INF219 – Software Environments

Second class

Understanding a problem: Empirical Evaluation of Software Environments

Marco Aurélio Gerosa

University of California, Irvine

Spring/2014

Creating a new tool

Inventor-oriented development of a tool

Clipart from http://www.clker.com/

Issues

- Are you a typical developer? Really?

- How do you know that your “problem” is really a problem?

- How do you know the extension of your problem?

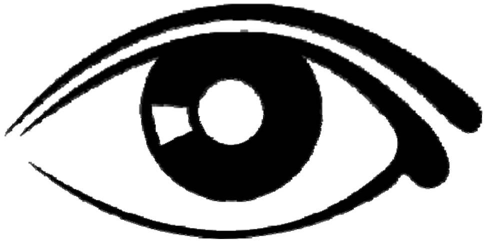
- How do you know that you actually solved or mitigated the problem?

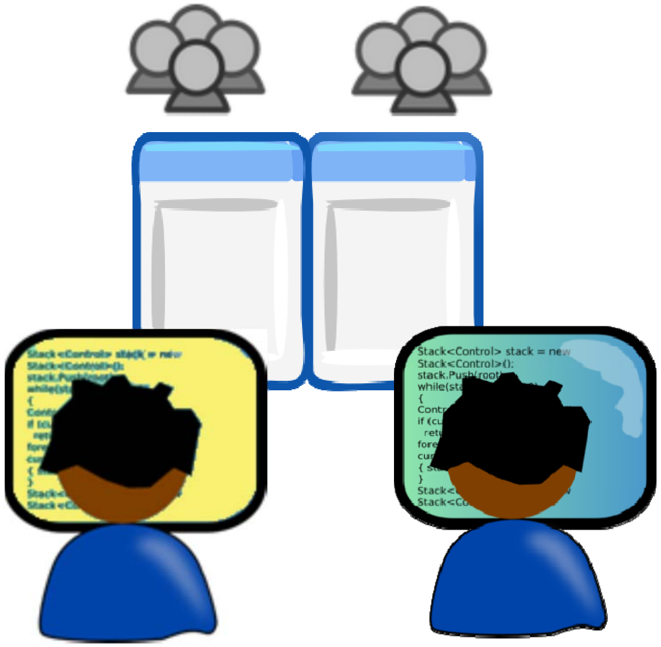
- How do you identify collateral effects of your solution?

