CSE 291 I: Usability of Programming Languages ("Programmers Are People Too")

Michael Coblenz
Homework

- Dylan on homework 1
- Homework 2
  - Rust error message challenges
  - Bank: use Coq instead of COBOL?
    - Reliability
    - Recruiting
USABILITY STUDY OVERVIEW

- Running usability studies requires:
  - Ethics approval
  - Recruiting
  - Training
  - Task design
  - Data collection/analysis
ETHICS REVIEW

• For research: need to submit proposal to Institutional Review Board (IRB)

• For class: no need to get IRB approval (IRB only supervises research)
ETHICS

• What if incentive is too high?
  • IRB reviews incentives

• What if incentive is too low?

• What if recruitment is misleading?
  • IRB reviews recruitment materials
PARTICIPANT PRE-SCREENING

• Can issue a pre-test to avoid wasting time on unqualified participants.

• Issues:
  • How will you incentivize people to take the test?
  • Can you use the test results in your research?
Which of the following might be a valid Java constructor invocation?

- malloc(sizeof(Square))
- Square.new(5)
- square(5)
- new Square(5)

In Java, **encapsulation** refers to:

- Preventing clients from improperly depending on internal details
- Serializing data correctly so that it is transmitted correctly
- Using the `capsule` keyword to protect secret data

```java
void test() {
    ArrayList list1 = new ArrayList();
    list1.add(1);
    ArrayList list2 = list1;
    list2.add(2);
    System.out.println(list1.size());
}

If test() is run, what is the output?

1
2
```

Which statements are true of interfaces in standard Java?

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interfaces have no field declarations unless they are public static final.</td>
<td>❌</td>
</tr>
<tr>
<td>Methods in interfaces are public by default.</td>
<td>❌</td>
</tr>
<tr>
<td>Methods in interfaces (except for default methods) lack bodies.</td>
<td>❌</td>
</tr>
<tr>
<td>A class can implement no more than one interface.</td>
<td>❌</td>
</tr>
</tbody>
</table>
DEMOGRAPHICS

- Collect information if you want it!
- Programming experience? Languages?
- If they tell you, you can use it...
- e.g. Gender__________
TRAINING

• How will you prepare your participants?

• People don't read.

• People think they understand but in fact do not.

• Teach…and then assess.

• Or: decide that no training is necessary.
Obsidian Tutorial

- Ownership – Introduction
  - Principles of ownership
- Ownership – Transactions
  - Transaction return types
  - Transaction parameters
  - Transaction receivers (this)
- Ownership – Variables
  - Assignment
  - Fields
  - Local variables
  - Constructors
- Ownership – Miscellaneous
  - Ownership checks
  - Getting rid of ownership
  - Invoking transactions
  - Handling Errors
  - Return
- Assets
- States – Introduction
  - States and Ownership
- States – Manipulating State
  - The { → } Operator
  - Alternative field initialization
  - Optional compiler checks
  - Testing states with in
- States – Miscellaneous
  - Unowned references
  - Shared references
  - Implicit casts
- States and Assets
- Using Obsidian on a Blockchain
  - Concurrency
Write a contract called Person that has an Owned reference to a House and a Shared reference to a Park. The House and Park contracts are given below.

```solidity
contract House {
}
contract Park {
}
```

What is m in the above code fragment above?

- A Money object
- An Owned reference to a Money object
- An Owned object
- All of the above
- None of the above
TASKS

• This is the hardest part of study design.

• You will not get this right the first time.

• Solution: pilot repeatedly.

• But: you can use data from your "pilots" if you follow protocol.

• (a true "pilot" involves throwing the data out)

• What is the distribution over task times?
USABILITY STUDY TASKS

• Choose an *interesting* task
  • One that you think might be hard
  • One that is central to the usability of your design
• Can't test everything
TASK IDEAS

• Write a program according to this specification.
• Are there bugs in this code? If so, what are they?
• Fill in the missing code…
• What does this code do?
• Answer these questions about this code.
TASK DESIGN

• Must carefully restrict tasks!

• People will get stuck on irrelevant things

• Decide how much help to provide

• Ideally: scope task to focus on the variable of interest

• Constrain the task as much as possible.
DECOMPOSING TASKS

Fig. 2. The horizontal axis represents time; the vertical axis represents a dependent variable measured in a study. Part (a) shows how the variance increases over time. Shading shows how frequently a particular point in the space might be reached over many participants. In part (b), the task has been divided into three subtasks to reduce the variance in each subtask.

