Lecture 13: Presentations, user experience

CS 5150, Spring 2025

Administrative Reminders

- Assignment A3 released – due Mar 26
- Schedule your midpoint presentations (last date Mar 25)
  - Let me know if you need a meeting room
- For teams with external clients: include your client and instructor in the meeting
- Submit Peer Evaluation by Mar 7

Lecture goals

- Give effective presentations to stakeholders
- Solicit feedback on user interface designs

Presentations

Presentations in Software Engineering

- Important in every project phase
  - Marketing to potential clients
  - Reporting progress to senior management
  - Reports and demonstrations to clients
  - Communication with colleagues on dev team
- Important for career growth
  - Unlikely to achieve leadership position if you cannot give decent presentations

Not everyone is born a great presenter, but everybody can be well-prepared

- If you are uncomfortable, take every opportunity to gain experience

Presentations in CS 5150

- Two required presentations:
  - Progress update during 3rd sprint
  - Preliminary delivery during 5th sprint
- Every team member must present a portion of one presentation
  - Less experienced presenters will be more comfortable presenting things they personally worked on
- Audience: your client
  - May not be technical (internal projects: client is manager, not developer)
- Course staff will evaluate presentation contents and technique

Planning for presentations

- Know your purpose, audience, and resources
  - What is the presentation meant to achieve?
    - Confirm understanding?
    - Obtain client approval?
    - Propose new feature?
    - Solicit feedback on prototype?
    - Build excitement/buy-in?
    - Request assistance?
    - Report progress?
    - Train users?
  - Who must attend the presentation for it to achieve its purpose?
    - Prospective clients?
    - Project management?
    - System users?
    - Other developers?
  - How is your presentation constrained?
    - Time available
    - Projector/screen sharing?
    - Internet access?

Time management

- **Midpoint**: 15 min for presentation, 15 min for questions
- **Final**: 30 min presentation, 15 min for questions
- Expect interruptions (presentation must serve the audience; is not an end in itself)

- Have an agenda that fulfils the presentation's purpose
- Rehearse your presentation on the clock!

Remote presentations

- Good audio is essential
  - Make a practice recording with all presenters in their anticipated locations/positions
- Good video needs good lighting
- Client must be able to see all demonstrations and visual aids
  - Screen share
  - Whiteboard/annotations
  - Auxiliary camera

- Beware multiple computers in one room

In-person presentations preferred

Topics

- Topics on agenda should serve purpose of meeting
  - Description of what you have agreed to deliver to your client (shared definition of success)
  - Summary of progress since last presentation/report
  - Unexpected events and risks
  - Overview of remaining plan to complete and deliver project
  - Test plan and test cases
  - Results of user testing
  - Technical hurdles (if client is technical)
- Demonstrations are always welcome
  - *Show* mock-ups / demonstrations / prototypes *before* talking about them

CS 5150 topics

- Early-stage topics
  - Confirm agreement on scope and goals
    “The project will be a success if …”
  - Progress to date
    “This is our understanding of your requirements…”
    - Mock-ups, designs, etc.
  - Schedule and plan
    “The main risks are…”
    - What has changed since feasibility study?
- Mid-stage topics
  - Demonstration of operational prototype or delivered features
  - Results of user studies

Visual aids

- Slides
  - Common, but not required (and can be a liability)
  - Keep things simple (purpose is conveying information, not entertainment)
  - Must be legible
    - Large fonts (including in figures!) – 20pt minimum
    - Dark text on light background
  - Use to facilitate presentation, not as a reference source
- Handouts
  - Can accommodate more simultaneous detail than a slide
  - Beware potential for distraction

Preparations

- Must have a rehearsal
  - Include all demos and visual aids; don’t skip anything
  - Use same laptops as you plan to use later
  - Any unrehearsed changes are a risk – minimize them
  - Time each section
- Plan presenter coordination
  - Option 1: Moderator calls on each presenter
  - Option 2: Each presenter introduces the next
- Test equipment in location if possible
  - Projector connection, network connection, power availability

