With thanks to Prof Thomas LaToza …

Full slides at http://tinyurl.com/LaTozaTutorial

Evaluating Programming Languages and Tools in Studies with Human Participants

Track SPLASH 2015 Tutorials

When **Wed 28 Oct 2015 10:30 - 12:00 at Edenburg** - Tutorial 2

Abstract Programming languages and tools exist to enable software developers to program. How effectively they do so ultimately depends on the interaction between languages and tools and the developers who use them. As new language features and tools are designed, a fundamental and inherently empirical question raised is, do they help programmers work better. Answering such questions requires conducting studies with human participants. This tutorial will provide a broad overview of methods for evaluating programming languages and tools in studies with human participants. The tutorial will be aimed at SPLASH attendees that have never before conducted a human subjects study, helping introduce attendees to the basics of designing and con- ducting studies and methods for the analysis of data. Elements of a study design will be surveyed, including recruitment and selection of human participants, informed consent, experimental procedures, demographic measurements, group assignment, training, the selection and design of tasks, the measurement of common outcome variables, and study debriefing. Broader elements of the research process will also be surveyed, including finding and refining research questions for studies, techniques and models for analysing empirical data from human participants, and finding the right balance between quantitative and qualitative methods.

File attachments

Slides (SPLASH15 Experiments Tutorial.pdf)	1.50MiB

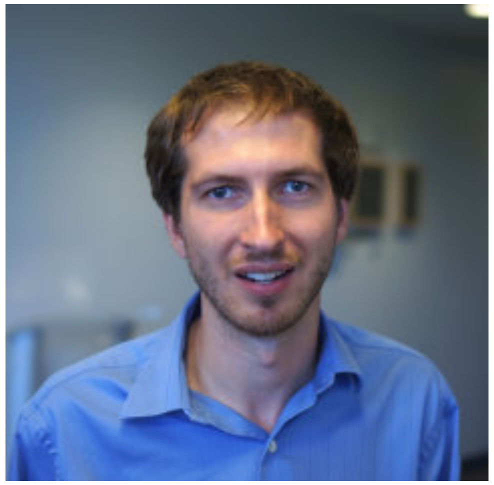
Bio Thomas LaToza is an Assistant Professor of Computer Science at George Mason University. He has degrees in psychology and computer science from the University of Illinois and a PhD in software engineering from Carnegie Mellon University. His research is in the area of human aspects of software development, encompassing empirical and design work on environments for programming, software design, and collaboration. He has been active in bringing human subjects studies to the investigation of software development activity and the evaluation of software development tools and has conducted over 20 studies with software developers, including observational studies, surveys, interviews, field deployments, and controlled experiments. He has served on various program committees and is currently the co-chair of the Sixth Workshop on the Evaluation and Usability of Programming Languages and Tools.

Session Program

Wed 28 Oct

10:30 - 12:00: Tutorials - Tutorial 2 at Edenburg

10:30 - 12:00 *Talk* **Evaluating Programming Languages and Tools in Studies with Human Participants** *Thomas LaToza*



Thomas LaToza

George Mason University

Evaluating Research, and Studies with Human Participants

Andrew P. Black
based on material by Thomas LaToza

Motivation

- Evaluate the **usability** of a feature or tool to its users
  - usually productivity effects
  - perhaps security, correctness ...
- Given a *context*, what is effect of your tool or technique on its *intended audience*

Issues for Studies with Human Subjects

- How many **participants** do I need?
- Is it ok to use students?
- What do I **measure**? How do I measure it?
- What's an IRB?
- Should I train participants?
- What **tasks** should I pick?

Empirical Software Engineering
February 2015, Volume 20, Issue 1, pp 110-141

A Practical Guide to Controlled Experiments Evaluating Software Engineering Tools with Human Participants

Andrew J. Ko, University of Washington, *ajko@uw.edu*

Thomas D. LaToza, UC Irvine, *tlatoza@ics.uci.edu*

Margaret M. Burnett, Oregon State University, *burnett@eecs.oregonstate.edu*

Abstract Empirical studies, often in the form of controlled experiments, have been widely adopted in software engineering research as a way to evaluate the merits of new software engineering tools. However, controlled experiments involving *human* participants *using* new tools remain rare. When they are conducted, some have serious validity concerns. Recent research has also shown that many software engineering researchers view this form of tool evaluation as too risky and difficult to conduct, as it might ultimately lead to inconclusive or negative results. In this paper, we aim to help researchers design studies that minimize these risks and increase the quality of controlled experiments with developers by offering practical methodological guidance. We explain, from a practical perspective, options in the recruitment and selection of human participants, informed consent, experimental procedures, demographic measurements, group assignment, training, the selection and design of tasks, the measurement of common outcome variables such as success and time on task, and study debriefing. Throughout, we situate this guidance in the results of a new systematic review of the 345 tool evaluations with human participants that were reported in over 1,700 software engineering papers published from 2001-2011.

Keywords *Research methodology, tools, human participants, human subjects, experiments.*

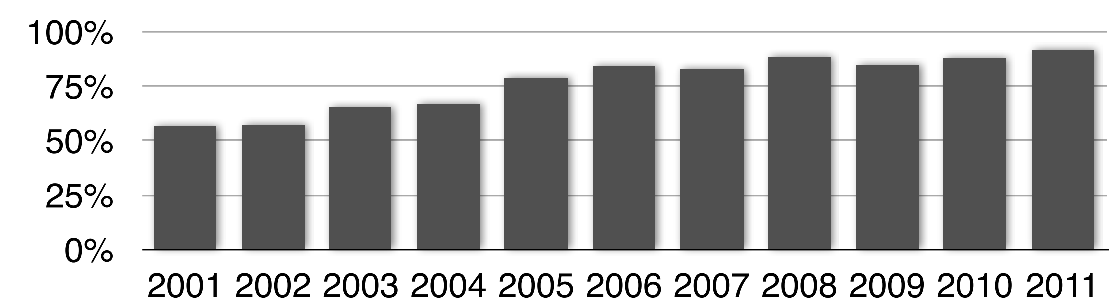
1. Introduction

Over the past three decades, empirical studies have become widely accepted as a way to evaluate the strengths and weaknesses of software engineering tools (Zannier et al. 2006, Basili et al. 1986, Basili 1993, Basili 2007, Rombach et al. 1992, Fenton 1993, Tichy et al. 1995, Basili

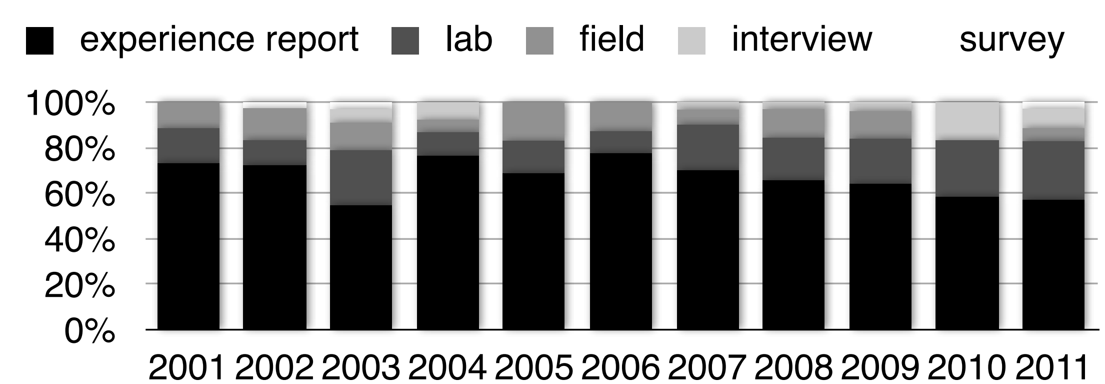
Data on how software engineering community conducts experiments w/ humans

- Systematic review of 1701 software engineering articles
  - All papers published at ICSE, FSE, TSE, TOSEM 2001 - 2011