Most of the developed tools are **never** used in practice…

Answer

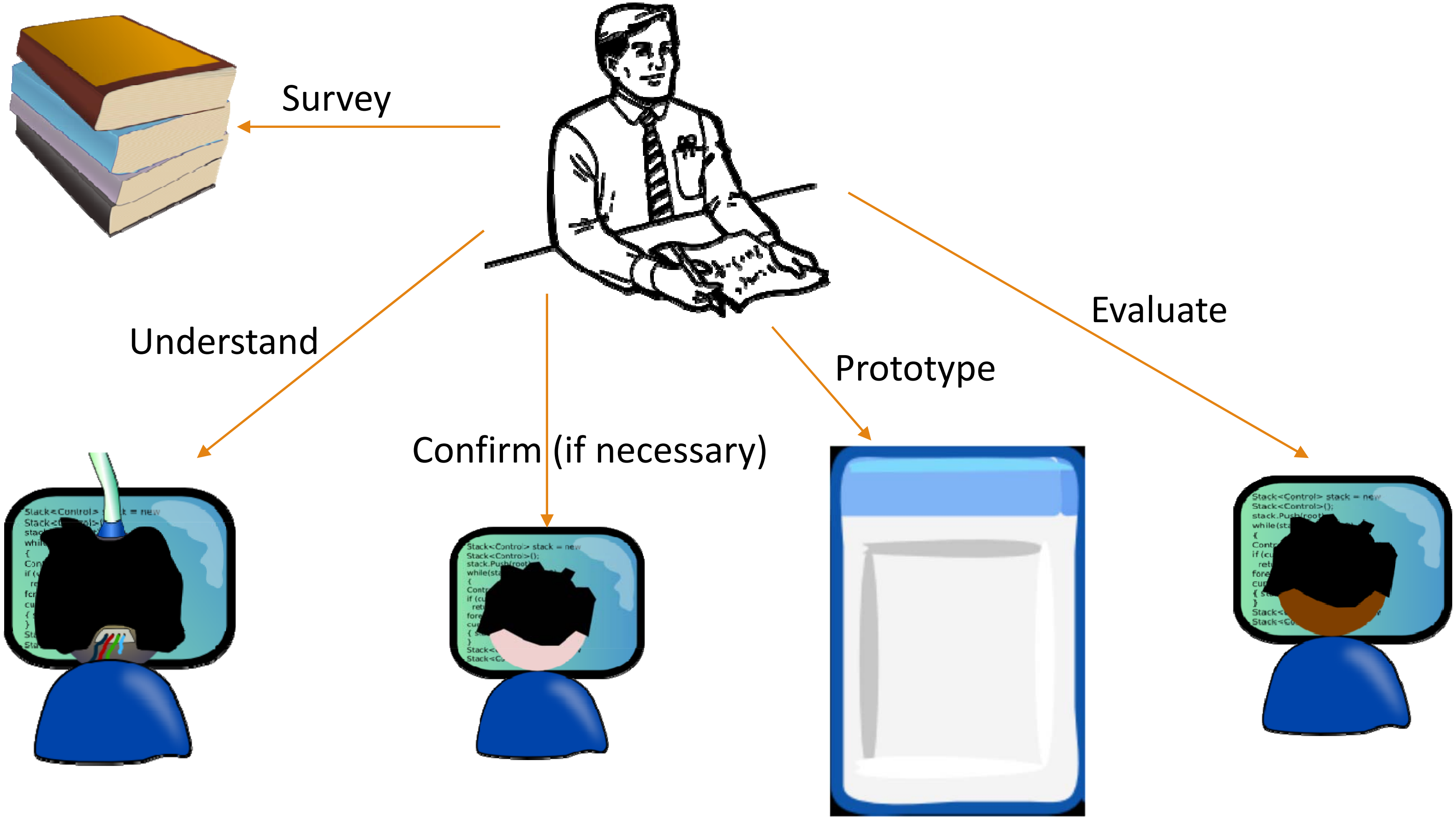
Conduct empirical/experimental studies





Why and **when** to do empirical studies while developing tools for software environments?

Researcher-based development of a tool



Empirical Research vs Pure Invention

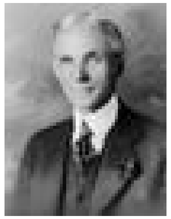
Research is more **solid**, **systematic**, and based on relatively **low biased data**

But, it **takes more time** and it can **refrain creativity**

In practice, try to reach a balance

However, if you want to publish, nowadays you really need to overkill – but it is how science works

Remember:



“If I had asked people what they wanted, they would have said faster horses.”

(Henry Ford)

You can avoid that if you collect the **right data** using the **right means**

Also, you may understand better the real context and effects of the tool, making a better case for the **tool adoption**

- How many tools end up not being used?
- 80s SigChi bulletin: ~90% of evaluative studies found no benefits of tool

