User-Centered Programming Language Design: an ICFP Tutorial

Michael Coblenz and Jonathan Aldrich

Carnegie Mellon University

institute for SOFTWARE RESEARCH

HCII

LANGUAGES ARE INTERFACES

- PLs are interfaces for *humans* to use to write programs.
- Therefore they should be subject to the principles and methods of HCI.
- For the rest of the tutorial:
  - What are these principles and methods?
  - How can we apply them to programming languages?

AGE-OLD DESIGN DECISIONS

- Some of you probably have strong opinions about these
- In some cases, most people agree
- In others, the question seems to be "religious"!

DESIGN QUESTIONS

- Dynamic scoping in LISP
  - Widely regarded as a mistake (but it was easy to implement!)
- Laziness in functional languages
  - Does this have to be a religious war?
  - Is there a right answer?
- SML equality types (which things can I use = on, again?)
- Order of evaluation of function arguments in OCaml (right to left)

ARE TYPES USEFUL?

- Preliminary studies: yes
  - But depends on the setting...something PL designers should know
- More interesting: *how* are they useful?
  - Evidence: documentation benefits >> correctness benefits

USER-CENTERED METHODS CAN IMPACT FUNDAMENTAL* DECISIONS

- In Glacier (immutability types for Java):
  - Decision to support transitive immutability (interviews)
    - more useful vs. immutability for a single object, or vs. read-only references
  - Decision not to support parameterization by immutability (user studies)
    - complexity posed major barrier to both usability and correctness
  - Both decisions sacrificed (nonessential) expressiveness

* Fundamental = had a big effect on the formal system

USER-CENTERED METHODS CAN IMPACT FUNDAMENTAL* DECISIONS

- In Obsidian (smart contract PL)
  - Decision that typestate implies ownership (user studies)
    - reduced complexity (and expressiveness) vs. keeping them orthogonal
  - Decision to support dynamic state tests (user studies)
    - fulfilled user expectations (and added needed expressiveness)
    - low user-visible complexity, but challenging to get the formal semantics right

* Fundamental = had a big effect on the formal system

TUTORIAL OUTLINE

- What is usability?
- Overview of methods
- Practical considerations: ethics, recruitment, participants
- Usability studies
- Other kinds of qualitative studies
- Quantitative studies
- You will have a chance to practice today.

PART 1: WHAT IS USABILITY?

WHAT IS USABILITY?

- "The extent to which a product can be used by *specified users* to achieve *specified goals* with *effectiveness*, *efficiency* and *satisfaction* in a specified *context of use*." (ISO 9241-11, Ergonomics of human-system interaction)

USABILITY

https://pixabay.com/vectors/speedometer-kilometers-dashboard-309118/

USABILITY

- Who are the users?
- What are the users trying to do?
- In what context are they doing it?
- How well do they do it?
- How much work does it take them to do it?
- Do they like it?