82%	63%	17%
1392	1065	289
described tool	empirical eval	empirical eval w/ humans



% of papers w/ tools w/ empirical evals

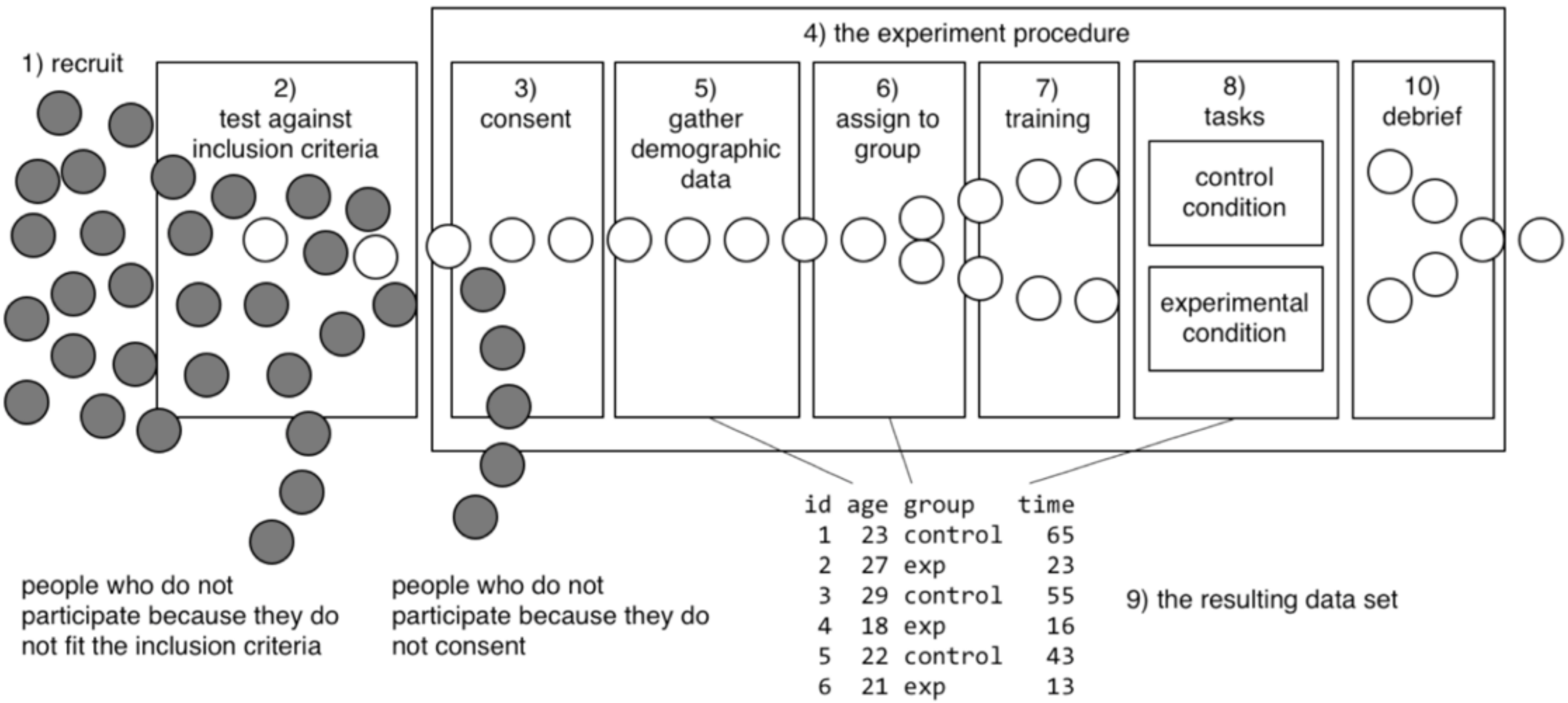


% of human evals by method

Controlled experiment

- Only way to argue **causality**: change in var x causes change in var y
- Manipulate **independent** variables
  - Creates “conditions” that are being compared
  - Can have >1, but number of conditions usually exponential in number of independent variables
- Measure **dependent** variables (a.k.a “measures”)
  - Quantitative variable you calculate from collected data
    - e.g., time, nr of questions, nr of steps
- **Randomly** assign participants to condition
  - Ensure that participants differ *only* in condition
  - *Not* different in other **confounding** variables
- Test hypotheses
  - Change in independent variable causes dependent variable to change
  - e.g., t-test, ANOVA, other statistical techniques

Anatomy of controlled experiment w/ humans

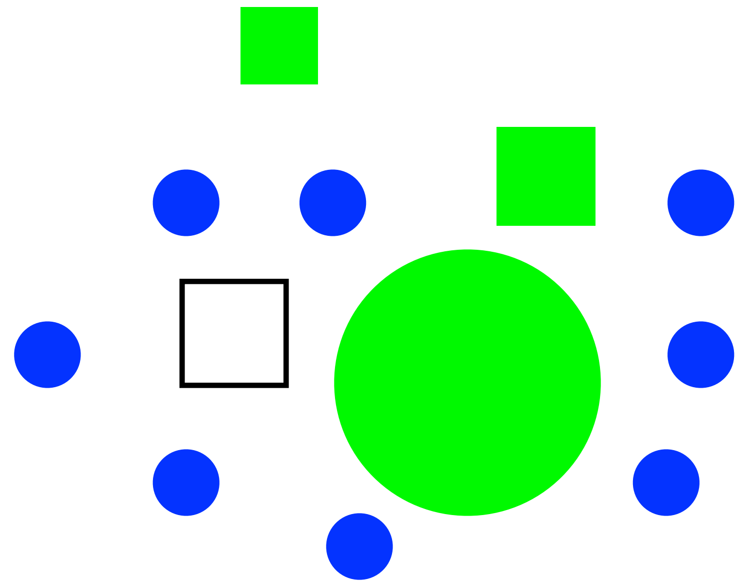


Terminology

- “Tool” — any **intervention** manipulating a subject’s work environment
  - e.g., in software engineering: programming language, language feature, software development environment feature, build system tool, API design, documentation technique
- Data — what you collected in study
- Unit of analysis — individual **item** of data
- Population — **all** members that exist
- Construct — some **property** of member
- Measure — **approximation** of construct computed from data

Example — Study of shapes

Real world

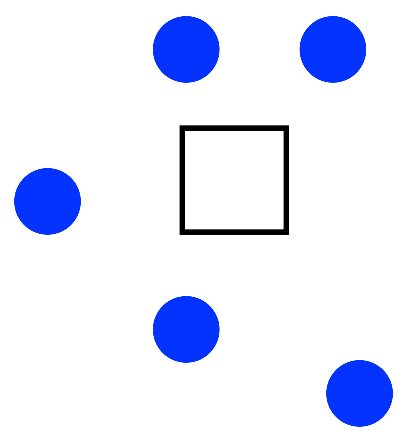


Population

shape
size
filled / empty
color

Constructs

Study



Sample of population

is blue

size < 10

Measure

(Some) types of validity

- **Validity** = should you believe a result
- **Construct** validity
  - Does measure correspond to construct, or something else?
- **External** validity
  - Do results generalize from participants to population?
- **Internal** validity (experiments only)
  - Are the differences between conditions caused only by experimental manipulation and not other variables? (confounds)

Example: Typed vs. untyped languages

S. Hanenberg. (2009). What is the impact of static type systems on programming time? In the *PLATEAU workshop, OOPSLA 09*.

Participants 26 undergrads **Task** write a parser 27 hrs

Setup new OO language 16 hr instructions

Conditions type system vs. no type system

found errors at compile time errors detected at runtime

RESULTS

Developers with untyped version significantly faster completing task to same quality level (unit tests).

Example: Study validity

- **Construct** validity
  - Does measure correspond to construct or something else?
- **External** validity
  - Do results generalize from participants to population?
- **Internal** validity (experiments only)
  - Are the differences between conditions caused only by experimental manipulation and not other variables? (confounds)
- **Other** reasons you're skeptical about results?