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Types of empirical studies

Goal

Exploratory

- define and understand problems and raise hypotheses

Descriptive

- describe a situation

Confirmatory / causal / explanatory

- test hypotheses

Objects of analysis

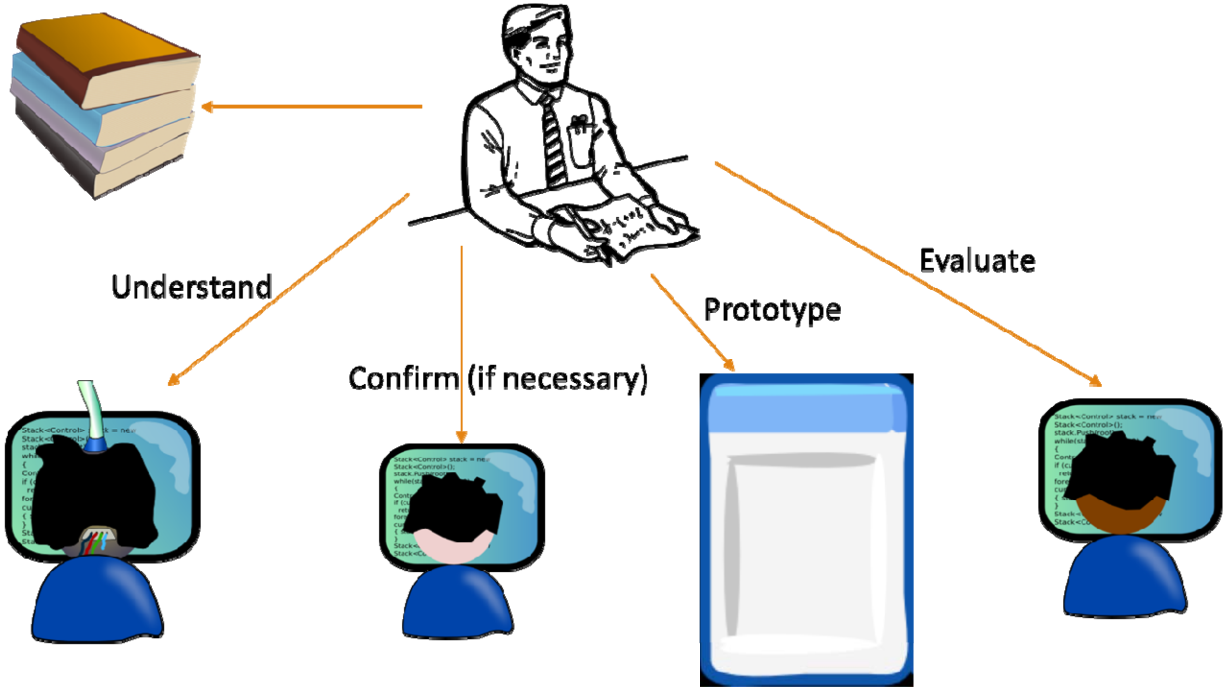
Quantitative

- Numerical measurements / analysis

Qualitative

- Interpretive data acquisition / analysis

What kinds of research can be used?

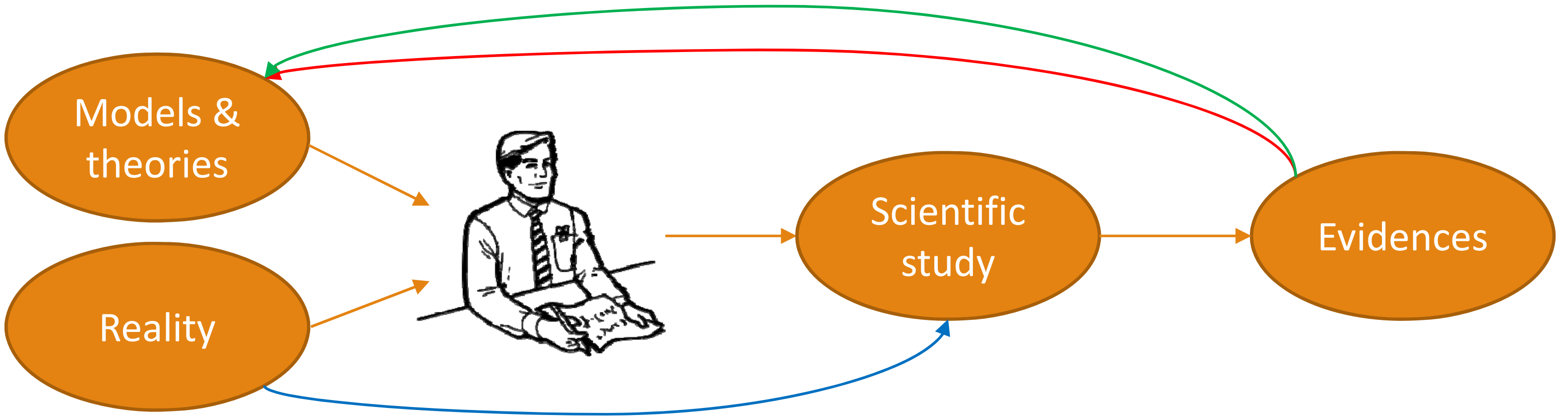


- Exploratory?
- Descriptive?
- Causal?

