User-Centered Design

SWE 632
Fall 2015
In class exercise

• Today’s question:

• What makes great software?
What makes great software?
Administrivia

- HW2 due today
- HW3 due in 1 week
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 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?
- What features does it have?
- How might users use it?
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
The Usability Life Cycle

Any project, even a small one, 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 creative. That is, the design team continually evaluates whether the design suits the users’ needs. Emphasis on usability throughout the Implementation and Deployment phases helps maintain a constant focus on usability. Then, the product is delivered. As you plan for the next project, determine what usability steps will best meet your user and business needs—this is the square, the most important step.

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 the main elements will be (such as menus). A flow diagram of the system elements can represent a navigation model.

User Scenarios

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. Interviews with the design team is a valuable tool, as it helps us understand how real world people and tasks can be used to improve design.

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. Different walkthrough techniques can be used to evaluate early concepts and designs.

Detailed Design Implemented Too Soon—Oops!

Congratulations! Now you have the info needed to make informed design decisions.

Document User Scenarios

Document user scenarios and develop a task analysis.

Create User Profiles

User profiles documents the various categories of users and their characteristics. They help a user get a handle on who will be using the system in what way. User profiles can include such information as demographic, technology experience, subject matter expert, attitudes and motivations, and frequency of use.

Task Analysis

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 do their work.

User Interaction

 eerste early often with users!

Start

The multidisciplinary team consists of experts in usability, human factors, marketing, graphic design, technology engineering, and project management. Collecting data for each discipline during the analysis, design, and implementation phases will ensure the design mixes usability needs, accommodate the technology being used, and match the interface in a way that will not overpower the user and reflect the intended functionality.

Field Studies

Field studies are an excellent way to gather information about the users’ environments and how they use real work in those environments. Field studies can include both traditional users and observing their behavior. However, these studies are often conducted in the users’ workplace. Data from field studies drives development of user profiles, task analysis, scenarios, and usability testing protocol.

Field Studies

Congratulations! Thorough groundwork increases your chance of success.

Field Studies

look at competitive products

Field Studies

Assemble a multidisciplinary team to ensure complete expertise!

Field Studies

Include usability tasks in the project plan.

Acknowledgements:

Meg Bass - digitalMag
Julia Newkirk - Questica Corporation
Dana Solomon & Larry Yerbury - OUL, Inc.
Charlotte Schumacher - Consultant

© 2000 Usability Professionals’ Association
Flexibility-usability tradeoff
Flexibility-usability tradeoff

• Jack of all trades, master of none

• Better understanding needs enables specialization and \textit{optimization} for common cases

• System evolution over time:
  
  • flexibility $\rightarrow$ 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
Observations of investigation & debugging in a complex codebase

Participants

13 developers
median 2.5 yrs industry experience

Tasks

90 minute investigate & fix design problem x 2
55 KLOC Java application (jEdit)

When painting the last line of a file, Buffer.isFoldStart() doesn't call getFoldLevel(), hence the foldLevelChanged() event might not be sent for the previous line. Identify and fix problems with this design:

**Code smells**
- Ignoring the return value of a getter
- Using getter for its effects

**Architecturally questionable**
- Changing buffer state from another component

Task 1

When a file is opened, a number of redundant UI updates are performed that reduce performance. Identify these updates and fix the design to reduce them.

Task 2
Data collected

Wide video camera
Camtasia screen video
observer notes

Demographic questions
Thinkaloud audio
Post task interviews
Code after changes
Participant typed notes
Notepaper video camera

Participant handwritten notes
# Transcripts and analysis

<table>
<thead>
<tr>
<th>time</th>
<th>code</th>
<th>goal</th>
<th>action</th>
<th>target</th>
<th>think aloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:56</td>
<td>Scrolling</td>
<td>[top]</td>
<td>Add fold level caching</td>
<td>Edit</td>
<td>Start adding field</td>
</tr>
<tr>
<td></td>
<td>Scrolling</td>
<td>[top]</td>
<td>ReferencesTo</td>
<td>getFoldLevel()</td>
<td>Is there a way to remember this position so I can come back to exactly this place? [didn’t know] Oh, I can remember the line number, right?</td>
</tr>
<tr>
<td>7:57</td>
<td>Scrolling</td>
<td>[top]</td>
<td>add field - int latestFoldLevel = 0</td>
<td>getFoldLevel()</td>
<td></td>
</tr>
<tr>
<td>7:58</td>
<td>Edit</td>
<td>OutlineTo</td>
<td>getFoldLevel()</td>
<td>adding method - setFoldLevel()</td>
<td></td>
</tr>
<tr>
<td>7:59</td>
<td>Edit</td>
<td>[copying and pasting code from getFoldLevel()]</td>
<td>getFoldLevel()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8:00</td>
<td>Reading</td>
<td>DPC</td>
<td>getFoldLevel()</td>
<td>Ok, so I’m thinking of having a hash table that associates each line number with the fold level of that line. And then when I...[Breakdown - LineManager already doing this] So then I can have two functions, one which modifies the fold level associated with the line number and the other that just gets it from the hashtable. So can I look up the hashtable API [yes, allowed to]</td>
<td></td>
</tr>
<tr>
<td>8:01</td>
<td>Edit</td>
<td>small fix to setFoldLevel()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8:01</td>
<td>Scrolling</td>
<td>JTextBuffer, latestFoldLevel</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(11,821 lines)