Good (not perfect) study designs

- Goals

  Maximize **validity** — often requires more

  participants, data collected, measures

  longer tasks

  more realistic conditions

  Minimize **cost** — often requires

  fewer participants, data collected, measures

  shorter tasks

  less realistic, easier to replicate conditions

- Studies are **not proofs**: results could always be invalid

  don't sample all developers, or tasks, or situations

  measures imperfect

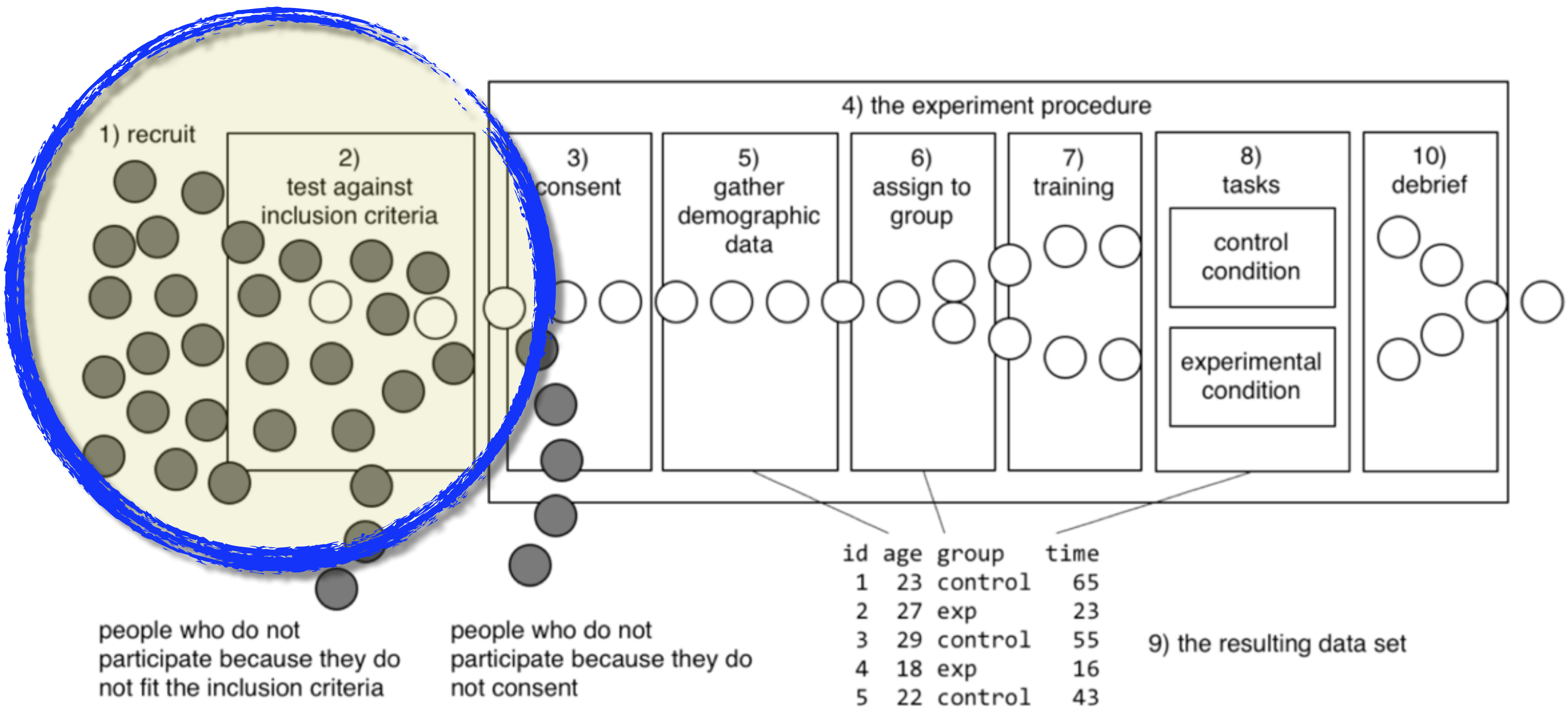
- Goal is to find results that are

  interesting

  relevant to research questions

  valid *enough* your target audience believes them

Overview



1) recruit
2) test against inclusion criteria
4) the experiment procedure
3) consent
5) gather demographic data
6) assign to group
7) training
8) tasks
control condition
experimental condition
10) debrief
id age group time
1 23 control 65
2 27 exp 23
3 29 control 55
4 18 exp 16
5 22 control 43
6 21 exp 13
9) the resulting data set
people who do not participate because they do not fit the inclusion criteria
people who do not participate because they do not consent

Deciding who to recruit

- **Inclusion criterion**: attributes participants *must* have to be included in study
- Goal: reflect characteristics of those that researchers believe would benefit
- Example: Nimmer & Ernst (2002) “Invariant inference for static checking: An empirical evaluation”
  - Support those without experience of similar inference tools
  - Chose graduate students
  - Developed items to assess
    (1) no familiarity with tool, (2) experience with Java
    (3) experience writing code

Common inclusion criteria for Software Studies

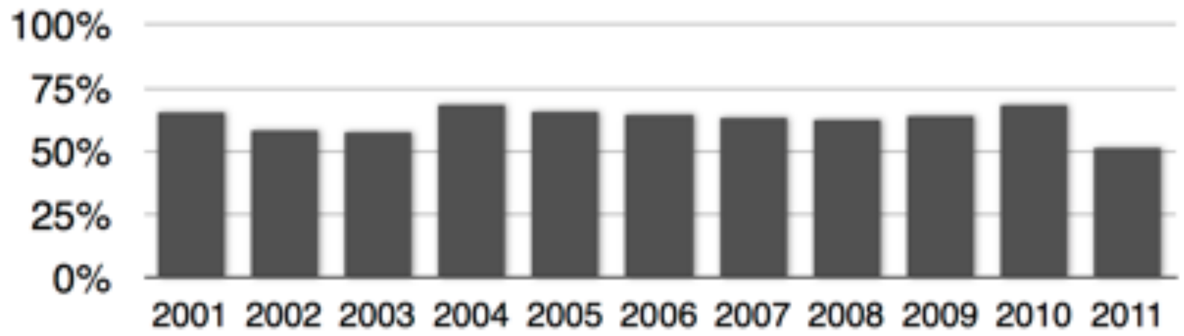
- Experience w/ a programming language
  - Self-estimation of **expertise**; time
- Experience w/ related **technologies**
  - Important for learning new tool
- **Industry experience**
  - Indicator of skills & knowledge; could also ask directly
- (Natural) language proficiency

Poor criteria: Paper authors

- **62%** of studies evaluating a tool involved **tool's authors** using the tool & reporting personal experiences

- Tool authors far more likely to use own tool successfully than those new to tool

- Tool authors more likely to overlook weaknesses of tool



Proportion of evaluations involving humans in which authors were study participants

What about using students?

- **72%** of 113 SE experiments 1993–2002 used students [Sjoberg 2005]
- 23% reported using students in studies 2001–2011 (many did not report if, or if not)
- Students can be too inexperienced to be representative of tools intended users; observer-expectancy effect
- But
  - depends on task & necessary expertise
  - professional masters students may have industry experience
  - can minimize observer-expectancy effect

How many participants?

- More participants ⇒ more statistical power
  - higher chance to observe **actual** differences
  - **power analysis** — given assumptions about expected effect size and variation, compute participants number
- Experiments recruited median **36** participants, median **18** per condition
  - Some studies smaller

Recruiting participants

- Marketing problem: how to attract participants who meet inclusion criteria

- Questions:

    - Where do such participants pay **attention**?

    - What **incentives** to offer for participation?

Sources of participants

- Students
  - Class announcement, fliers, emailing lists
  - Incentives: small compensation & intrinsic interest
- Software professionals
  - Relationships w/ industry researchers
  - Studies by **interns** at companies
  - **Partnerships** or contracts with companies
  - **In-house** university software teams
  - **Meetup** developer groups, public mailing lists, FB groups
  - CS Alumni mailing lists, LinkedIn groups

Remote participants

- Online labor markets focused on, or including, developers (e.g., MTurk, oDesk, TopCoder)
- Pros
  - Can quickly recruit hundreds or **thousands** of participants
  - Use their own space & tools; work at own time
- Cons
  - May **misreport** levels of experience
  - Might leave task temporarily; more extraneous variation

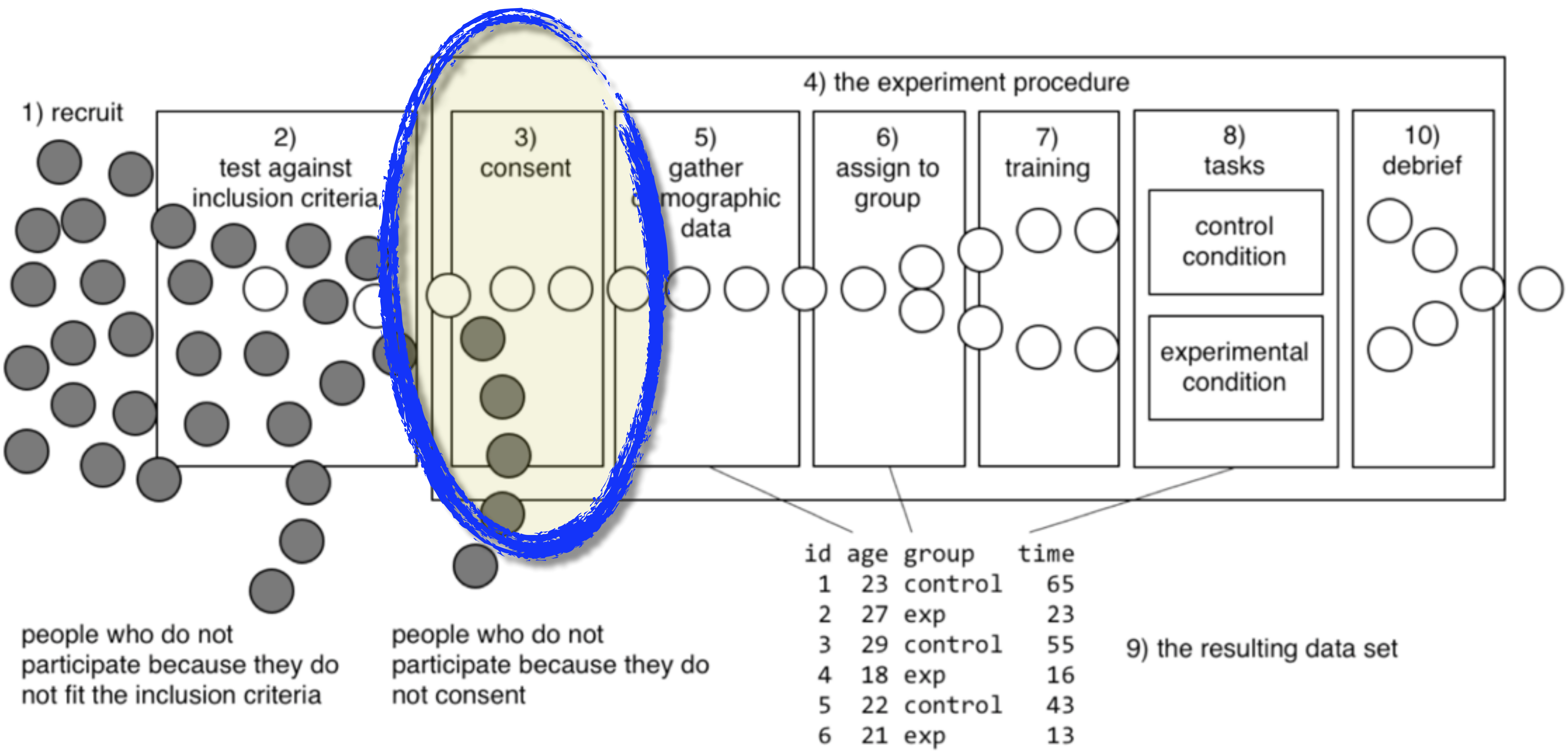
Remote participants: MTurk example

- Recruited participants from MTurk across 96 hours
- Used **qualification test** to screen for programming expertise
  - multiple choice question about program output
- Paid $5 for <= 30 mins