Presentation behavior

- Presenter (1) should stand; others should sit
- Appoint a recorder
- Briefly introduce each team member
- When asked a question,
  - If presenter knows answer, answer it
  - Presenter may ask another team member to respond
  - Okay to make note and reply later
- Never interrupt your colleagues
  - If you have information to add, raise your hand, allow presenter to decide if/when to call on you

Demonstrations

- Require preparation and practice to be successful
- Technical preparations:
  - Load and configure all software before presentation. Test it, then change nothing
  - If you need test data or accounts, create them in advance
  - If complex commands must be typed, create a cheat sheet or shell script. Ensure they work verbatim
- Prepare a script
  - Include setup, list of examples, task assignments, and cleanup
- Tell audience what they are seeing
  - Production-ready code? Mock-up? Proof-of-concept?

Presentation tips

- Not a lecture!
  - Also not an advertisement
- You are not the audience
  - Try to imagine the client's perspective
- Not an end in itself
  - Be able to articulate its purpose
- Not a controlled document
  - Should not serve as primary documentation
- Not about showing off
  - Don't mislead audience or overpromise
- Explain purpose of topics, figures
  - Why should the audience pay attention to this?

Looking ahead: CS 5150 final presentation

- Goals
  - Personal & team satisfaction from handing over good work to client
  - Complete course in good style with good grade
  - Clean handover without loose ends
  - A good basis for future involvement with client, team, or project
- Audience interests
  - Client: has invested effort in this project
    - Is it ready for production?
    - Should they invest more to deploy/maintain it?
    - Should this approach be abandoned?
  - Course staff
    - What has been accomplished?
    - What has been learned?
    - Is the client satisfied?
    - Are you handing over a maintainable system?

Final presentation components

- Demonstration of operational system
  - Walk through scenario
  - Be honest about gaps, weaknesses
- Presentation
  - Brief review of context, goals
  - Honest summary of achievements and misses
  - Summary of what is being delivered
- Time for discussion
- Must fit within 30 min
  - Cannot walk through everything