- Quantitative?
- Qualitative?

Empirical research

Studies provide evidences for or against theories



Theories of developer activity

A **model** describing the **strategy** by which developers **frequently** do an **activity** that describes **problems** that can be **addressed** (“design implications”) through a better designed tool, language, or process that more effectively supports this strategy.

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Exercise

Let’s improve how developers design software systems

How can we do that in a research-based way?

A single study will not answer all questions

Remember!

- Set scope

- Describe limitations of study

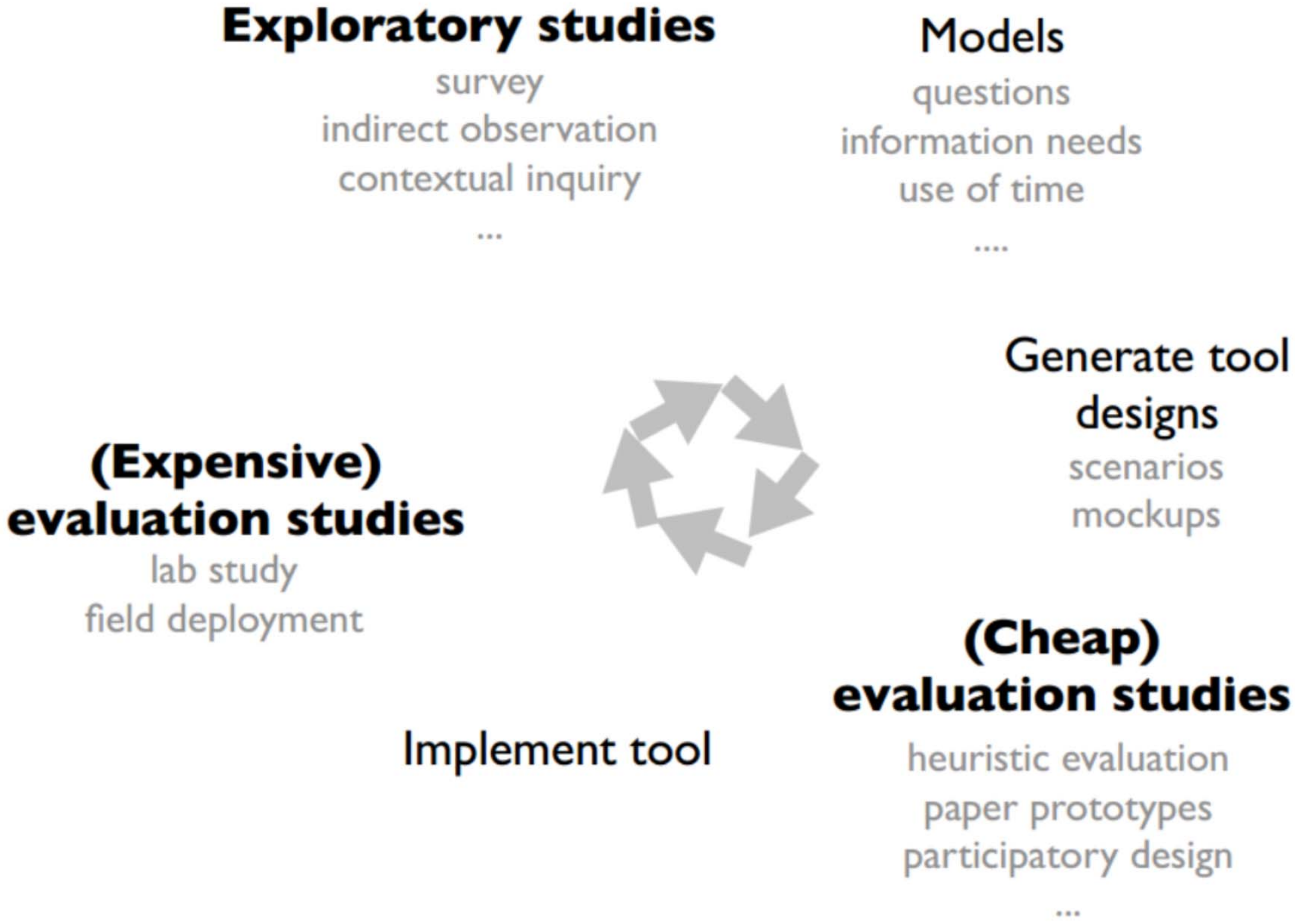
- Pick population to recruit participants from