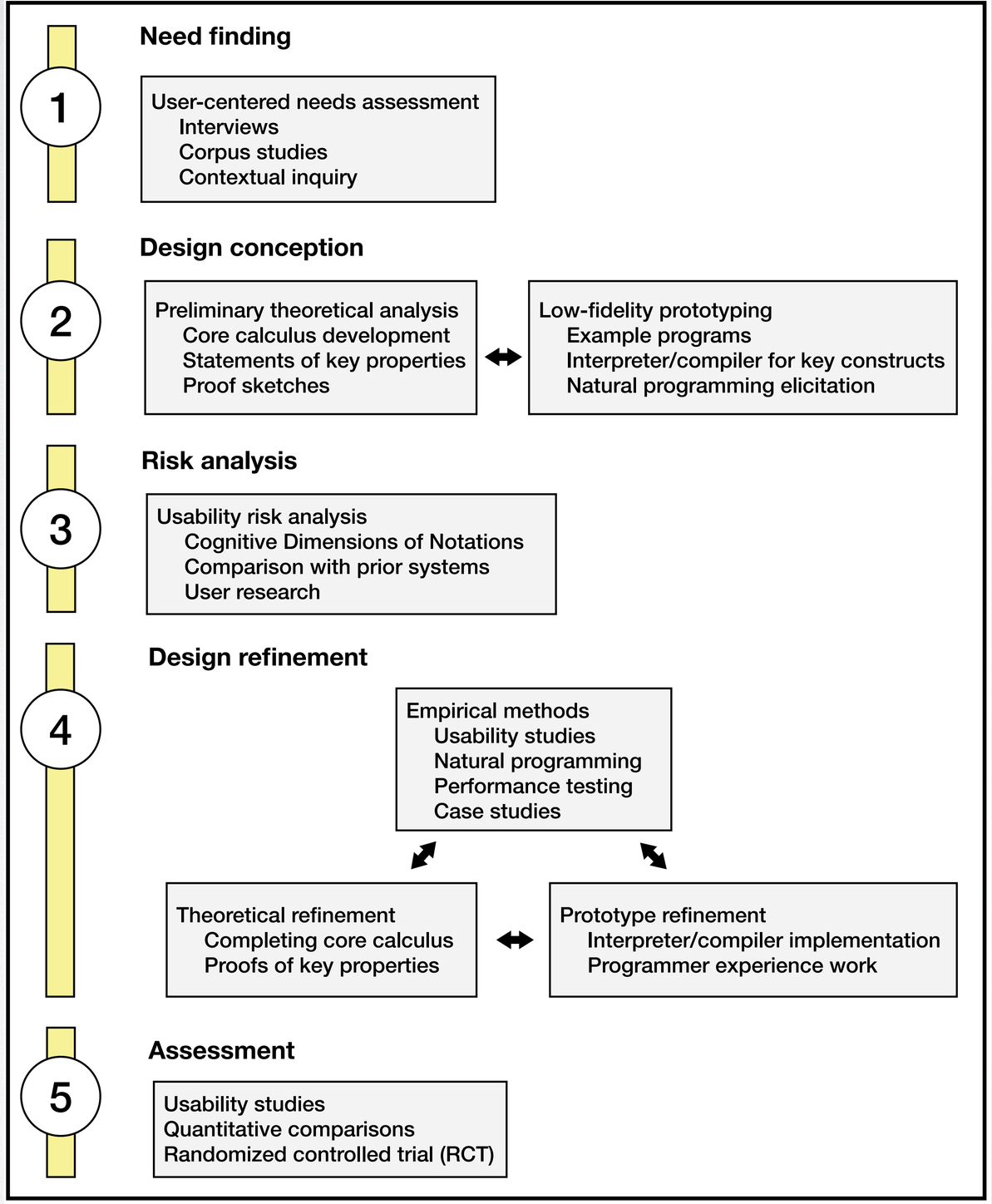
FORMATIVE METHODS CAN HELP ANSWER…

- (Formative methods: *before* your design is complete)
- What problems or goals do users have?
- How common are those problems?
- What are the most significant problems with the design I have so far?
- How do users *already* express solutions to their problems?

SUMMATIVE METHODS CAN HELP ANSWER…

- (Summative methods: *after* your design is complete)
- What fraction of participants can complete task T in N minutes?
  - Which ones succeeded? Which failed?
- What is the distribution over completion times for task T?
- Do more participants succeed if they use language A instead of language B?
- Are there fewer bugs if people use language A instead of language B?

PLIERS: **P**rogramming **L**anguage **I**terative **E**valuation and **R**efinement **S**ystem



WHO ARE THE USERS?

- Scope your target audience
  - Education
  - Experience
  - Knowledge
  - Skills
  - Work context

"The extent to which a product can be used by ***specified users*** to achieve *specified goals* with *effectiveness*, *efficiency* and *satisfaction* in a specified *context of use*."

A MANTRA: "THE USER IS NOT LIKE ME."

- Most programmers are not like ICFP participants!
  - Are only barely aware of functional programming
  - Deal with mutable state all the time
  - Use the language their manager told them to use
  - Haven't heard of lambda calculus
- Plus, even if you target functional programmers, they might differ from you in other ways!

"WHO CARES? MY LANGUAGE IS FOR PEOPLE LIKE ME."

- A language is a communication tool for *humans*
- Used over a period of time
- Inevitably, software has to be maintained
- Want to broaden participation, not narrow it!

PERSONAS

- Help you think like the user
- Build empathy

CHARLIE

Charlie is in her late 20s to mid 30s. She has a Bachelor's degree but not necessarily in IT. She's a self-taught developer. Her coding is unconventional and she mixes genius lines with simple errors. She seeks to reinvent her software development career but the how is still unclear. Charlie has a family, which makes financial stability and work-life balance essential. She's new to the industry and thus looks for a company that offers a supportive, people-oriented environment, where she can learn and improve her skills.

https://www.meistertask.com/blog/there-are-four-archetypal-developer-personas-which-one-are-you/

MICROSOFT PERSONAS (CLARKE)

THE SYSTEMATIC DEVELOPER

Writes code defensively. Does everything they can to protect their code from unstable and untrustworthy processes running in parallel with their code.
Develops a deep understanding of a technology before using it.
Prides themselves on building elegant solutions.

THE PRAGMATIC DEVELOPER

Writes code methodically.
Develops a sufficient understanding of a technology to enable them to use it. Prides themselves on building robust applications.

THE OPPORTUNISTIC DEVELOPER

Writes code in an exploratory fashion.
Develops a sufficient understanding of a technology to understand how it can solve a business problem.
Prides themselves on solving business problems.

USABILITY GOALS

- What does the user want to achieve?
- Start vague, and then drill down
- What does a C programmer want to achieve?
  - Fix a bug — what kind?
  - Implement something — what?
  - Understand code — for what purpose?

"The extent to which a product can be used by *specified users* to achieve ***specified goals*** with *effectiveness*, *efficiency* and *satisfaction* in a specified *context of use*."

CONTEXT

- Starting from scratch, maintaining a system, or legacy code?
- Large team, or lone developer?
- Beginning of project, or near shipping (risk-averseness)?
- One-off, or repeated task?

"The extent to which a product can be used by *specified users* to achieve *specified goals* with *effectiveness*, *efficiency* and *satisfaction* in a specified ***context of use***."

HOW WELL DO THEY DO IT?

- Count and describe bugs/errors
- How readable is their code?
  - What does "readable" mean, anyway?
- What fraction of target population can do it?

"The extent to which a product can be used by *specified users* to achieve *specified goals* with ***effectiveness, efficiency*** and *satisfaction* in a specified *context of use*."

HOW HARD WAS IT?

- What fraction of participants succeeded?
- How long does it take?
- What obstacles did they encounter along the way?
- Other measures
  - Galvanic skin response (emotional response)
  - EEG (cognitive load)

"The extent to which a product can be used by *specified users* to achieve *specified goals* with ***effectiveness, efficiency*** and *satisfaction* in a specified *context of use*."

DO THEY LIKE IT?

- Of course they do, if it's YOUR system!
- "We find that respondents are about 2.5x more likely to prefer a technological artifact they believe to be developed by the interviewer, even when the alternative is identical." [Dell et al., CHI 2012]
- "When the interviewer is a foreign researcher requiring a translator, the bias towards the interviewer's artifact increases to 5x."

"The extent to which a product can be used by *specified users* to achieve *specified goals* with *effectiveness*, *efficiency* and ***satisfaction*** in a specified *context of use*."

SO WHAT IF THEY LIKE IT?

- Adoption
- "Why not?"
- But maybe you don't care!

KEY TAKEAWAY: MANY KINDS OF USABILITY

- My "more usable" system might be better because:
  - Learnability: it's easier/faster to learn
  - Task performance: people (who?) finish tasks (which?) faster or more people finish tasks
  - Audience: a new kind of person can do the task
  - (Not an exhaustive list)

YOUR TURN

- Identify a usability question YOU have about a COMMON PL.
  - Discuss in breakout rooms (on Clowdr, 10 minutes)
  - Share afterward.