Participant numbers:

4776	3699	999	777	489
completed informed consent	took qualification test	qualified	completed 1 task	completed all tasks

Overview



1) recruit
2) test against inclusion criteria
3) consent
4) the experiment procedure
5) gather
mographic
data
6) assign to group
7) training
8) tasks
control condition
experimental condition
10) debrief
people who do not participate because they do not fit the inclusion criteria
people who do not participate because they do not consent
id age group time
1 23 control 65
2 27 exp 23
3 29 control 55
4 18 exp 16
5 22 control 43
6 21 exp 13
9) the resulting data set

Informed consent

- Enables participants to **decide** to participate given a short document
- Key elements
  - Names & contact info for you and other experimenters
  - **Purpose** of the study
  - Brief (one or two sentence) high-level description of the types of work participants will be asked to do
  - Expected **length** of the study
  - A statement of any possible **benefits** or compensation
  - A statement of any possible **risks** or discomforts
  - Overview of the data you will collect (think-aloud, screencast, survey questions, etc.)
  - Clear statement on **confidentiality** of data (who will have access?)

UNIVERSITY OF CALIFORNIA, IRVINE
CONSENT TO ACT AS A HUMAN RESEARCH SUBJECT

Crowd Programming

You are being asked to participate in a research study. Participation is completely voluntary. Please read the information below and ask questions about anything that you do not understand. A researcher listed below will be available to answer your questions.

RESEARCH TEAM
Lead Researcher
Dr. Thomas LaToza
Department of Informatics
tlatoza@uci.edu

Faculty Sponsor
Prof. André van der Hoek
Department of Informatics
andre@ics.uci.edu

Other Researchers
Lee Martie
Christian Adriano
Micky Chen
Luxi Jiang

STUDY LOCATIONS
Your own workspace

STUDY SPONSOR
National Science Foundation

WHY IS THIS RESEARCH STUDY BEING DONE?
The purpose of this research study is to examine how design competitions might be used in crowdsourcing software and user interface design.

HOW MANY PEOPLE WILL TAKE PART IN THIS STUDY?
This study will enroll approximately 40 participants. All study procedures will be conducted in your own workspace.

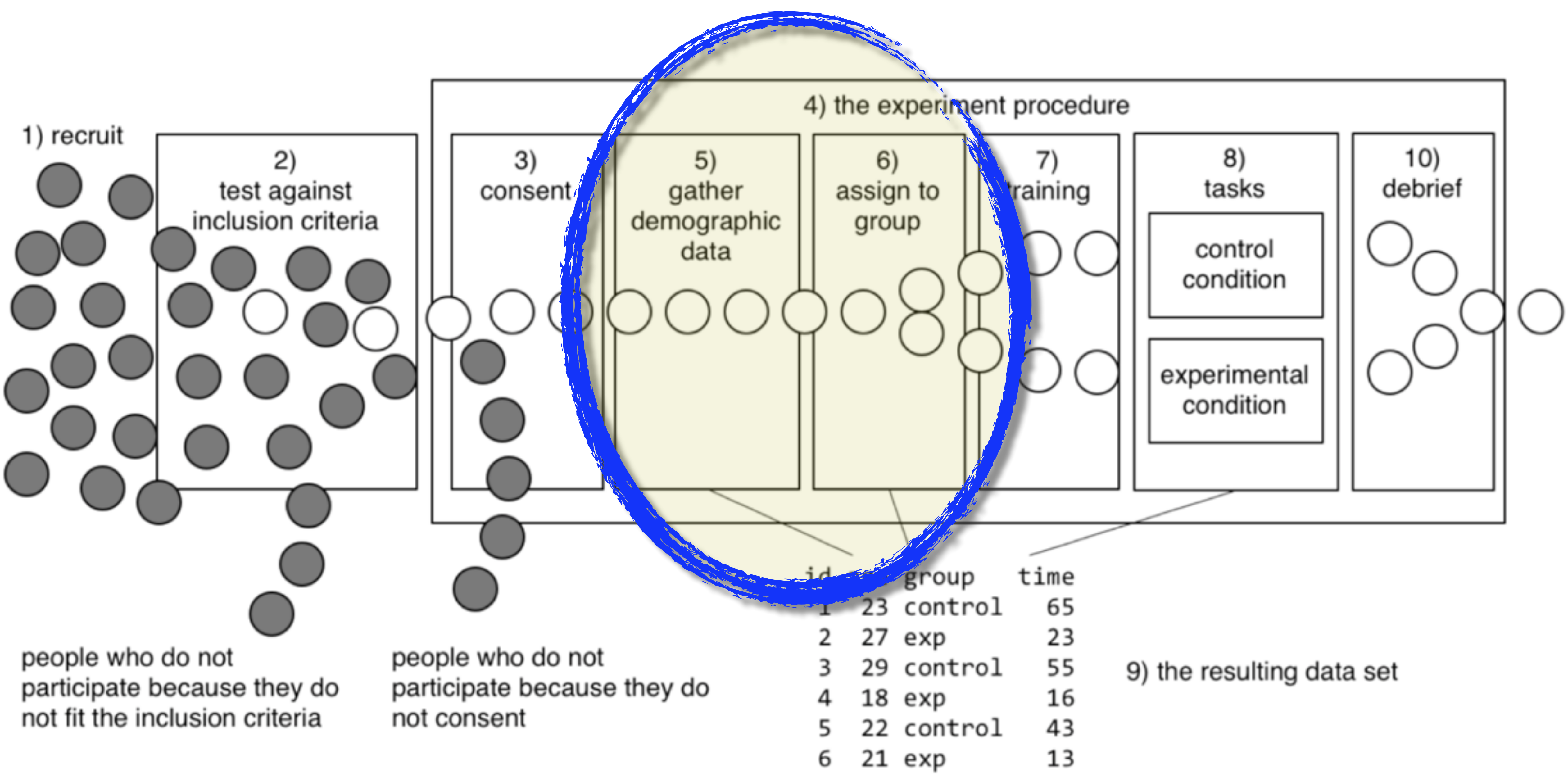
WHAT PROCEDURES ARE INVOLVED WITH THIS STUDY AND HOW LONG WILL THEY TAKE?

1. You are being asked to participate in a design competition. In the first one-week period, you'll be given instructions for a design task and be asked to submit a design. After submitting your design,

IRB Approval

- US universities have an **Institutional Review Board** (IRB) responsible for ensuring human subjects treated ethically
- Before conducting a study with human subjects:
  - Must complete human subjects **training** (first time only)
  - Submit an application to IRB for **approval** (2–??? week approval time)
- During a study:
  - Must administer "**informed consent**" describing procedures of study and any risks to participants

Overview


4) the experiment procedure
1) recruit
2) test against inclusion criteria
3) consent
5) gather demographic data
6) assign to group
7) training
8) tasks
control condition
experimental condition
10) debrief
people who do not participate because they do not fit the inclusion criteria
people who do not participate because they do not consent
group time
1 23 control 65
2 27 exp 23
3 29 control 55
4 18 exp 16
5 22 control 43
6 21 exp 13
9) the resulting data set

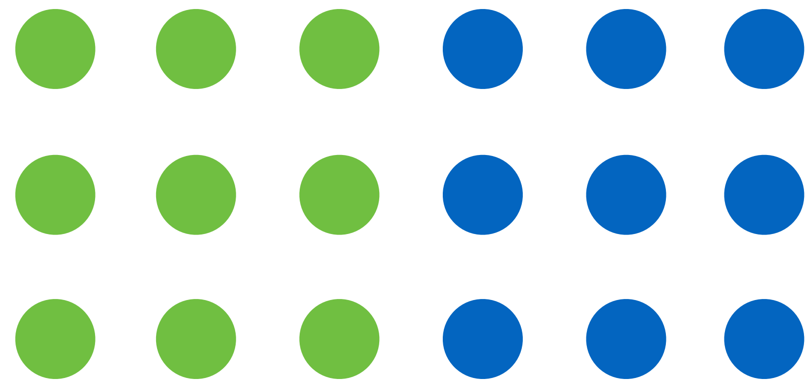
Collecting demographic data

- Goal: understand expertise, background, tool experience, …
- **Interviews** — potentially more comfortable, informative
  - Before or after tasks
- **Surveys** — more consistent, can be used to test against inclusion criteria during recruiting