- Plan follow-up complementary studies

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Research Methods

Example of a cycle



Exploratory studies
survey
indirect observation
contextual inquiry
...

Models
questions
information needs
use of time
....

Generate tool designs
scenarios
mockups

(Cheap) evaluation studies
heuristic evaluation
paper prototypes
participatory design
...

Implement tool

(Expensive) evaluation studies
lab study
field deployment

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Some methods for exploratory studies

- Field observations / ethnography
  - **Observe** developers at work in the field
- Surveys
  - Ask **many** developers specific questions
- Interviews
  - Ask a **few** developers **open-ended** questions
- Contextual inquiry
  - Ask questions **while** developers work
- Indirect observations
  - Study **artifacts** (e.g., code, code history, bugs, emails, ...)

To do high quality studies is quite difficult, but even quick-and-dirty ones can provide useful [Check the literature for additional guidelines]

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Field observations / ethnography

Find software developers

- Pick developers likely to be doing relevant work

Watch developers do their work in their office

Ask developers to **think-aloud**

- Stream of consciousness: whatever they are thinking about
- Thoughts, ideas, questions, hypotheses, etc.

Register it

- Sometimes can be invasive, but permits detailed analysis
- Audio: can analyze tasks, questions, goals, timing
- Video: can analyze navigation, tool use, strategies
- Notes: high level view of task, interesting observations

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Surveys

- Can reach many (100s, 1000s) developers
  - Websites to run surveys (e.g., SurveyMonkey, Google Docs)
- Find participants
  - Probabilistic sampling is only possible for small and well defined population (e.g. within a company)
  - Snowball sampling (mailing list, twitter etc.)
- Prepare multiple choice & free response questions
  - Multiple choice: faster, standardized response
  - Free response: more time, more detail, open-ended
- Background & demographics questions
  - E.g., experience, time in team, state of project, ...
- Open comments

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Semi-structured interviews

Define a script

Prompt developer with question on focus areas

- Let developer talk
- Follow to lead discussion towards interesting topics

Manage time

- Move to next topic to ensure all topics covered

It is hard to not bias the interviewee

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

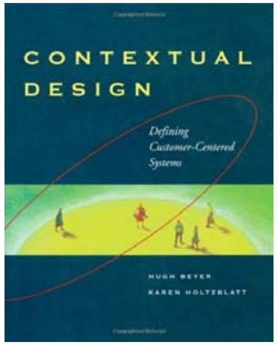
Contextual inquiry

Interview while doing field observations

Learn about environment, work, tasks, culture, breakdowns

Principles of contextual inquiry

- Context - understand work in natural environment
- Ask to see current work being done
- Seek concrete data - ask to show work, not tell
  - Bad: usually, generally
  - Good: Here’s how, let me show you
- Partnership - close collaboration with user
  - User is the expert
- Interpretation - make sense of work activity
  - Rephrase, ask for examples, question terms & concepts
- Focus - perspective that defines questions of interest



LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Indirect observations

Indirect record of developer activity

Examples of **artifacts** (where to get it)

- Code & code changes (version control systems)
- Code changes
- Bugs (bug tracking software)
- Emails (project mailing lists, help lists for APIs)

You can also collect data from instrumented tool (e.g., Hackstat)

Advantage:

- **Lots** of data, easy to obtain

Disadvantages:

- Can only observe what is in the data

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Examples

Which methods would you use in these situations?