"The extent to which a product can be used by *specified users* to achieve *specified goals* with *effectiveness*, *efficiency* and *satisfaction* in a specified *context of use*."

PART 2: METHODS OVERVIEW

CATEGORIES OF METHODS

- Qualitative methods
  - Focus is on *depth* of data
  - Does not imply *no quantities*
- Quantitative methods
  - Focus is on *statistical analysis* of data

MAIN FOCUS TODAY: QUALITATIVE STUDIES

- Easier and lower-cost to run
- Identify problems quickly
- Iterate rapidly
- We will discuss quantitative studies somewhat
  - But we can't teach you enough statistics right now

GENERATING HYPOTHESES

NOT JUST ANY HYPOTHESES…

- Want to only test hypotheses that are probably true.
- You can publish a paper even if all you have is a hypothesis!
  - (if it is well-justified)
  - HATRA, PLATEAU workshops (at SPLASH)
- And what if your  is empty?

STAGES

- I don't know what I'm doing.
  - What problems are there to solve?
  - What hypotheses are worth testing?
- I have a tool. Let's make it better.
- I have a tool. Let's try to show that it IS better.

Qualitative studies

QUALITATIVE STUDIES

- Want to understand something we don't understand yet.
  - What problems do factory workers have?
  - What is it like to write code for race cars?
    - [ICSE 2015 keynote, Claudio Silenzi]
  - What usability problems do people have when they use my "awesome" system?

USABILITY STUDIES

- Give people tasks and observe what happens.
- NOT experiments
- NOT controlled
- NOT comparative
- Just want to see what problems people encounter.

USABILITY STUDIES CAN SHOW

- X% of my participants completed the task in 30 minutes.
- Participants encountered the following problems…
- Only participants who knew X were able to do the task.

USABILITY STUDY RESULT EXAMPLE

- Context: type system where some references are *owning* references.
- P18 asked what happens when passing an `@Owned` object to a method with an unowned formal parameter (ownership was not passed in this case).
- P19 said, "when I [annotate this constructor type `@Owned`], I'm not sure if I'm making a variable owned or I'm transferring ownership."
- P17 was surprised that assignment from an owned reference to an unowned-type variable did not transfer ownership.
- Conclusion: ownership transfer was confusing.
- Attempted solution: made changes of ownership explicit. Problem: too verbose!
- Revised solution: specify ownership changes in formal parameters. Assignment always transfers ownership when possible.

USABILITY STUDIES CANNOT SHOW

- My system is better than an existing system.

PART 3: PARTICIPANTS AND PRACTICALITIES

WHO SHOULD PARTICIPATE?

If you want to argue your results generalize to X, then ideally you should sample from X.

Plan B: argue X is similar to the population you sampled from.

Examples?

Population of interest

AN APPROACH

- 41% of professional developers have <= 5 years of professional experience [StackOverflow developer survey 2019]
- Master's students frequently have 0-5 years of professional experience (or >= 2, depending on program)
- Therefore, Master's students may be representative of *some* professional developers

RECRUITMENT

- Flyers
- Emails
- Social network
- Buy ads
- The street

See: Report from Dagstuhl Seminar 19231 Empirical Evaluation of Secure Development Processes

INCENTIVES

- $$$ (in person, MTurk)
- Desire to contribute to science / help you out
- Food
- Fame (leaderboard)
- Rare experience
- Learning opportunity
- Distraction from work
- Credit

TARGETS

- Programmers
  - Architects
  - Code reviewers
- Testers
- Security teams
- Designers
- Domain experts
- End users
- Requirements engineers

ETHICS

- What if incentive is too high?
- What if incentive is too low?
  - IRB reviews incentives
- What if recruitment is misleading?
  - IRB reviews recruitment materials