Assigning participants to an experimental condition

- Random assignment
  - distributes random **variation** in participant skills and behavior across all conditions
  - minimizes chance that observed difference is due to participant differences
- Used with a **between-subjects** experiment
  - (each participant is subjected to just one condition)
- Alternative designs can reduce number of participants necessary to recruit

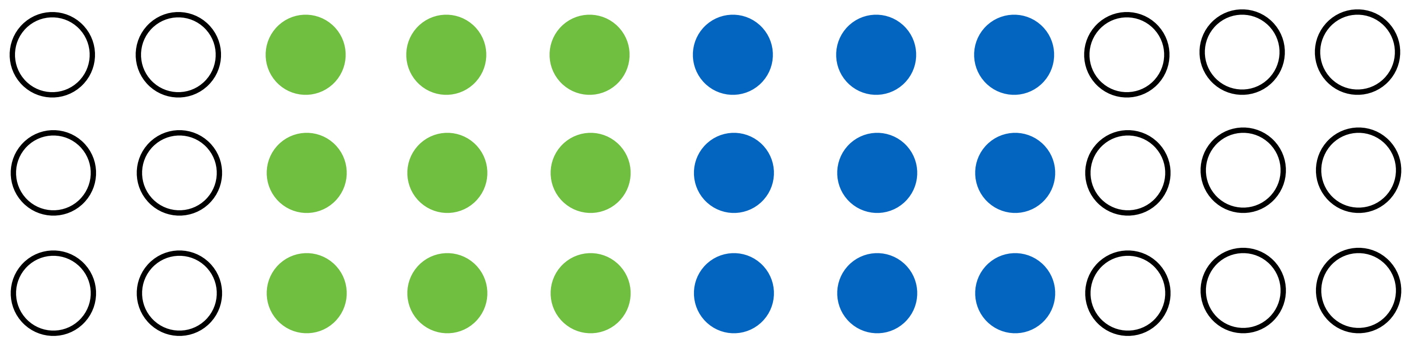
Within-subjects design



- All participants use all tools being compared, one at a time, across several tasks
    - e.g., participant uses tool in task 1 but not task 2
- **Learning effect** — doing first task may increase performance on second task
- —> **Counterbalancing** — randomize order of tasks, & on which task, participants use each tool
    - Latin Square design

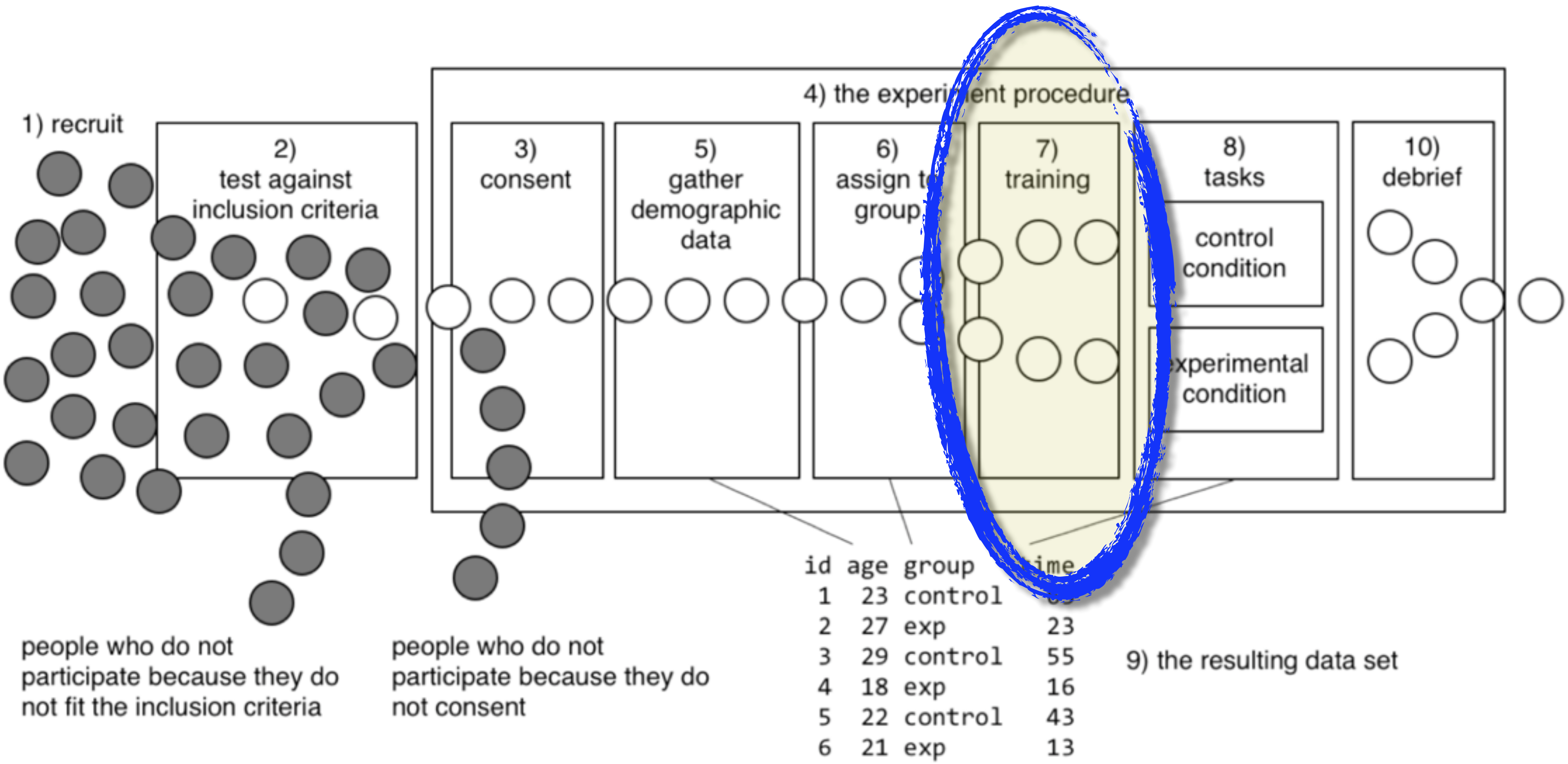
no two rows or columns the same

Interrupted time-series design



- Measure outcome variable **before** tool introduced, **after** introduced, **after removing** tool
- Can see possible causal effects of tool
- Enables participants to articulate effects of tool
- Could be "trial run" of new tool in a field deployment of tool to a company

Overview


4) the experiment procedure
1) recruit
2) test against inclusion criteria
3) consent
5) gather demographic data
6) assign to group
7) training
8) tasks
control condition
experimental condition
10) debrief
id age group time
1 23 control
2 27 exp 23
3 29 control 55
4 18 exp 16
5 22 control 43
6 21 exp 13
people who do not participate because they do not fit the inclusion criteria
people who do not participate because they do not consent
9) the resulting data set

Training participants

- Participants need to know:
  - how to use **tools** in the given environment
  - terminology & domain **knowledge** used in task
  - design of programs they will work with during task
- Can provide background and **tutorial** materials to ensure participants have required knowledge.

To train or not to train?

- This is a key question. Training changes assumptions about context to which the results may apply

- Training
  - Ensures participants are **proficient** and **focused** on the task

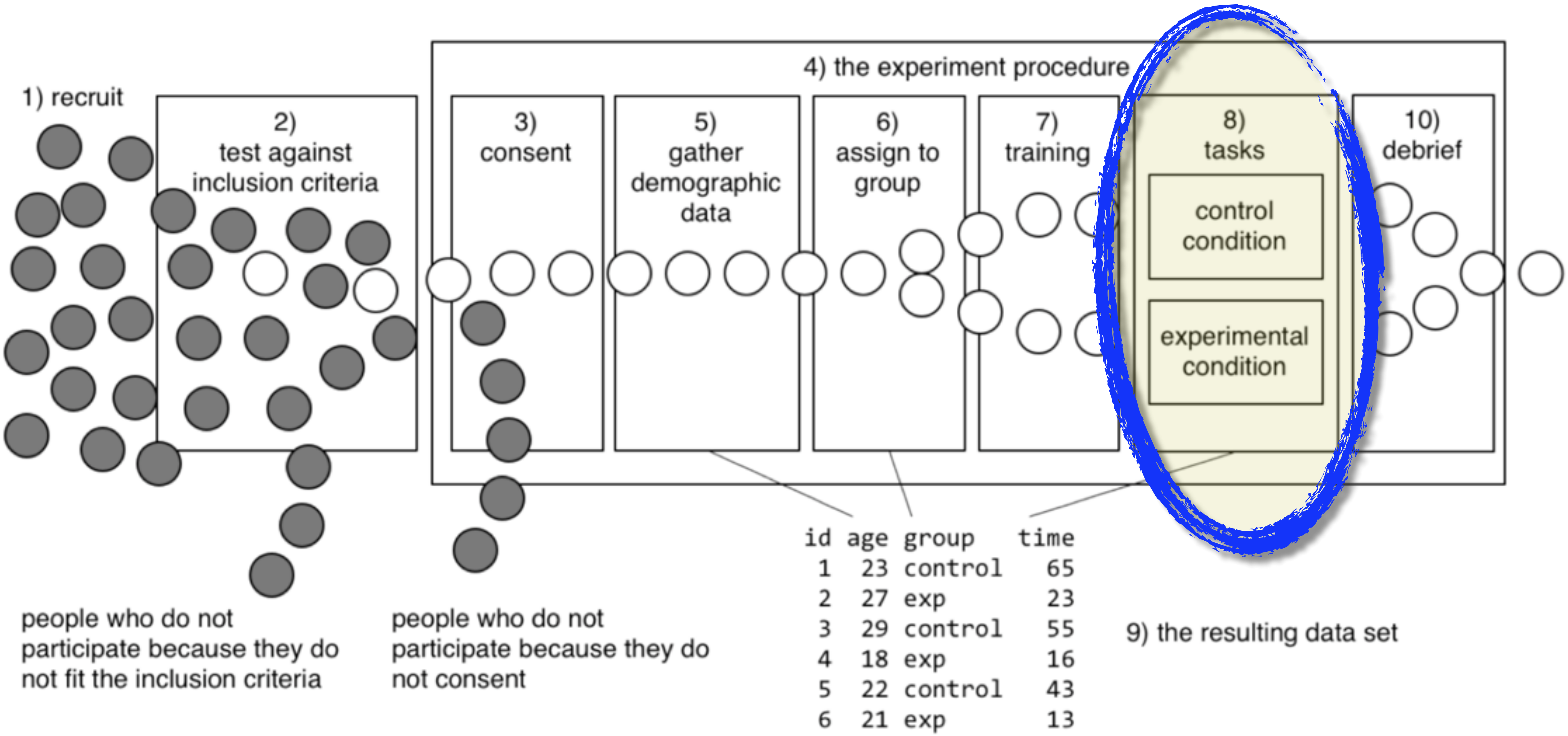
- No training
  - Results generalize to new (untrained) users, but risks study being dominated by learning