1. You'd like to design a tool to help web developers reuse code more easily.

2. You'd like to help developers better prioritize bugs to be fixed.

- Field observations?
- Surveys?
- Interviews?
- Contextual inquiry?
- Indirect observations?

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Evaluation

Ok, you figured out a problem and conceived a tool.

But is this the right tool? Would it really help?

Which features are most important to implement?

Solution: low cost evaluation studies

- Evaluate mockups and prototypes before you build the tool!

Tool isn't helpful: come up with a new idea

Users have problems using tool: fix the problems

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Low cost evaluation methods

Paper prototyping

- Do tasks on paper mockups of real tool
- Simulate tool on paper

Wizard of Oz

- Simulate tool by computing results by hand

Heuristic evaluation

- Assess tool for good usability design

Cognitive walkthrough

- Simulate actions needed to complete task

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Paper prototyping

Build paper **mockup** of the tool

- May be rough sketch or realistic screenshots

Often surprisingly effective

Experimenter plays the computer

- Experimenter **simulates** tool by adding / changing papers

Good for checking if user

- Understands interface terminology
- Commands users want match actual commands
- Understands what tool does
- Finds the tool useful

Challenges - must **anticipate** commands used

- Iteratively add commands from previous participants
- Prompt users to try it a different way

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Wizard of Oz

Participant believes (or pretends) to interact with real tool

- Experimenter **simulates** (behind the curtain) tool
- Computes data used by tool by hand

Participant's computer is "slave" to experimenter's computer

Especially for AI and other hard-to-implement systems

- E.g.: Voice user interface - experimenter translates speech to text

Advantages

- High **fidelity** - user can use actual tool before it's built

Disadvantages

- Requires **working** GUI, unlike paper prototypes

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Prototyping

Increased fidelity

- Paper
- Implemented UI (no business logic)
- Wizard of Oz
- Implemented prototype
- Real system

Better if sketchier for early design

- Use paper or “sketchy” tools, not real widgets

- People focus on wrong issues: colors, alignment, names

- Rather than overall structure and fundamental design

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Heuristic evaluation [Nielsen]

Multiple evaluators use dimensions to identify usability problems

Evaluators aggregate problems & clarify

Structured assessment based on experience

Problems may be are categorized according to their estimated impact on user performance or acceptance

Examples of heuristics: (Jakob Nielsen)

- Visibility of system status; Match between system and the real world; User control and freedom; Consistency and standards; Error prevention; Recognition rather than recall; Flexibility and efficiency of use; Aesthetic and minimalist design; Help users recognize, diagnose, and recover from errors; Help and documentation

Advantage:

- Users are not necessary

Disadvantage

- Highly influenced by the knowledge of the expert reviewer

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Cognitive walkthrough

How easy it is for new users to accomplish tasks with the system?

Cognitive walkthrough is task-specific

Task analysis - sequence of actions required by a user to accomplish a task and the responses from the system to those actions