PARTICIPANT PRE-SCREENING

- Can issue a pre-test to avoid wasting time on unqualified participants.
- IRB typically likes to review the pre-test.
- 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
- Serializing data correctly so that it is transmitted
- Using the `capsule` keyword to protect secret da

```
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

Do not use any external resources to answer this question.

Which statements are true of interfaces in standard Java?

	True	False
Interfaces have no field declarations unless they are `public static final`.	○	○
Methods in interfaces are public by default.	○	○
Methods in interfaces (except for `default` methods) lack bodies.	○	○
A class can implement no more than one interface.	○	○

YOUR TURN

- Assume you have some tasks for the usability question from before.
- Who will you try to recruit?
- How will you:
  - Recruit?
  - Incentivize?
  - Screen?

PRACTICALITIES OF RUNNING STUDIES

- If you are in academia, you need Institutional Review Board approval
  - IRB makes sure ethics rules are being followed
  - IRB weighs benefits and risks
  - Protects both you and the participants
- Process depends on your institution

Consent Form for Participation in Research

Study Title: CMU Science of Security Lablet: Taking on the Hard Problems

Principal Investigator: Michael Coblenz, Doctoral Student
Computer Science Department, 5000 Forbes Ave, Pittsburgh, PA 15213
412-956-9500, mcoblenz@cs.cmu.edu

Faculty Advisor: Brad Myers

Other Investigator(s): Jonathan Aldrich, Joshua Sunshine, Gauri Kambhatla

Sponsor(s): National Science Foundation, US Department of Defense

---

Purpose of this Study
The purpose of the study is to inform and evaluate the design of a programming language for blockchain computing platforms.

Procedures
In this study, you will be asked to complete programming-related tasks, such as writing code, reading code, or debugging. By doing so, you can help us refine our design of a blockchain-oriented programming language or compare it to existing approaches for blockchain programming.

While you are working, we will record audio and video of your screen. We may also use an eye tracker to record where on the screen you are looking during the study. The recordings will only be available to investigators in the study, but we may use anonymized quotes, screenshots, or code samples in our publications.

The study will take approximately four hours and will take place on the Carnegie Mellon campus.

Participant Requirements
Participants must be at least 18 years old and have at least a year of experience programming in Java. If you have photosensitive epilepsy or use medical devices that are susceptible to disturbance by IR light or magnetic fields, please inform the experimenter that you cannot use the eye tracking device.

Risks
The risks and discomfort associated with participation in this study are no greater than those ordinarily encountered in daily life or during software engineering work. Risks include repetitive strain injury, but the risks are very low in a short duration study.

Benefits
There may be no personal benefit from your participation in the study but the knowledge received may be of value to humanity. We will use the results to improve our design of a blockchain programming language, which may result in more reliable software in a variety of domains.

Consent Form for Participation in Research

Compensation & Costs
We will compensate participants with a $75 Amazon gift certificate for completing the study. Partial payment will be given to participants who do not complete the study. Participants are responsible for reporting income on their tax returns as applicable.

There will be no cost to you if you participate in this study.

Confidentiality
By participating in the study, you understand and agree that Carnegie Mellon may be required to disclose your consent form, data and other personally identifiable information as required by law, regulation, subpoena or court order. Otherwise, your confidentiality will be maintained in the following manner:

Your data and consent form will be kept separate. Your research data will be stored in a secure location on Carnegie Mellon property. Sharing of data with other researchers will only be done in such a manner that you will not be identified. By participating, you understand and agree that the data and information gathered during this study may be used by Carnegie Mellon and published and/or disclosed by Carnegie Mellon to others outside of Carnegie Mellon. However, your name, address, contact information and other direct personal identifiers will not be mentioned in any such publication or dissemination of the research data and/or results by Carnegie Mellon. Note that per regulation all research data must be kept for a minimum of 3 years.

(1) Each participant will be assigned a number; (2) The researchers will record any data collected during the study by number, not by name; (3) Any original recordings or data files will be stored in a secured location accessed only by authorized researchers. Data that are stored electronically will be stored on an encrypted volume.

This research is being sponsored by the U.S. Department of Defense. DoD representatives are authorized to review research records.

Optional Permission
I understand that the researchers may want to use a short portion of any audio, screen video, or eye tracking recording for illustrative reasons in presentations of this work for scientific or educational purposes. I give my permission to do so provided that my name and face will not appear.

Please initial here: _____YES _____NO

Rights
Your participation is voluntary. You are free to stop your participation at any point. Refusal to participate or withdrawal of your consent or discontinued participation in the study will not result in any penalty or loss of benefits or rights to which you might otherwise be entitled. The Principal Investigator

may at his/her discretion remove you from the study for any of a number of reasons. In such an event, you will not suffer any penalty or loss of benefits or rights which you might otherwise be entitled.

Right to Ask Questions & Contact Information

If you have any questions about this study, you should feel free to ask them now. If you have questions later, desire additional information, or wish to withdraw your participation please contact the Principal Investigator by mail, phone or e-mail in accordance with the contact information listed on the first page of this consent.

If you have questions pertaining to your rights as a research participant; or to report concerns to this study, you should contact the Office of Research Integrity and Compliance at Carnegie Mellon University. Email: irb-review@andrew.cmu.edu . Phone: 412-268-1901 or 412-268-5460.

Voluntary Consent

By signing below, you agree that the above information has been explained to you and all your current questions have been answered. You are encouraged ask questions about any aspect of this research study during the course of the study and in the future. By signing this form, you agree to participate in this research study. A copy of the consent form will be given to you.

_____

PRINT PARTICIPANT'S NAME

_____ _____

PARTICIPANT SIGNATURE DATE

I certify that I have explained the nature and purpose of this research study to the above individual and I have discussed the potential benefits and possible risks of participation in the study. Any questions the individual has about this study have been answered and any future questions will be answered as they arise.

_____ _____

SIGNATURE OF PERSON OBTAINING CONSENT DATE

DEMOGRAPHICS

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

Participant code_____

Pre-study questionnaire

1.How long have you been programming?_____ years _____ months

2.Gender: _____

3.If you have any academic computer science background, what degrees have you completed? If you are partway through a degree, what degree and how far?

_____

4.How much professional (paid) software development experience do you have?_____ years _____ months

5.List any programming languages in which you are currently comfortable programming in DECREASING order of familiarity.

_____

6.How much experience do you have programming in Java? Include time spent doing Java part-time, such as in a course.

_____ years _____ months

7.Please rate your level of expertise in Java by circling one option:

beginner intermediate advanced

8.Please rate your level of expertise in Rust by circling one option:

none beginner intermediate advanced

9.Please rate your level of expertise in each of the following blockchain programming languages/environments:

Solidity:

I don’t know what this is/none beginner intermediate advanced

Hyperledger Fabric:

I don’t know what this is/none beginner intermediate advanced

Other (specify which)_____

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

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.

```
contract House {

}

contract Park {

}
```

Please write your answer in the VSCode window (code1.obs). You may compile your code to check your answer.