- Software studies often choose to provide training materials for tool

Design of training materials

- Goal: **teach** required concepts quickly & effectively
- Possible approaches
  - Background materials
  - Video instructions
  - Tutorial where participants complete example task w/ tool
  - Cheat sheets
- Can also include **assessment** to ensure learning
- Can be helpful for experimenter to answer participant questions

Overview


1) recruit
2) test against inclusion criteria
4) the experiment procedure
3) consent
5) gather demographic data
6) assign to group
7) training
8) tasks
control condition
experimental condition
10) debrief
people who do not participate because they do not fit the inclusion criteria
people who do not participate because they do not consent
id age group time
1 23 control 65
2 27 exp 23
3 29 control 55
4 18 exp 16
5 22 control 43
6 21 exp 13
9) the resulting data set

Tasks

- Goal: design tasks that have **coverage** of work affected by tool
- Key tradeoff: realism vs. control
  - How are real, messy programming tasks **distilled** into brief, accessible, actionable activities?
- More realism ⇒ messier, fewer controls
- More control ⇒ cleaner, less realism
- Tradeoff often takes the form of tradeoff between bigger tasks vs. smaller tasks

Feature coverage

- Of all functionality and features of tool, which will receive **focus** in tasks?
- More features ⇒ more to learn, more variation in performance, higher risk of undue negative results
- Fewer features ⇒ less to learn, less ecological validity, more likely to observe differences

Experimental setting

- Experiments can be conducted in lab or in developer's actual workspace
- Experiments most often conducted in **lab** (86%)
    - Enables **control** over environment
    - Can minimize distractions
    - But: less realism, as may have different computer, software, … from participants' normal setting

Task origin

- **Found** task — task from real project (15%)
  - e.g., bug fix task from an OSS project
  - More **ecologically** valid
  - May not exist for new tools
  - Can be hard to determine what feature usage found task will lead to
- **Synthetic** task — designed task (85%)
  - Can be easier to tailor for effective feature **coverage**
  - Must compare synthetic task to real tasks

Task duration

- **Unlimited** time to work on a task
    - Allow either participant or experimenter to determine when task is complete
    - Hard to find participants willing to work for longer time periods
- **Fixed** time limit
    - More **control** over how participants allocate time across tasks
    - Can introduce **floor effect** in time measures, where no one can complete task in time
- Typical length of **1–2** hours

Measuring outcomes

- Wide range of possible measures
  - Task completion, time on task, mistakes
  - Failure detection, search effort
  - Accuracy, precision, correctness, quality
  - Program comprehension, confidence
- Most frequent: **success** on task, **time** on task, tool **usefulness**

Determining when goal is reached

- Experimenter **watches** participant for success
    - Requires **consistency**, which can be challenging
- Success is **automatically** measured (e.g., unit tests)
    - Requires researcher to identify all goal states in advance, which can be challenging
- **Participants** determine they believe they have succeeded
    - Most ecologically valid
    - Introduces variation, as participants may vary in confidence they obtain before reporting they are done

Defining success to participants

- Need to **unambiguously** communicate goal to participants
- When participants themselves determine, may ask experimenter about what is success
  - Experimenter can reiterate **instructions** from beginning
- When experimenter determines
  - Experimenter should respond “I am unable to answer that question”

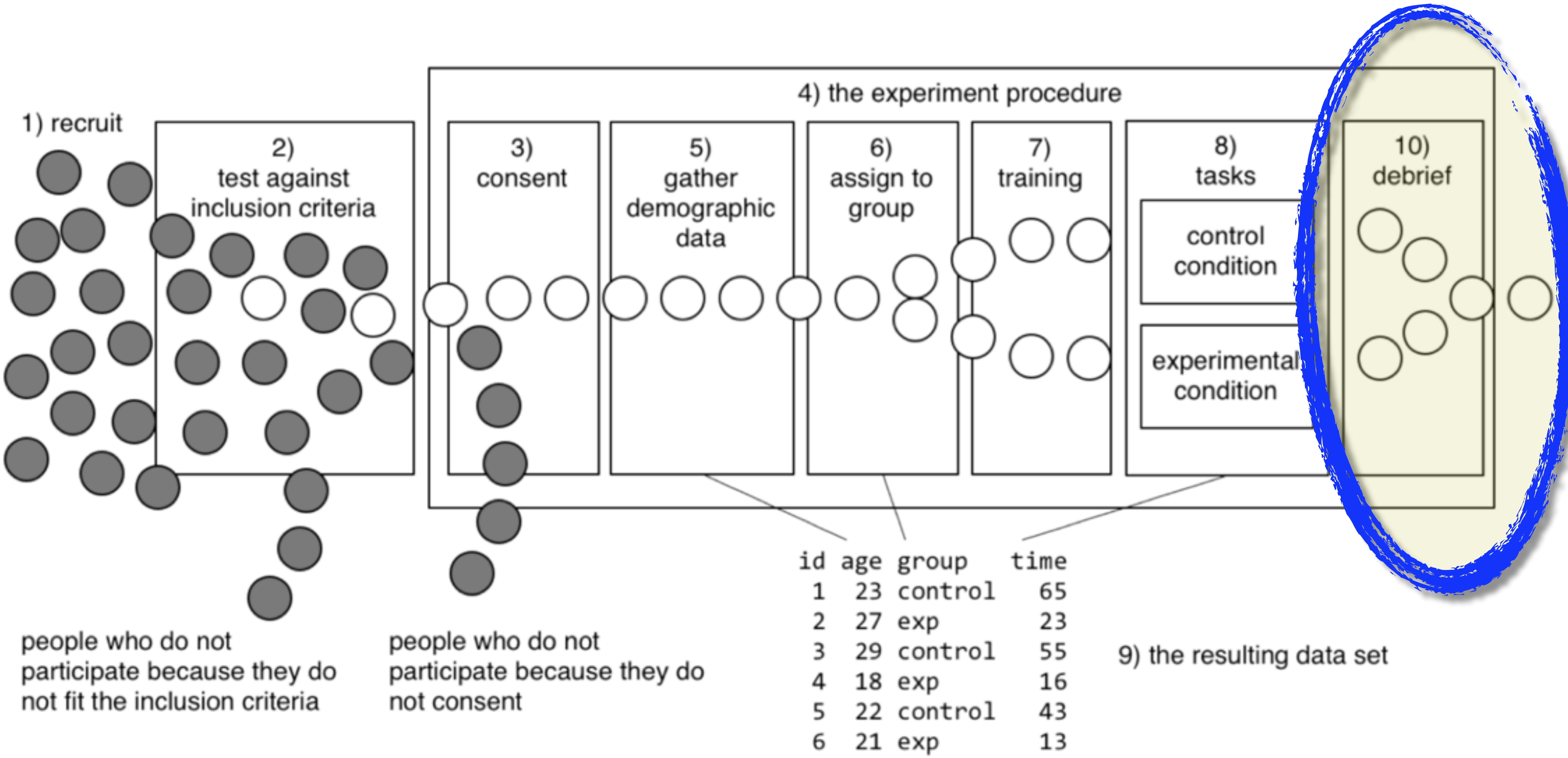
Measuring time on task

- Need to define task **start,** task **end,** and who determines when task has finished
- What is **start?**
  - When participant starts reading task — includes variation in time spent reading
  - When participants starts working
- What is **end?**
  - What happens if participant succeeds but does not realize it?
  - What happens if they think they succeeded, but failed?