- Hierarchy of goals & actions
- Design decisions
- Length & reason for breakdowns
- Different strategies used for achieving the same goals
- Differences in pieces of system explored
- Code changes and defects introduced
Analysis - Facts

Developers navigated code to answer questions and learn facts about code.

Examples:

Whenever the window scrolls, the caret status must be updated.

Whenever the cursor moves, the caret status must be updated.

Whenever the buffer changes, the caret status should be updated once.

EditBus is for low frequency events, not high frequency events like buffer edits.

When the buffer change EditBus message is sent, the text area has not yet been updated with the new buffer's info.

Developers sometimes were unsuccessful answering their questions. Made optimistic or pessimistic assumptions.

Developers sometimes made false assumptions.
Examples of false beliefs and questions answered incorrectly

<table>
<thead>
<tr>
<th>False assumption</th>
<th>Correct fact about control flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method $m$ need not invoke method $n$, as it is only called in a situation in which $n$ has already been called.</td>
<td>$m$ is called in several additional situations in which $n$ has not been called.</td>
</tr>
</tbody>
</table>

**Question answered incorrectly**

<table>
<thead>
<tr>
<th>Why is calling $m$ necessary?</th>
<th>$m$ indirectly calls a function that updates the screen.</th>
</tr>
</thead>
</table>

![Diagram](image)
Findings

- Developers seek task-relevant information by asking questions and navigating code to learn facts about code.

- Developers built mental models (sometimes externalized in sketches and notes) of control flow.

- Developers sometimes hold false beliefs about code because they answered questions incorrectly or made false assumptions.

- False beliefs about control flow led developers to introduce defects.

32 changes:
- 16 inserted a defect
- 16 did not insert a defect
- 5 related to false assumption about control flow
- 3 related to question about control flow answered incorrectly
- 8 unrelated to control flow
Limitations

Study of developers making changes to codebase they’ve never seen before

➡  Maybe developers working with unfamiliar code in a familiar codebase do not have these challenges?

Two tasks in a single codebase

➡  Maybe other tasks or codebases do not have these challenges?

Are these challenges typical of real world software development?
Observations of developers in the field

**Participants**
17 professional developers

**Tasks**
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)
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>Question</th>
<th>Time (mins)</th>
<th>Related action</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>33 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>Method</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>
</tbody>
</table>

Found the answer

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

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

Reachability Questions
(common characteristics of evidence sought)
feasible paths \( \cap \) search criteria

Reachability Questions
(common characteristics of evidence sought)

A search along feasible paths downstream or upstream from a statement for target statements matching search criteria.
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 $e$)
## 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 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 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>40 19</td>
<td>Search downstream for <strong>calls</strong> into thread pools</td>
</tr>
</tbody>
</table>
Overall findings

› Found that developers can construct incorrect mental models of control flow, leading them to insert defects

› Found that 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

› Built formalism describing information needs in reachability questions
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
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
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 × 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.
REACHER helped developers stay oriented

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.”
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
Design in the world of business
Norman’s law of product development

• The day a product development process starts, it is behind schedule and above budget.

• Teams often not budgeted time for understanding users, iterating design

• In some markets, competitive forces can (sometimes) drive design evolution
Featuritis

• Existing customers like the product, but express wish for more feature, functions, capability

• Competing company adds features, producing competitive pressure to match and exceed

• Customers are satisfied but market saturated, leading to pressure for new features

• Leads products towards more power, but also more complexity

• Antidote: focus on customer and strengths, strengthen even more
Legacy problems

- Users used to existing version of system
- Changing system functionality may force users into relearning how to use system
- Discourages design innovation
Technology changes

• Fundamental user needs stable

• Technology enables new ways for these to be addressed
Group activity
Group activity

• In groups of 3 or 4

• Scenario: Your customers tell you, our organization is large, and we all just get too much email that wastes too much time. Build us a new communication and messaging system.

• Answer the following questions:
  
  • What would you focus on learning through needfinding?
  
  • What problems and practices might lead to this issue?
  
  • Make an (arbitrary) choice on which you think it is
  
  • Based on this choice, generate design ideas for addressing these issues