```
contract Money {
    ...
}

contract Wallet {
    Money@Owned m;

    Wallet@Owned() {
        m = new Money();
    }

    transaction spendMoney() returns Money@Owned {
        ...
    }

    transaction receiveMoney(Money@Owned >> Unowned mon) {
        ...
    }
}
```

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

PART 4: USABILITY STUDIES

TASKS, IN GENERAL

- 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

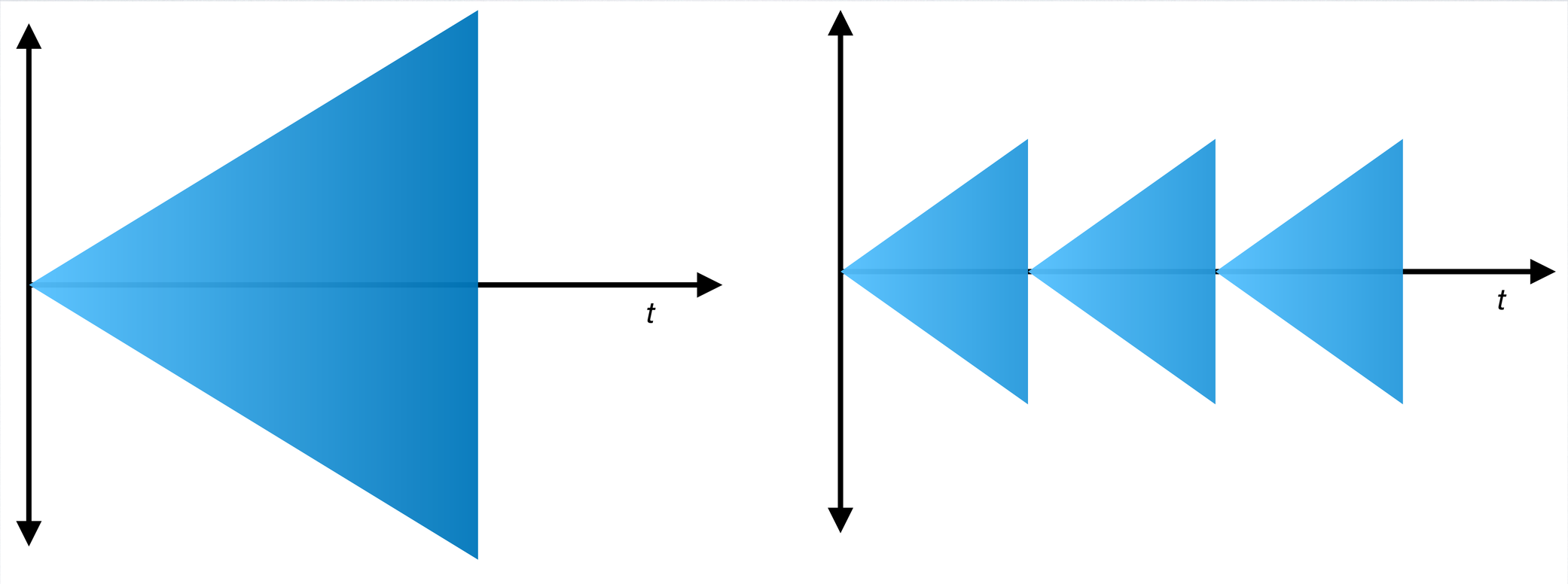
TASK IDEAS

- 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

DECOMPOSING TASKS (AN ILLUSTRATION)



Monolithic task

Subtasks

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?

```
main asset contract Auction {
  Participant@Unowned seller;

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

  . . .

  transaction bid(Auction@Shared this,
                  Money@Owned >> Unowned money,
                  Participant@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()) {
            //1. TODO: fill this in.
            //   You may call any other transactions as needed.
            maxBidder.receivePayment(maxBid);
            maxBidder = bidder;
            maxBid = money;
          }
          else {
            //2. TODO: return the money to the bidder,
            //   since the new bid wasn't high enough.
            //You may call any other transactions as needed.
            bidder.receivePayment(money);
          }
        }
        else {
          revert ("Can only make a bid on an open auction.");
        }
      }
    }
}
```

```
contract Auction {
  // the bidder who made the highest bid so far
  address maxBidder;
  uint maxBidAmount;

  // 'payable' indicates we can transfer money to this address
  address payable seller;

  // Allow withdrawing previous money for bids that were outbid
  mapping(address => uint) pendingReturns;

  enum State { Open, BidsMade, Closed }
  State state;
  . . .
  function bid() public payable {


    if (state == State.Open) {
      maxBidder = msg.sender;
      maxBidAmount = msg.value;
      state = State.BidsMade;
    }


    else {
      if (state == State.BidsMade) {
        //if the newBid is higher than the current Bid
        if (msg.value > maxBidAmount) {
          //1. TODO: fill this in.
          //   You may call any other functions as needed.
          pendingReturns[maxBidder] += maxBidAmount;
          maxBidder = msg.sender;
          maxBidAmount = msg.value;
        }
        else {
          //2. TODO: return the newBid money to the bidder,
          //   since the newBid wasn't high enough.
          //You may call any other functions as needed.
          pendingReturns[msg.sender] += msg.value;