User experience

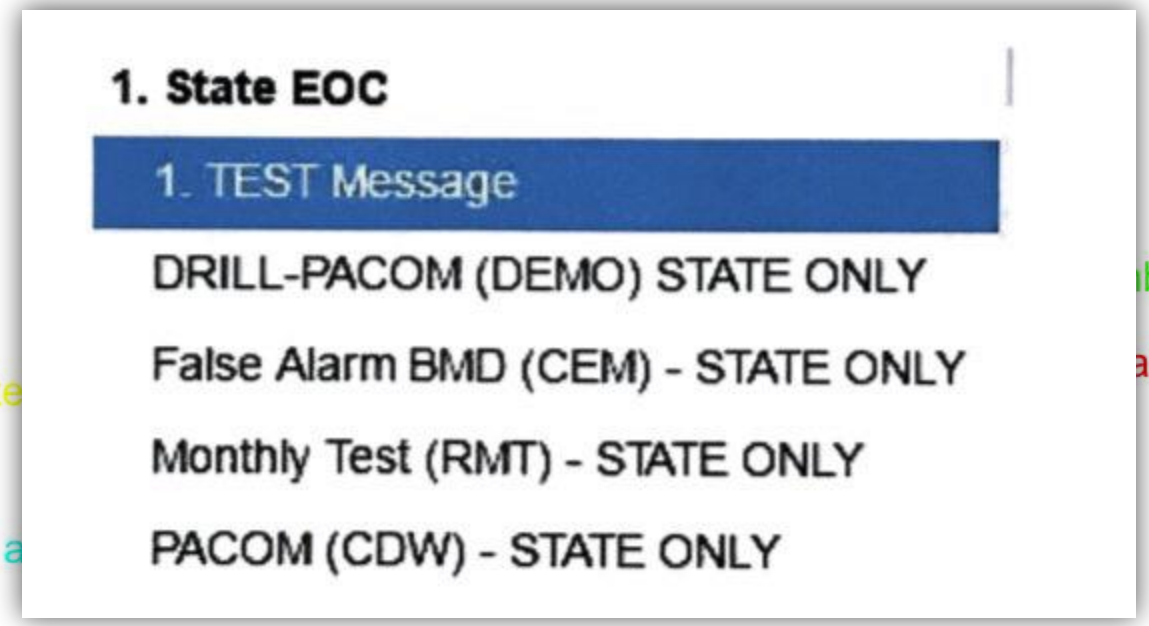
Overview

- A system is only as good as the interface it provides to users
  - Symptoms of poor usability:
    - Failure to attract, retain market share (users give up in disgust, Orkut vs Facebook)
    - Users fail to find or misinterpret important information
    - System can be operated in an unsafe manner (example: 737 MAX)
  - Usability aspects that improve system effectiveness:
    - Appropriate functionality
    - Easy navigation
    - Fast response times
    - Elegant, organized design
- Supporting users is more than a cosmetic flourish
  - Developing good UI takes skill and time

Example: Hawaiian public alerts

Drills/te

Weather a



ber alerts

alerts

Terminology

- User Interface (UI)
  - Look and behavior of system's controls
- User experience (UX)
  - All factors that contribute to usability of computing system
  - Encompass entire usage lifecycle, from discovery to accomplishing goals
  - Focus on user satisfaction
- Human-Computer Interaction (HCI)
  - Academic discipline studying how people interact with computers
  - Many courses and research programs in Information Science and Communications departments

Usability requirements and evaluation tools

	Initial	Mock-up	Prototype	Production
Client's opinions	✓	✓	✓	
Competitive analysis	✓			
Expert opinion	✓	✓	✓	
Focus groups	✓	✓		
Observing users		✓	✓	✓
Measurements			✓	✓

Focus group

- Group interview – helps generate ideas that would not have occurred individually
- Participants: 5-12 potential users with similar viewpoints
- Interviewer
  - Ask a structured set of questions
  - Encourage group discussions
  - May show mock-ups
  - Summarize conclusions
- Recorder takes notes
- Repeat with contrasting user groups

Internal project users

- Internal projects are code review tools; users are software developers
- Your classmates are candidate users!
  - Not including your team members
  - Not including teams working on the same feature
- Recruit classmates for focus groups, user testing
  - Can coordinate on Ed Discussion
  - May be easiest to pair teams
- Documentation of user studies will be expected in future report

Accessibility

- Users have varying ability to interact with computer interfaces
  - Color blindness (1/12 men, 1/200 women)
  - Poor or no vision
  - Lack of hearing
  - Poor manual dexterity
  - Limited language skills, domain vocabulary
  - Sensitivity to flashing light, motion sickness
- Accessibility requirements constrain the user interface
  - Many systems have a legal requirement to support users with disabilities
  - Example: Compliance with Section 508 of US Rehabilitation Act https://www.section508.gov/
- Some technologies may not be suitable
  - Examples: Flash, untagged PDF, immature widget toolkits

Equipment requirements

- Software runs on wide variety of devices, with diverse configurations, in many environments
  - Screen size
  - Graphics performance
  - Network bandwidth, latency, stability
  - Peripheral hot plugging

- Be explicit about equipment assumptions/requirements
- Be explicit about failure handling
- Test on variety of equipment (including extremes)