Measuring usefulness

- Usefulness — does the tool provide functionality that satisfies a **user need** or provides a benefit?
  - *Not* "usability" — ease of use for task
- Might **ask** subject
  - Did they find the tool useful?
  - Would they consider using it in the future?
- Technology Acceptance Model
  - **Validated** instrument for measuring usefulness through a questionnaire

Overview



4) the experiment procedure
1) recruit
2) test against inclusion criteria
3) consent
5) gather demographic data
6) assign to group
7) training
8) tasks
control condition
experimental condition
10) debrief
id age group time
1 23 control 65
2 27 exp 23
3 29 control 55
4 18 exp 16
5 22 control 43
6 21 exp 13
9) the resulting data set
people who do not participate because they do not fit the inclusion criteria
people who do not participate because they do not consent

Debriefing & compensation

- Explain to participant what study **investigated**
- Explain the correct solutions to tasks
- Instructions about information that should not be shared with others
  - e.g., don't share tasks with friends who might participate
- Get speculative **feedback** about tool
  - Can use semi-structured interview to get perceptions of tool

Piloting

Most important step in ensuring useful results!

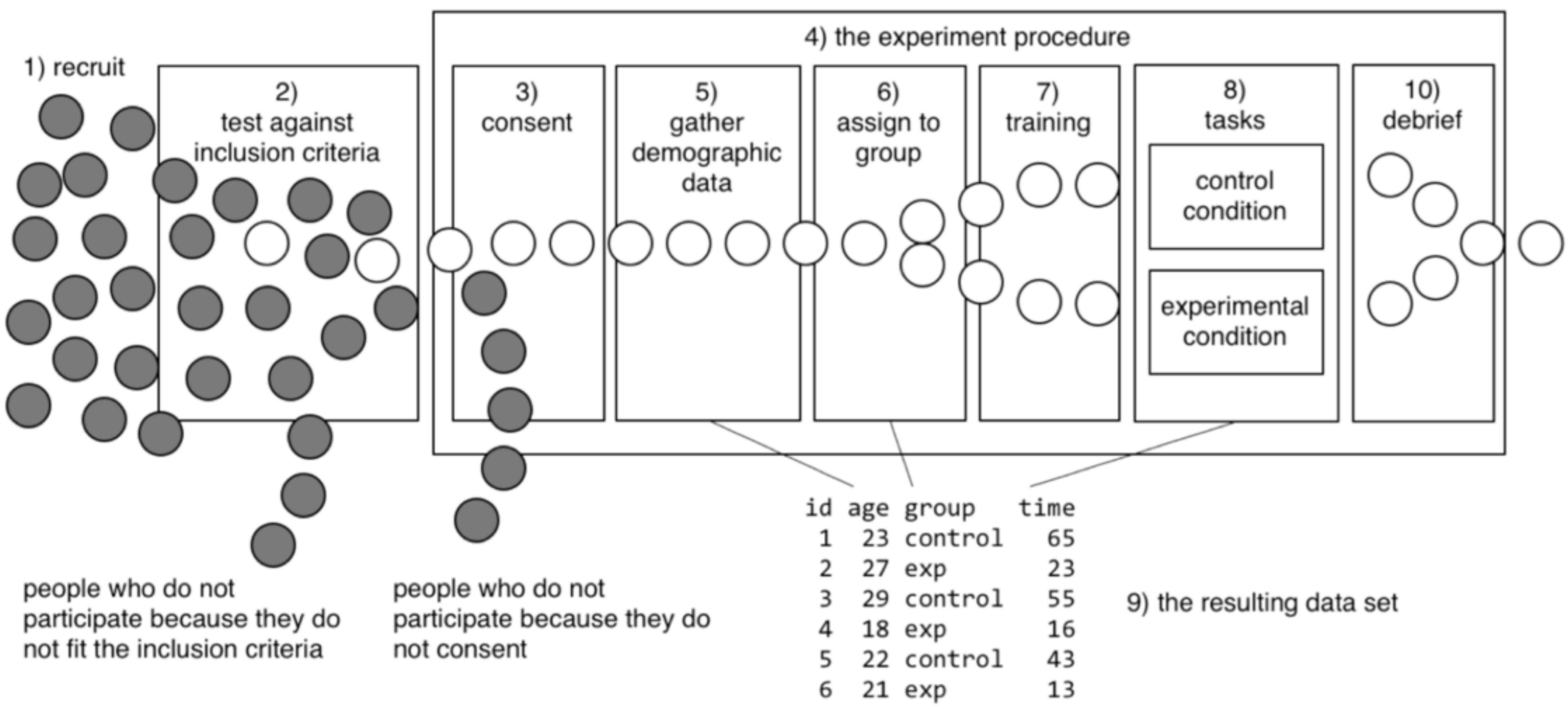
(1) Run study on **small** (1–4) number of participants

(2) Fix **problems** with *study design*:
Was the tool tutorial sufficient?
Did tasks use your tool? Enough?
Did subjects understand your materials?
Did you collect the right data?
Are your measures correct?

(3) Fix **usability** problems
Are developers doing the “real” task, or messing with tool?
Are users confused by terminology in tool?
Do supported commands match commands users expect?

(4) **Repeat** 1, 2, and 3 until no more (serious) problems

Done!


4) the experiment procedure
1) recruit
2) test against inclusion criteria
3) consent
5) gather demographic data
6) assign to group
7) training
8) tasks
control condition
experimental condition
10) debrief
id age group time
1 23 control 65
2 27 exp 23
3 29 control 55
4 18 exp 16
5 22 control 43
6 21 exp 13
people who do not participate because they do not fit the inclusion criteria
people who do not participate because they do not consent
9) the resulting data set

Qualitative data

On the value of qualitative data

- Experiment may provide evidence that A is “better” than B
- But always generalizability questions about **why** and **when**
- Qualitative data offers possibility of *explanation*: why result occurred.
- Can use **coding** to convert qualitative data to categorical data, which can be counted or associated with time to create quantitative data

Collecting qualitative data

- Screencasts
    - **Record** screen as participants do tasks

      Many video recorders (e.g., SnagIt)
    - Offers insight into **what** participants did
- What was time consuming?
    - Permits quantitative analysis of **steps & actions**
- Can code more fine-grained time data
    - Does not provide insight into *why* developers did what they did

Collecting qualitative data

- Think-aloud
  - Ask participants to **verbalize** what they are thinking as they work
  - **Prompt** participants when they stop talking for more than a minute or two
  - Offers insight into **why** participants are doing what they are doing
    - What barriers are preventing progress on task?

Analyzing qualitative data

1. **open** coding: read through the text
   look for **interesting** things relevant to research questions
   add notes in the margin (or column of spreadsheet)
   add "**codes**" naming what you saw
   make up codes as you go, not systematic

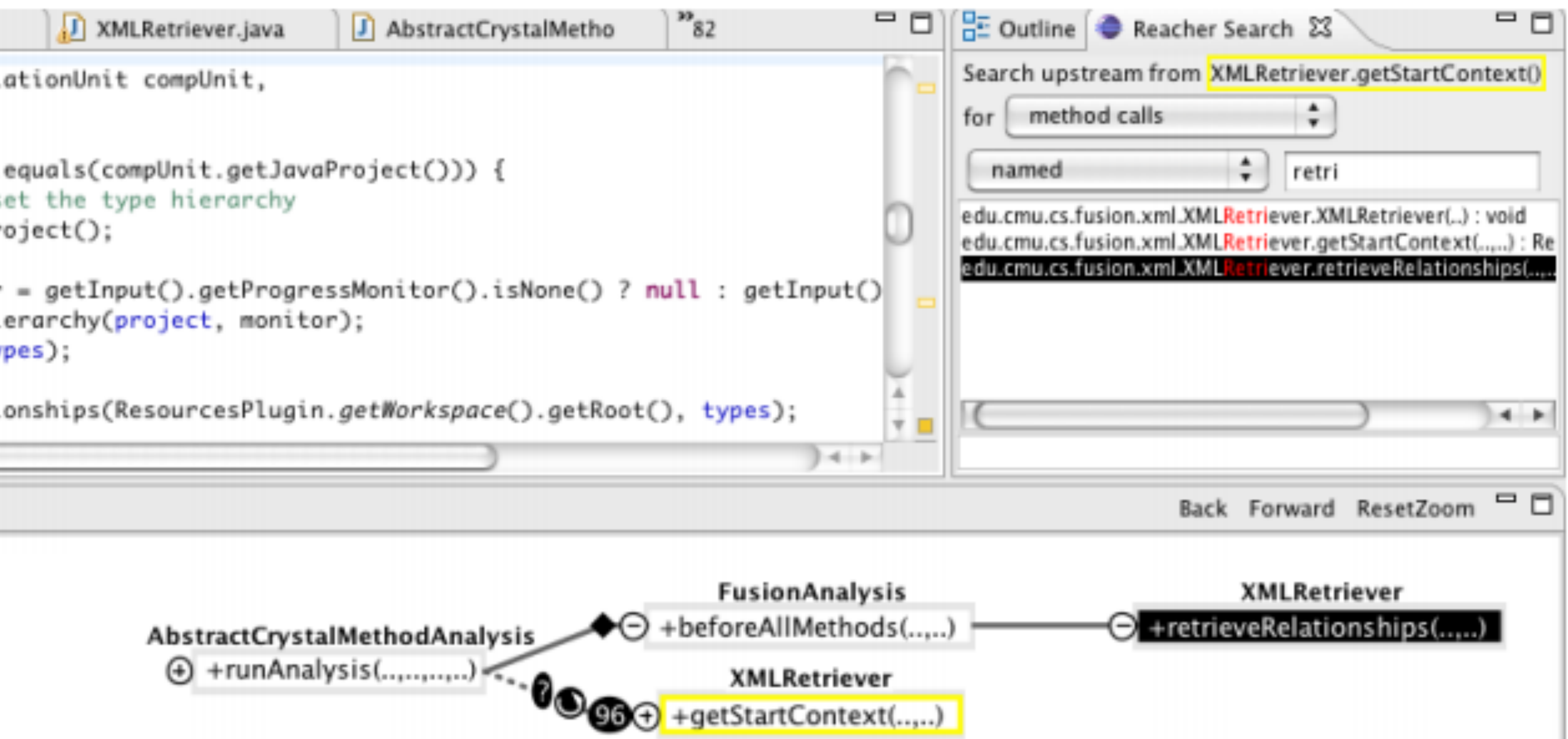
2. **axial** coding: how are open codes related to each other?
   look for **patterns**: causality, ordering, alternatives, groups