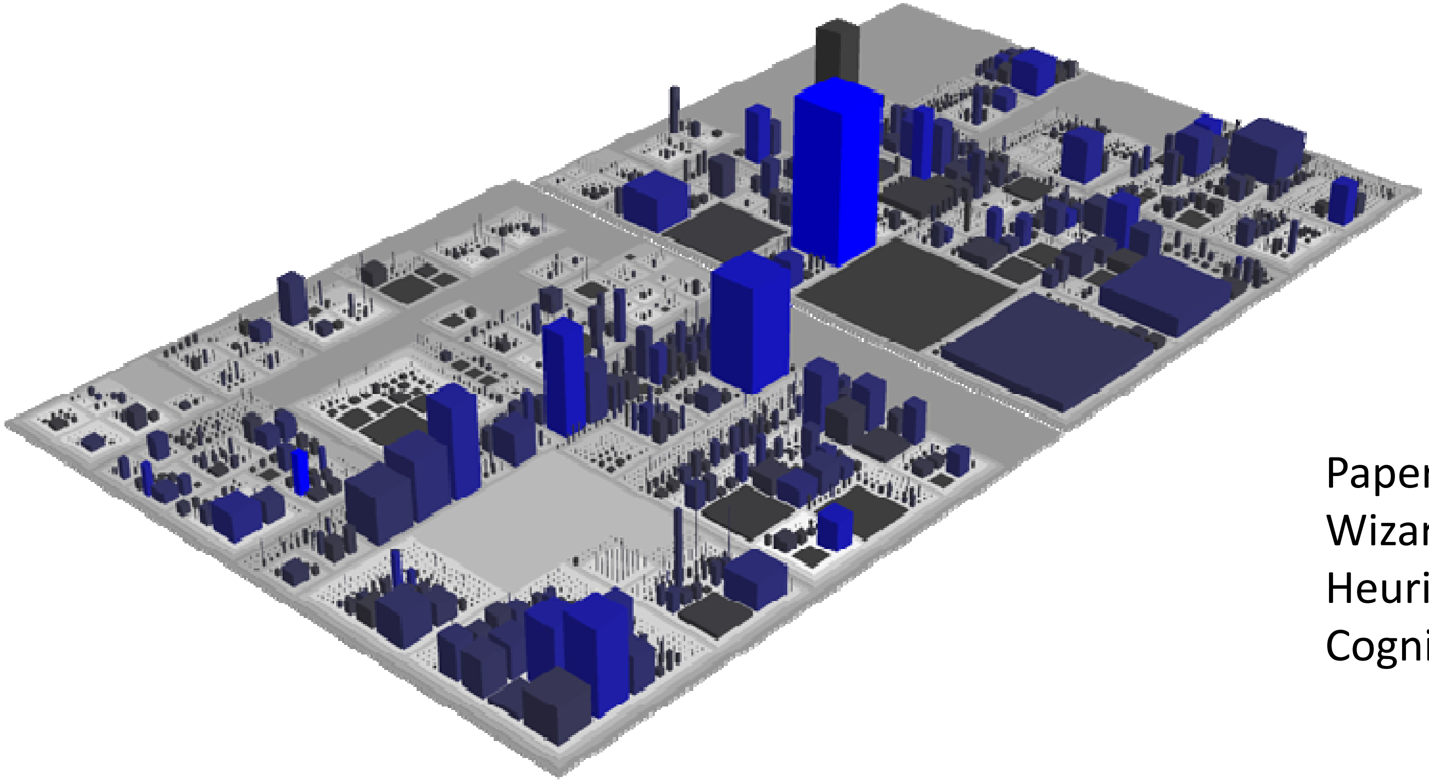
Evaluators walk through the steps, asking themselves a set of questions

- Will the user try to achieve the effect that the subtask has? Does the user understand that this subtask is needed to reach the user's goal?
- Will the user notice that the correct action is available?
- Will the user understand that the wanted subtask can be achieved by the action?
- Does the user get feedback? Will the user know that they have done the right thing after performing the action?

Exercise

How would you use the evaluation methods in this situation?

You're designing a new notation for visualizing software



Paper prototyping
Wizard of Oz
Heuristic evaluation
Cognitive walkthrough

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf
Codecity: http://www.inf.usi.ch/phd/wettel/codecity.html

More robust evaluations

You want to write a paper claiming that your tool is useful

You want to get a company to try it out.

Solution: run a higher cost, but more convincing evaluation study

Lab experiments - controlled experiment to compare tools

- Measure differences of your tool w/ competitors
- Usually based on quantitative evidence

Field deployments

- Users try your tool in their own work
- Data: usefulness perceptions, how use tool
- Can be more qualitative

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Lab studies

Users complete tasks using your tool or competitors

- Within subjects design - all participants use both
- Between subjects design - participants use one

Typical measures - time, bugs, quality, user perception

- Also measures from exploratory observations (think-aloud)
- More detailed measures = better understood results

Advantage

- Controlled experiment (more precision)

Disadvantages

- Less realism and generalizability
- Users still learning how to use tool, unfamiliar with code
- Benefits may require longer task

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Field deployments

Give your tool to developers. See how they use it

Low control, more realism

Data collection: interviews, logging data, observations

Qualitative measures

- Perception: do they like the tool?
- Use frequency: how often do they use it?
- Uses: how do they use it? what questions? tasks? why?
- Wishes: what else would they like to use it for?

