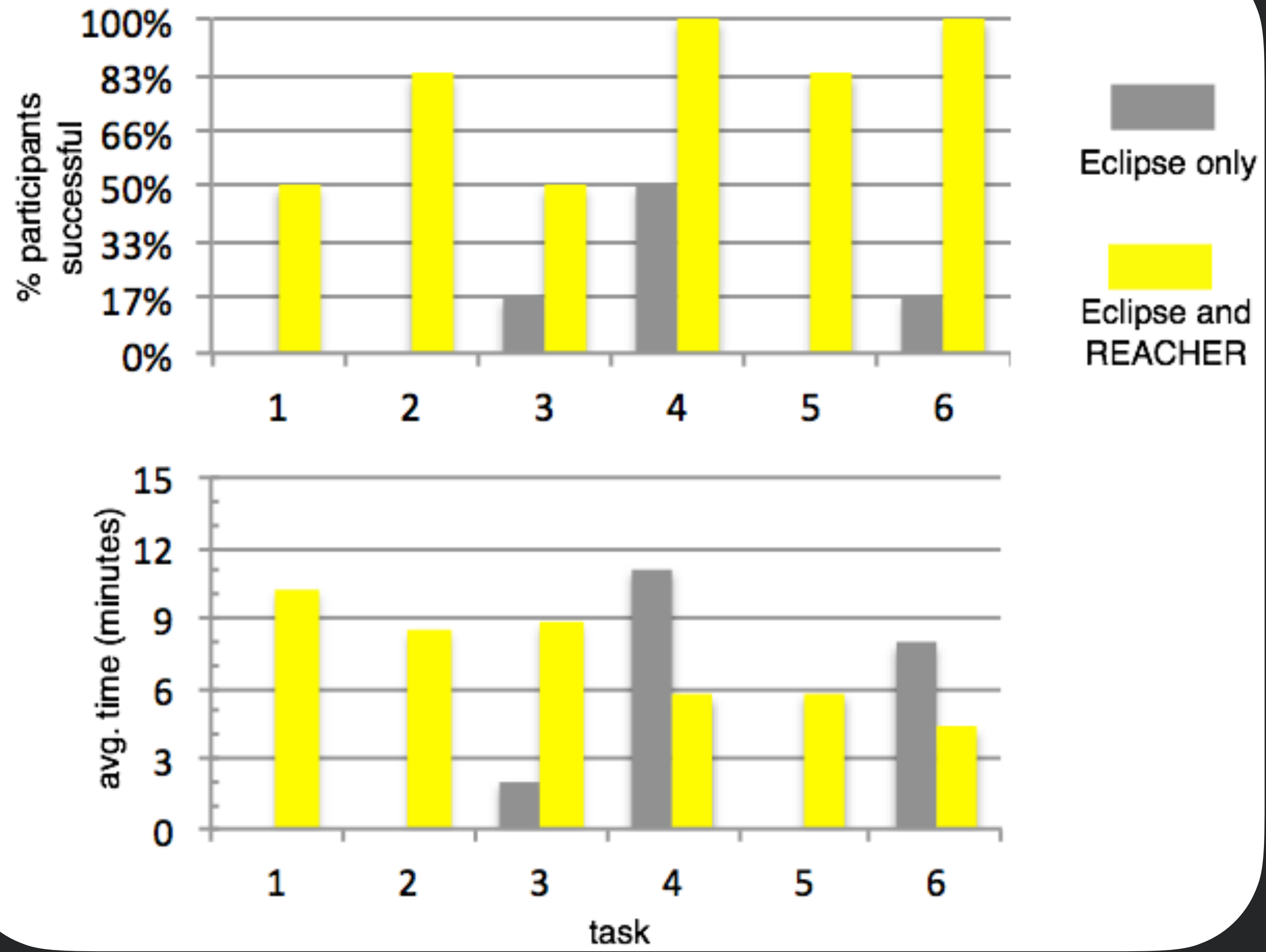
Example:

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.

(not enough successful to compare time)



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

> *"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

Acknowledgements

- Slides adapted from Dr. Thomas Latoza's SWE 632 course