Looking ahead

• Examined human cognition
• Have 2 ways to identify usability issues
• 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?
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 with 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
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
Iterative model of design

(re) Define the problem

Needfinding
understand the users

Brainstorm
ideate

Prototype
build

Test
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
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 a 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?
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?
Observations of developers in the field

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)
Coding activities working with unfamiliar code

Circle size: % of time

Edge thickness: % of transitions observed
### Longest activities related to control flow questions

**4 out of the 5 longest investigation activities**

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

**5 out of the 5 longest debugging activities**

<table>
<thead>
<tr>
<th>Primary question</th>
<th>Time (mins)</th>
<th>Related control flow question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where is method ( m ) generating an error?</td>
<td>66</td>
<td>Search downstream from ( m ) for <strong>error</strong> text</td>
</tr>
<tr>
<td>What resources are being acquired to cause this deadlock?</td>
<td>51</td>
<td>Search downstream for <strong>acquire</strong> method calls</td>
</tr>
<tr>
<td>“When they have this attribute, they must use it somewhere to generate the content, so where is it?”</td>
<td>35</td>
<td>Search downstream for <strong>reads</strong> of attribute</td>
</tr>
<tr>
<td>“What [is] the test doing which is different from what my app is doing?”</td>
<td>30</td>
<td><strong>Compare</strong> test traces to app traces</td>
</tr>
<tr>
<td>How are these thread pools interacting?</td>
<td>24 19</td>
<td>Search downstream for <strong>calls</strong> into thread pools</td>
</tr>
</tbody>
</table>
Longest debugging activity

Where is method $m$ generating an error?

Rapidly found method $m$ implementing command
Unsure where it generated error

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

(66 minutes)
Why was this question so 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?
Why are control flow questions frequent?

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)
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** \( \cap \) **search criteria**

**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**

**feasible paths**
- **filter**
- **compare**

**search criteria**
- identifier
- statement type (field write/read, library call)
Reachability question: example

Where is method $m$ generating an error?

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

$\bigcap$ feasible paths statements matching search criteria
Longest activities related to reachability questions

### 4 out of the 5 longest investigation activities

<table>
<thead>
<tr>
<th>Primary question</th>
<th>Time (mins)</th>
<th>Related reachability question</th>
</tr>
</thead>
<tbody>
<tr>
<td>How is this data structure being mutated in this code?</td>
<td>83</td>
<td>Search downstream for <strong>writes</strong> to data structure</td>
</tr>
<tr>
<td>“Where [is] the code assuming that the tables are already there?”</td>
<td>53</td>
<td><strong>Compare</strong> behaviors when tables are or are not loaded</td>
</tr>
<tr>
<td>How [does] application state change when ( m ) is called denoting startup completion?</td>
<td>50</td>
<td>Find field <strong>writes</strong> caused by ( m )</td>
</tr>
<tr>
<td>“Is [there] another reason why ( \text{status} ) could be non-zero?”</td>
<td>11</td>
<td>Find statements through which values <strong>flow</strong> into ( \text{status} )</td>
</tr>
</tbody>
</table>

### 5 out of the 5 longest debugging activities

<table>
<thead>
<tr>
<th>Primary question</th>
<th>Time (mins)</th>
<th>Related reachability question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where is method ( m ) generating an error?</td>
<td>66</td>
<td>Search downstream from ( m ) for <strong>error</strong> text</td>
</tr>
<tr>
<td>What resources are being acquired to cause this deadlock?</td>
<td>51</td>
<td>Search downstream for <strong>acquire</strong> method calls</td>
</tr>
<tr>
<td>“When they have this attribute, they must use it somewhere to generate the content, so where is it?”</td>
<td>35</td>
<td>Search downstream for <strong>reads</strong> of attribute</td>
</tr>
<tr>
<td>“What [is] the test doing which is different from what my app is doing?”</td>
<td>30</td>
<td><strong>Compare</strong> test traces to app traces</td>
</tr>
<tr>
<td>How are these thread pools interacting?</td>
<td>19</td>
<td>Search downstream for <strong>calls</strong> into thread pools</td>
</tr>
</tbody>
</table>
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.
public int getFoldLevel(int line) : 1463 - 1475

```java
    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);
```

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
Step 2: Find statements matching search criteria

**Examples of observed reachability questions Reacher supports**

<table>
<thead>
<tr>
<th>Question</th>
<th>Steps to use Reacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>What resources are being acquired to cause this deadlock?</td>
<td>Search downstream for each method which might acquire a resource, pinning results to keep them visible</td>
</tr>
<tr>
<td>When they have this attribute, they must use it somewhere to generate the content, so where is it?</td>
<td>Search downstream for a field read of the attribute</td>
</tr>
<tr>
<td>How are these thread pools interacting?</td>
<td>Search downstream for the thread pool class</td>
</tr>
<tr>
<td>How is data structure <em>struct</em> being mutated in this code (between <em>o</em> and <em>d</em>)?</td>
<td>Search downstream for <em>struct</em> class, scoping search to matching type names and searching for field writes.</td>
</tr>
<tr>
<td>How [does] application state change when <em>m</em> is called denoting startup completion?</td>
<td>Search downstream from <em>m</em> for all field writes</td>
</tr>
</tbody>
</table>
Step 3: Help developers understand paths and stay oriented

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
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())?
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.
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
Needfinding
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 time
Ideation
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