        }
      }
      else {
        revert ("Can only make a bid on an open auction.");
      }
    }
  }
}
```

YOUR TURN

- Context
- Materials
- Task format
- Data collection
- Evaluation

- Design one task for a usability study of your language feature.
- For now, assume you have a full implementation.
- Take 10 minutes.

POST-STUDY SURVEY

- Collect quotes you can use
- Did people like it?

Post-programming study survey

Thank you for participating in the programming language design study. We would greatly appreciate it if you could fill out the following quick survey to tell us what you thought.

* Required

How much did you like the language you used?

	1	2	3	4	5	
Didn't like it at all	○	○	○	○	○	Liked it a lot

How well do you feel you understand the concept of ownership?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Completely understand it

How useful do you think ownership is, as implemented in the language you used?

	1	2	3	4	5	
Not at all useful	○	○	○	○	○	Very useful

How well do you feel you understand the concept of states?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Completely understand it

How useful do you think states are, as implemented in the language you used?

	1	2	3	4	5	
Not at all useful	○	○	○	○	○	Very useful

How well do you feel you understand the concept of assets?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Completely understand it

How useful do you think assets are, as implemented in the language you used?

1	2	3	4	5
○	○	○	○	○

How well do you feel you understand the relationship between ownership and compile-time knowledge of states?

	1	2	3	4	5	
Don't understand it at all	○	○	○	○	○	Understand it completely

Please use this space to write any additional comments you have about the language or the study.

Your answer

Do not edit the pre-filled answer to this question. *

○ S

○ O

HOW MANY?

- Until it's not useful anymore.
- Fix problems if you can, and repeat.
  - Help people past problems (so they can try the rest of the task)
- Or you can see how frequently problems are encountered.

SCAFFOLDING

- I only have a language *prototype*. How do I evaluate my language?

WIZARD OF OZ

- Have an experimenter simulate incomplete parts of the system
  - e.g., we simulated Obsidian compiler errors before implementation
- Can give any kind of programming task this way if it doesn't require running the code.

NATURAL PROGRAMMING

- Want to find out how people "naturally" specify software.
- Yes, depends on prior experience
  - Find people without experience
  - Put people in a novel situation
- Constrain choices

EXAMPLE

- Gave seven participants a description of a voter registration system.
- "Please implement this system in pseudocode using any language features you want."
  - Two invented syntax denoting states and transitions.
  - Many of the rest were unsafe, e.g. separate lists for unregistered and registered voters.
- "Here's a state diagram that models the system. Modify your pseudocode to use states and transitions."
  - Two participants: explicit state blocks with variables nested inside

BACK-PORTING DESIGN CHOICES

- Context: designing a feature in a new language. Want to:
  - Analyze choice A
  - Compare choices A and B
- Teach people language L + A, L + B?
  - People get confused: was it L, or A?
- Alternative: add A to a *standard language*.

BACK-PORTING EXAMPLE

- Wanted to assess usability of typestate
- Teach people Obsidian?
- No, add typestate to Java.

PART 5: QUALITATIVE STUDIES, MORE GENERALLY

QUALITATIVE STUDIES, MORE GENERALLY

- Data sources:
  - In-depth, open-ended interviews
  - Direct observation
  - Written documents
- "Research, like diplomacy, is the art of the possible." (Patton)

ETHNOGRAPHY

- Immerse yourself in the environment.
- Long-form results (e.g. a book)
- Example: "This is what I learned from a year living with the X people."

INTERVIEWS

- Prepare questions in advance
- "Semi-structured" — not restricted to your script
- Ask about experiences

PREPARING AN INTERVIEW

- Write questions in advance
  - Including probes (follow-ups)
- Ask for permission to record
- Phrase questions neutrally
- Progress from general to specific

Specific probes, but also:
"How do you mean that?"
"Tell me more about that."
"Anything else?"

1. How long have you been programming professionally?
2. Can you give an order of magnitude estimate of the size of the largest project you've made significant contributions on? Number of people, lines of code?
3. In what programming languages do you consider yourself proficient?
4. How did you get into software development? Do you have a computer science background?
5. Let's talk about changes that happen to state in software you've worked on. Many kinds of software maintain state, such as object graphs, files, or databases, but there's a possibility of corruption during changes due to bugs. How do you make sure that state in running programs remains valid?
    1. Are there specific techniques do you use? If so, what are they?
    2. Do you sometimes want to make sure that some operations don't change any state or don't change certain state?