3. **selective** coding: from initial codes, select interesting ones
   which codes relate to interesting findings?
   from initial examples, build **definitions** of when a code applies
   systematically reanalyze data and apply codes

4. **second** coder (optional)
   2nd person independently applies codes from definitions
   check for **inter-rater reliability**: if low, iterate defns, try again

Example

REACHER:
Interactive, compact visualization of control flow



Evaluation

Does REACHER enable developers to answer reachability questions *faster*, or *more successfully*?

Method

12 developers

15 minutes to answer 6 **reachability** questions

(order counterbalanced)

Tasks

Based on developer questions in prior observations of developers.

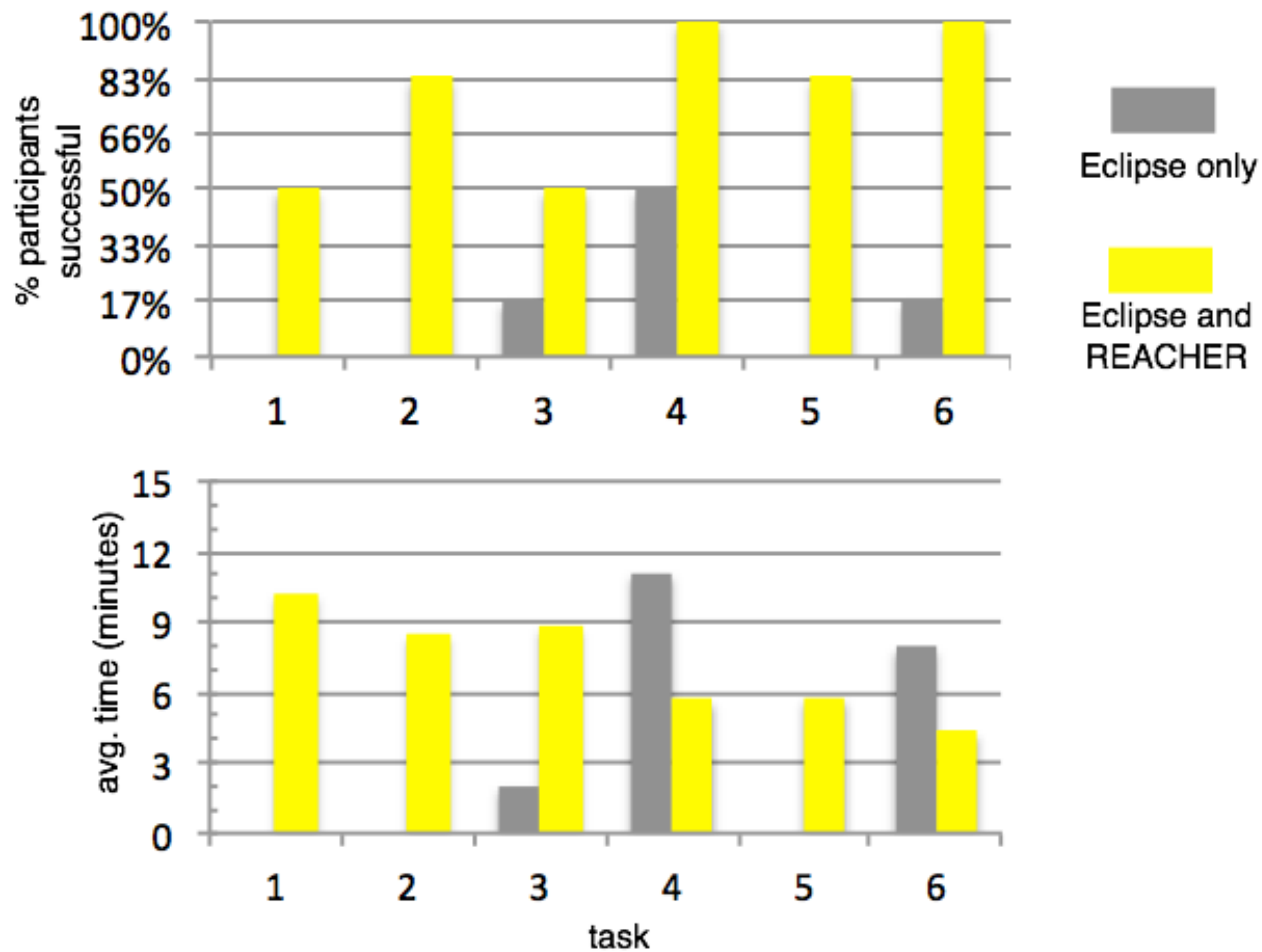
Example:

When a new view is created in jEdit.newView(View), what messages, in what order, may be sent on the EditBus (EditBus.send())?

Results

Developers with REACHER were **5.6** times more **successful** than those working with Eclipse only.

(not enough successful to compare time)

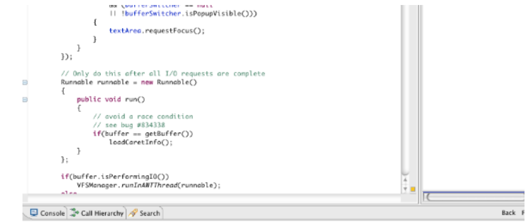


Task time includes only successful participants.

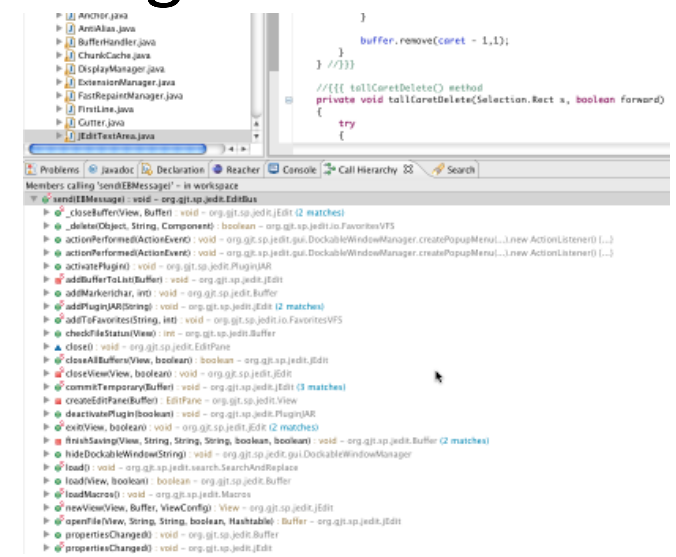
REACHER helped developers stay oriented

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."*

Conclusions

- Controlled experiments with humans can demonstrate **causal** relationship between tool & productivity effects of tool
    - But: results are valid only in specific **context** where study conducted
- Key role for more research to understand **representativeness** of context
    - High value in qualitative understanding of productivity effects to help bridge this gulf

Resources

- Andrew J. Ko, Thomas D. LaToza, and Margaret M. Burnett. (2015) A practical guide to controlled experiments of software engineering tools with human participants. Empirical Software Engineering, 20 (1), 110-141.

- Robert Rosenthal & Ralph Rosnow. (2007). Essentials of Behavioral Research: Methods and Data Analysis. McGraw-Hill.

- Forrest Shull, Janice Singer, Dag I.K. Sjoberg (eds). (2008). Guide to Advanced Empirical Software Engineering. Springer-Verlag, London.

- D. I. K. Sjoeberg, J. E. Hannay, O. Hansen, et al. (03 September 2005). A survey of controlled experiments in software engineering. IEEE Transactions on Software Engineering, Vol. 31, No. 9. pp. 733-753.