3.9 Summary

Table 1 summarizes the approaches we have found effective when designing user studies of programming languages.

4 USABILITY STUDIES FOR GLACIER

4.1 Formative studies

We used the Cognitive Dimensions of Notations framework [27] to reason about some of the design choices. For example, including features that provided weaker guarantees than programmers actually needed could be error-prone if those features could be easily confused with stronger ones. Likewise, the inverse is error-prone too: if a programmer applied a weaker specification than could actually be applied, this could lead to undesirable tradeoffs. For example, if an interface is annotated to return a read-only object (to an object that could be mutated through other references), the programmer might add locks to ensure safety in a concurrent context. But if the object is actually immutable (that is, no reference could be used to mutate the object), then the locks would be unnecessary and reduce performance.

Although the Cognitive Dimensions analysis was lightweight, it did not answer some of our higher-level design questions. In order to narrow the space of possible language designs, we conducted semi-structured interviews with eight software engineers who were working on large software projects at several organizations. Our participants had an average of fifteen years of experience, with a minimum of seven years, and had worked on projects with millions of lines of code and hundreds of people.

In order to both obtain unbiased data on problems with mutability in general as well as to obtain feedback on concrete language designs, we carefully ordered the interview questions. First we asked general questions, such as "How do you make sure that state in running programs remains valid?" We got wide-ranging answers, including ones such as "We've essentially done away with mutability to avoid security and concurrency problems" as well as recommendations for regular use of testing and assertions. Afterward, we asked about existing language features, such as const.
DATA COLLECTION

- Think-aloud
- Audio recordings
- Videos
- Screen capture
- Eye tracking
- Post-study survey

- Take lots of notes!, including timestamps! You do not want to watch the videos.
- Include a clock on the screen.
THINK-ALOUD

- Two varieties: concurrent and retrospective
- "Please keep talking."
- Can't use timing as a dependent variable due to effect of explanations.
TASK CONTEXTS

- Pencil/paper
- Text editor
- IDE
- Compiler?
- Debugger?
- Test cases?
TASK EXAMPLES

• Q: What challenges do web programmers encounter when using PHP to write web apps?
  • (who?)

• Plan: Recruit people who say they've made at least one web site in PHP
  • (what does one web site mean?)

• Task attempt 1: "write a gradebook app in PHP. You have 1 hour."
  • What is a gradebook app?
  • Where do you think people will get stuck?
TRY 2

• Refine task:

  • Here is a gradebook app, but the component that shows a student their grades is incomplete. Write `displayGrades()`, which will display a student's grades.

  • What format?

  • Are you measuring PHP, or the MySQL API, or the particular database schema, or something else?
TRY 3

• Refine research question
• What problems do PHP programmers encounter when handling errors?
• Put them in a situation where they will encounter errors!
• What kind of errors?
• Hypotheses:
  • They will forget to check for errors
  • They will misinterpret error codes
  • They will have trouble figuring out the causes of errors they encounter, even when the errors are common (e.g. "server unreachable")
contract Auction {
    Participant.Owner seller;

    state Open;
    state BidsMade {
        // the bidder who made the highest bid so far
        Participant.Owner maxBidder;
        Money.Owner maxBid;
    }
    state Closed;

    ... 

    transaction bid(Auction.Owner this,
        Money.Owner >> Unowned money,
        Participant.Owner Unowned bidder) {
        if (this in Open) {
            // Initialize destination state,
            // and then transition to it.
            BidsMade::maxBidder = bidder;
            BidsMade::maxBid = money;
            ->BidsMade;
        }
        else {
            if (this in BidsMade) {
                // if the newBid is higher than the current Bid
                if (money.getAmount() > maxBid.getAmount()) {
                    // if the newBid is higher than the current Bid
                    maxBidder.receivePayment(maxBid);
                    maxBidder = bidder;
                    maxBid = money;
                    pendingReturns[maxBidder] += maxBidAmount;
                } else {
                    bidder.receivePayment(money);
                }
            } else {
                revert("Can only make a bid on an open auction.");
            }
        }
    }
}

CONCLUSION

• Running usability studies requires:
  
  • Recruiting
  
  • Training
  
  • Task design
  
  • Data collection/analysis
  
• Task design is probably the trickiest. Start early and pilot!