        1. Tell me about a recent time you did this.
        2. How often does this come up?
        3. Do you use language features to help?
    3. Do you sometimes want to make sure that some state never changes?
        1. Tell me about a recent time you did this.
        2. How often does this come up?
        3. Do you use language features to help?
    4. Do you sometimes want to make certain kinds of modifications to state impossible for some users of an API but not others? If so, how do you do that?
6. How often do you work on concurrent aspects of software? What mechanisms do you use to control concurrency?
    1. Do you use immutability to help address or prevent concurrency issues?
7. How much work have you done on security-related aspects of your software? Have you found or fixed any vulnerabilities?
    1. Do you use immutability to help address or prevent security issues?
8. Can you recall a recent situation in which you created an immutable class or other data? If so, tell me about it.
9. Can you recall a recent situation where you changed a class from immutable to mutable? If so, tell me about it.
10. Can you recall a recent situation in which you changed a class from mutable to immutable. If so, tell me about it.
11. Can you think of a bug you investigated or fixed that was caused by a data structure changing when it should not have? What was the problem and how did you solve it (if you solved it)?
    1. Would const have prevented the bug?
12. Have you ever tried using an immutable class and had it not work? Why not?
13. When you create a new class, how do you decide whether instances should be immutable?
14. Have you ever been in a situation where you wanted to use const or final but it didn't say what you wanted to say?
    1. or where you discovered you couldn't use it? What was the situation and why couldn't you use it?
15. Have you been in a situation where you had to revise your plan because something you'd assumed could mutate state was disallowed from doing so due to const?
16. Have you been involved in training new members of the team? What do you tell new members about immutability or ensuring invariants are maintained?
17. Sometimes, though an object is mutable after creation, it only needs to be changed for a short amount of time. For example, when creating a circular data structure, the cycle must be created after allocating all the elements. After that, however, the data structure doesn't need to be changed. Have you encountered situations like that? Do you think it would help if you could lock the object after all necessary changes were made?
18. Now, I'd like to move on to API design in general. Think of a recent API you designed.
    1. Did you make any conscious design or implementation decisions, to make the API easier or more manageable for these users?
    2. Are there any recurring issues / challenges that users have had with your API? How did you handle those?
    3. How do you differentiate between users of your API? Are there parts of the API that you expose some users but not others? How do you manage that?
    4. Did you make any conscious design or implementation decisions to protect key data or data structures from modification (inadvertent or malicious) from your users?

WRITE ONE INTERVIEW QUESTION

- Context: You are interested in building debugging tools for functional programming languages.

SURVEYS

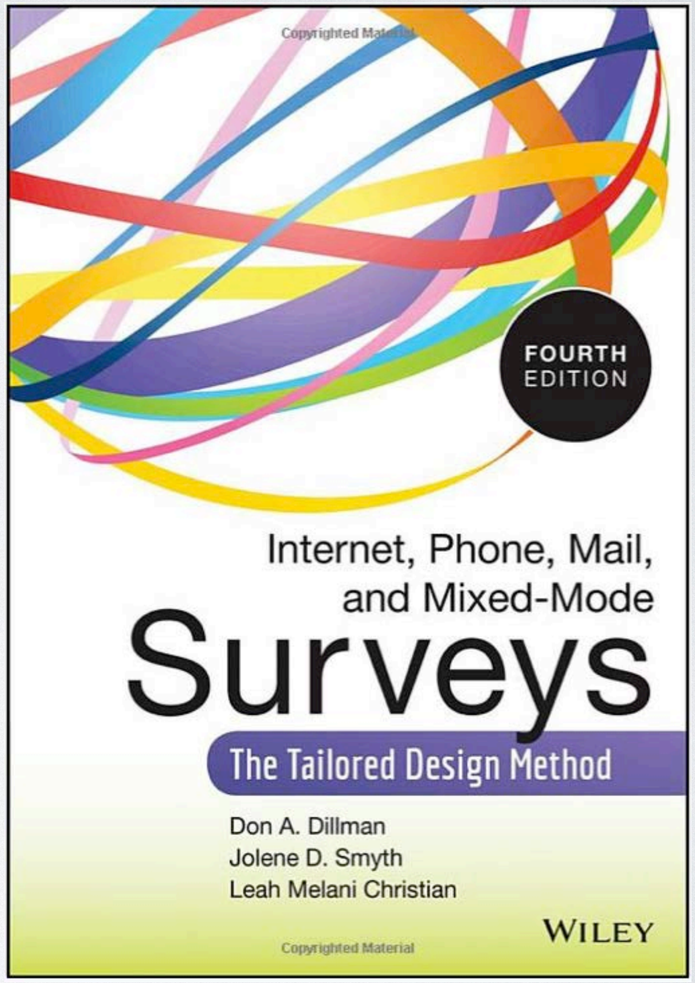
- Used to generalize findings
- Select a population; then figure out how to sample from it
  - Incentives?
- Don't need a lot of responses for a *qualitative* study

SURVEY DESIGN IS HARD!

- During how many days last week did you eat pasta?
- How many feet tall is your horse?
- "What businesses would you like to see in the Moscow area that are currently not available?" — average 1.8 answers
- "Are there any others?" — average 2.4 answers
- Not: "How about a Taco Bell?"

SURVEY DESIGN

- Hard to get surveys right. Run pilots!
- Get the Dillman, Smyth, Christian book. Lots of guidelines.



GROUNDED THEORY, BRIEFLY

- Goal: find themes and develop theories from qualitative data.
- Do not identify a hypothesis in advance.
- Instead, *observe* and learn.

GROUNDED THEORY, BRIEFLY

- Observe some phenomenon.
- Record events.
- "code" events. ("open coding")
- Establish relationships ("axial coding")

Research question: What irritates or upsets Millennials when receiving feedback on their work?

Open code	Properties	Examples of participants' words
Getting called out	Detesting verbal vomit and being ridiculed Feeling discouraged	Getting ripped apart Chewed out Bashed Chastised Criticized Thrown under the bus Negative tactics don't motivate us
Not being heard	Having work changed, which results in their voice not being heard Working so hard makes this frustrating Believing they don't have power to say anything	You slave away and they've completely changed what you've done My art was changed, which I worked really hard on People are always going to change what you do. Always! Co-worker presented my ideas as her own; no way to address those issues