Quantitative comparison is possible, but it is hard

- Different tasks, users, code

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Analysis

Techniques for qualitative data analysis

Contextual design

- Set of models for understanding how work is done

Content analysis / grounded theory

- Technique for analyzing texts
- Used both to find patterns in data & convert to quantitative data

Process models

- Models of steps users do in a task

Taxonomies

- What things exist, how are they different, and how are they related?

Affinity diagrams

- Technique for synthesizing many disparate observations or interpretations into a coherent whole

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Quantitative data analysis

Frequency

- How often do things occur? (counts, %s, avg times, ...)
- Descriptive statistics

Correlational

- How are multiple variables related? (correlations, ...)
- Estimate variable x from variables a, b, c (regression, classifiers, ...)

Controlled experiment (causal)

- Statistical test

Qualitative vs. quantitative

Qualitative analysis most useful for

- Figuring out what's there, what's being done
- What's important?
- How are things done?
- Why is person doing / using / thinking something?

Limitations

- Small n: few examples, may not generalize
- Interpretations could be biased

Quantitative analysis most useful for

- Testing hypotheses
- Investigating relationships between variables
- Predicting

Limitations

- Lack interpretation

Therefore, great studies mix both approaches

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Other aspects of a scientific study

What to observe

Learning a new tool

Pilot studies

IRB Approval

Maximize validity
- more participants, data collected, measures
- longer tasks
- more realistic conditions

Minimize cost
- fewer participants, data collected, measures
- shorter tasks
- less realistic, easier to replicate conditions

Studies are not proofs - results could always be invalid
- Software Engineering is very context dependent
- Don’t sample all developers x tasks x situations, and measures are imperfect

Search results that are
- interesting
- relevant to research questions
- valid enough so that your target audience believes them

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Some types of validity

Validity = should we believe the results?

Construct validity

- Does measure correspond to construct or something else?

External validity

- Do results generalize from participants to population?

Internal validity

- Are the differences between conditions caused only by experimental manipulation and not other variables? (confounds)

See also:
https://en.wikipedia.org/wiki/Bias_(statistics)
https://en.wikipedia.org/wiki/Cognitive_bias

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Conclusion

Final considerations

Field studies of programmers reveal interesting new areas for *tool* research and development

- Can focus research on important problems
- *Design from Data* about *real* problems, barriers, opportunities

Following research methods will result in *better quality tools*

- More usable, effective, etc.

Software Engineering tools and methods often benefit from valid *evaluation* with people

- Need real evidence to answer questions about what is better/faster/easier
- Often demanded by reviewers
- Relevant to any claims of better/faster/easier for people
- There are valid evaluation criteria beyond "Taste", "Intuition", "My experience," and anecdotes

LaToza (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf

Summary

- Pure invention x empirical research approach for developing innovative tools
- When to do empirical studies
- Types of empirical studies
- Research methods
  - exploratory studies
  - low cost evaluation
  - robust evaluation
- Qualitative and quantitative analysis
- Some points to observe when conducting studies
- Validity

To Do

- Define the groups and the topic of your seminar
- Read the papers assigned for the 3rd class (check your group):
  - http://www.ime.usp.br/~gerosa/classes/uci/inf219
- Produce the slides-based summary for the two papers

"Wherever you go, go with all your heart." (Confucius)

Thank you!

See you next class

Marco Gerosa
Web site: http://www.ime.usp.br/~gerosa
Email: mgerosa@uci.edu / gerosa@ime.usp.br
Office: DBH 5228 (by appointment)

Acknowledgments

This class was based on the slides from Thomas LaToza:

- http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture02-HCI%20Methods%201.pdf
- http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture03-HCI%20Methods%202.pdf