Mind reading and expectations for a miracle worker	Believing they have a combination of vague instructions and specific expectations, some of which are unrealistic	Vague instructions Having to mind read Inadequate explanation I'm not a miracle worker

Tiffany Gallicano. https://prpost.wordpress.com/2013/07/22/an-example-of-how-to-perform-open-coding-axial-coding-and-selective-coding/

CONCLUSIONS ABOUT QUALITATIVE STUDIES

- Pilot, pilot, pilot. Revise after each one.
- When in doubt, narrow your research/design question.
- Phrasing your usability question *specifically* is a critical step
- Design tasks that identify the kinds of usability problems you are interested in
- Iterate to design good materials.

PART 6: QUANTITATIVE STUDIES

QUANTITATIVE STUDIES

- Key differences:
  - Don't interfere with or help participants
  - Don't change ANYTHING between trials
  - Do statistics on results

BASIC VOCABULARY

- Independent variables: things the experimenter chooses
  - Can assign participants to languages
  - Sometimes "explanatory variables"
- Dependent variables: things the experimenter measures
- Confounding variables: also affect dependent variables

EXAMPLE

- Are Obsidian users more likely to lose assets than Solidity users?
  - Independent variable: programming language used
  - Dependent variable: whether assets are lost
  - Confounding variable: programmer expertise

DEALING WITH CONFOUNDING VARIABLES

- Two options:
  - Control them
  - Record them

BACK TO EXAMPLE

	Solidity	**Obsidian**
Ran out of time	1	2
Lost an asset in either subtask	7	0
Subtask 1		
omitted refund of old bid	3	0
overwrote old refund	4	0
refunded to wrong bidder	0	1
Subtask 2		
overwrote old refund	4	0
refunded via `transfer()` instead of `pendingReturns`	4	N/A

Table 5. Errors in Auction task. N=10 in each condition.

Fisher's exact test: $p \approx .002$

"There is a 0.2% chance that the likelihood of asset loss is the same in the two conditions."

HYPOTHESIS TESTING

- Context: drawing from two populations.
- Question: what is the probability the two populations are the same?
- This is what *p-value* captures.
- Small p-value does not imply a large effect!

BETWEEN VS. WITHIN

- Between participants: each participant gets assigned to one condition
  - Avoids learning effects
- Within participants: each participant tries ALL conditions
  - Reduces inter-participant variance
- Obsidian study: we chose between-participants
  - Learning effects would be significant if we re-used tasks (and maybe even if not!)

SAMPLE SIZE

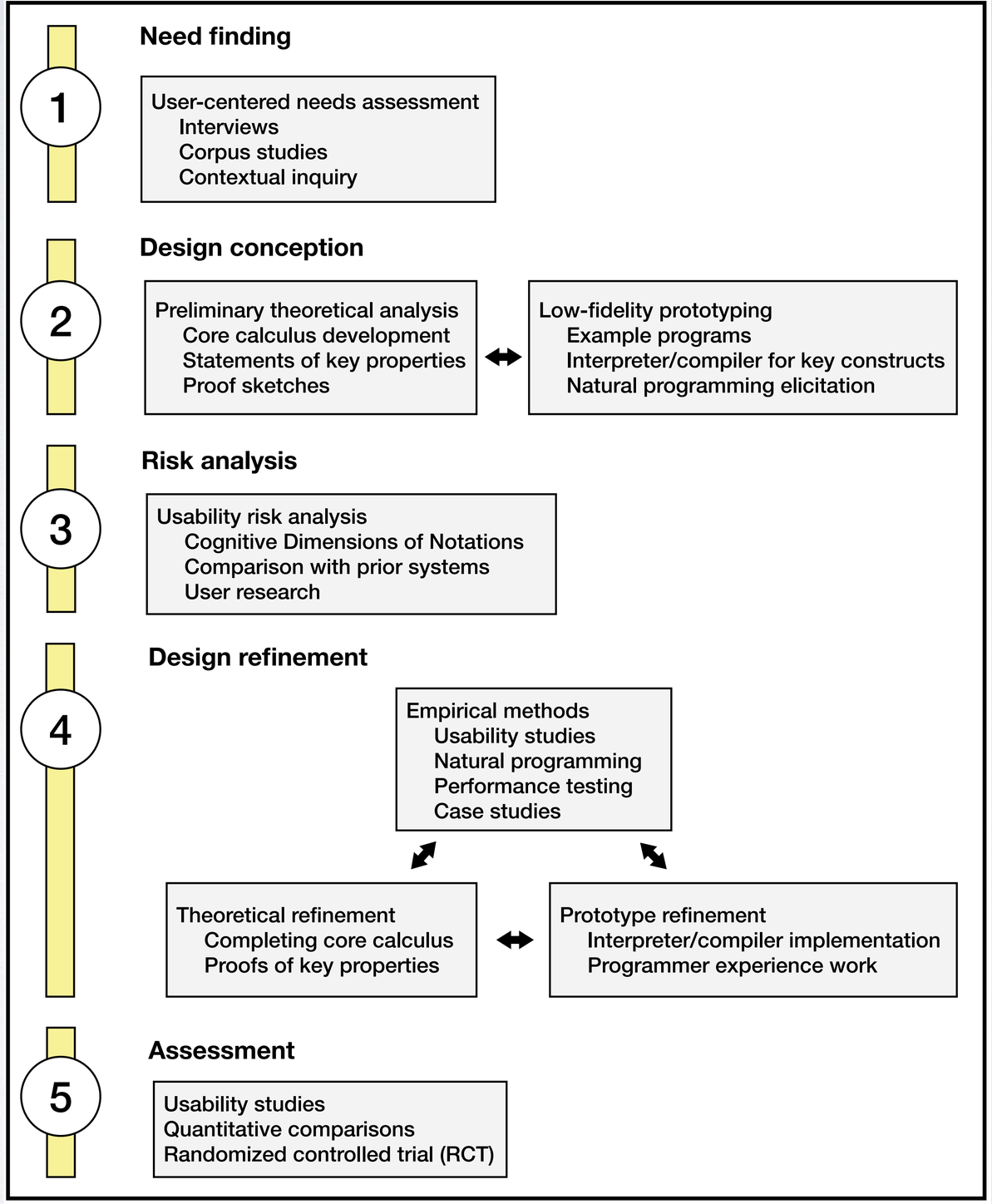
- How many participants?
  - Key questions: effect size and variance
  - Run a power analysis…but this requires estimating variance and effect size
  - Larger sample sizes allow detecting smaller effects
  - N=20 or N=30 are reasonable for large differences (half in each condition)
- But you can only generalize to your study population!

A TRADE-OFF

Low variance, low generality, small sample size	High variance, high generality, large sample size
• Short, highly-constrained tasks	• Long, open-ended tasks
• Narrow population (e.g., third-year students at University X)	• Broad population (e.g., all professional developers)

PART 7: WRAPPING UP

PLIERS: **P**rogramming **L**anguage **I**terative **E**valuation and **R**efinement **S**ystem



SOME QUESTIONS WE HAVE BEEN ABLE TO ADDRESS

- What parts of language A are confusing, despite the training we've developed?
- Are users of standard language A likely to insert bug B when doing task T?
  - Authenticity of the task matters! We try to base ours on real-world examples, or at least previous examples from the literature.
- Can users of language B complete task T about as fast as users of language A?
  - Sometimes yes, sometimes no
  - It's worth something to prevent bugs!

ORTHOGONALITY

- How do I know, when I combine these features together, results will be as advertised?
  - You don't — but if you don't get good results with one feature at a time, you won't with the combination.
  - Do summative studies to see.

GENERALITY

- "Your tasks are too small, you don't have enough participants, you got the wrong participants, you didn't train them enough…"
- But we have to start *somewhere*! Future studies will be more general.
- Matters less for formative studies: fixing a problem for someone likely fixes it for others, too

NOVICES VS. EXPERTS

- Does it do any good to address usability problems encountered by novices?
  - In many case, can fix them without hurting experts
  - No one becomes an expert without first being a novice
  - Patience is limited
  - Stockholm syndrome

CONCLUSION AND DISCUSSION

- Qualitative and quantitative usability methods are useful for programming language design because PLs are interfaces for people
- Qualitative studies can:
  - Help make major design decisions, e.g. by suggesting new parts of the design space
  - Identify latent problems in your design, so you can fix them
- Quantitative studies can show the benefits of your design relative to the status quo