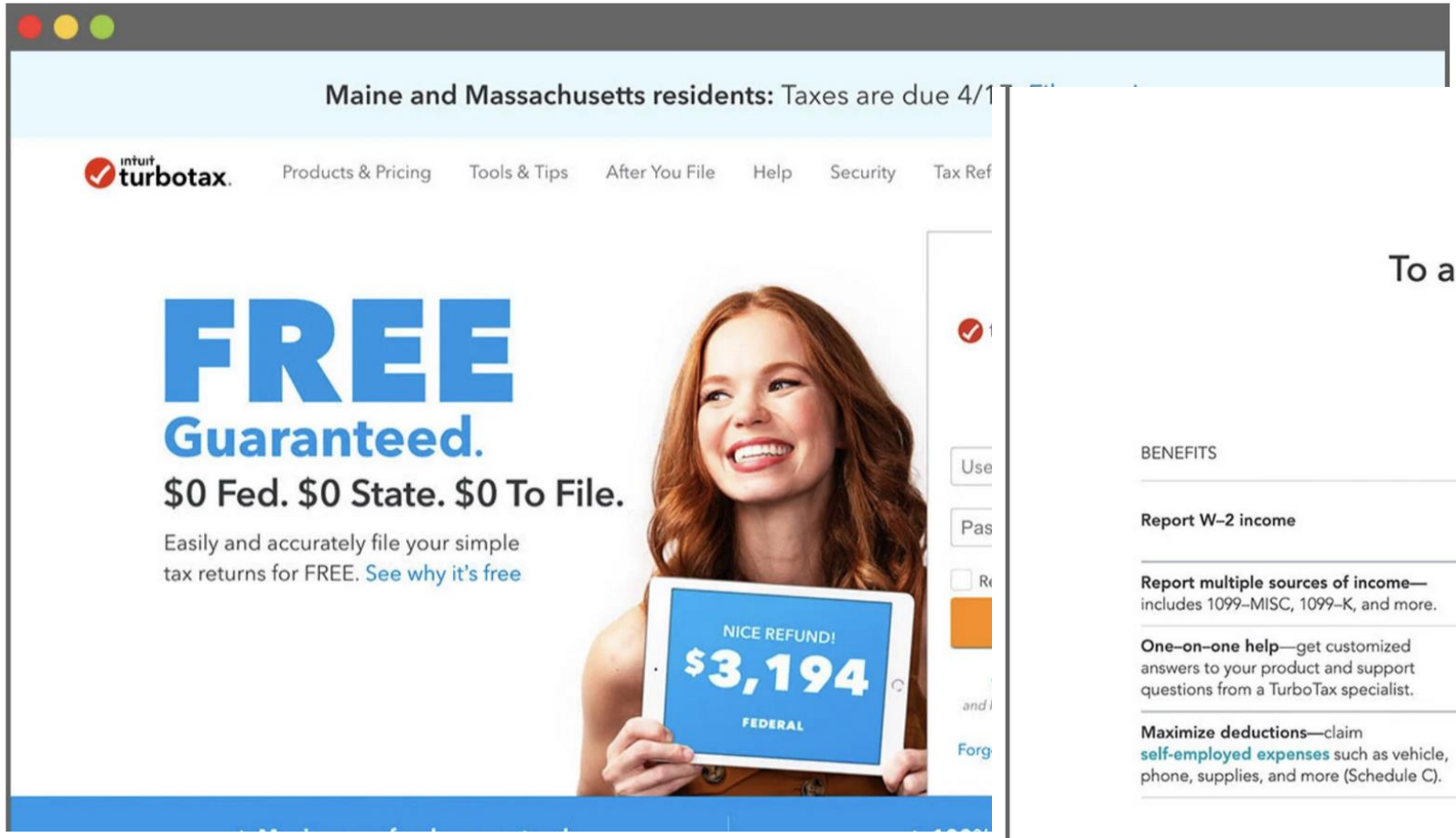
- Example: Chat application

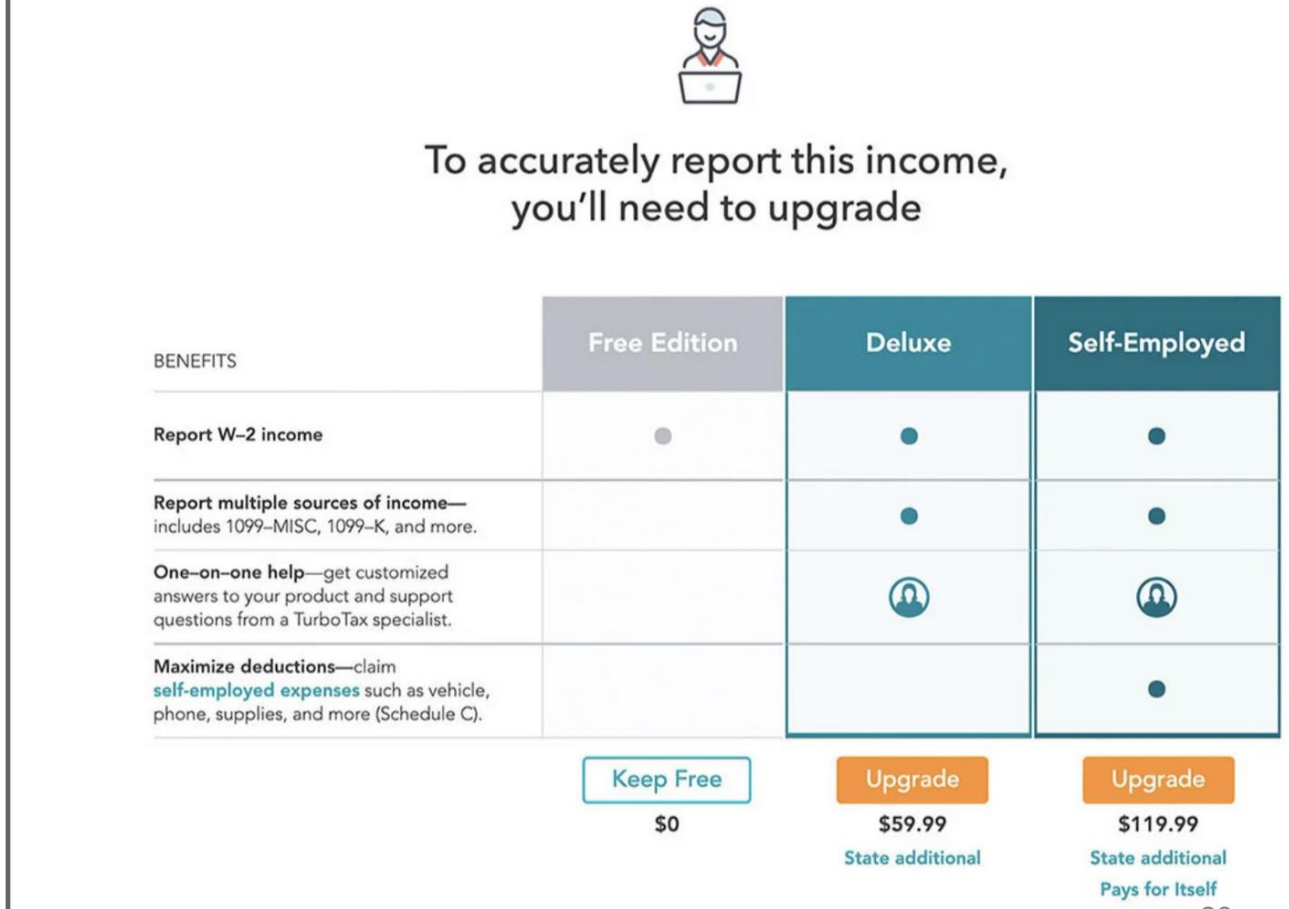
Dark patterns

- Many of our experiences with UI are in a marketing context
  - Goal is to maximize engagement and manipulate user decisions
  - Being commonplace and effective in marketing goals does not make a design pattern good
    - Avoid simply aping features of slick websites (even if libraries make it easy to do so)
- User-centric design
  - Interface should facilitate, not redirect, users' objectives

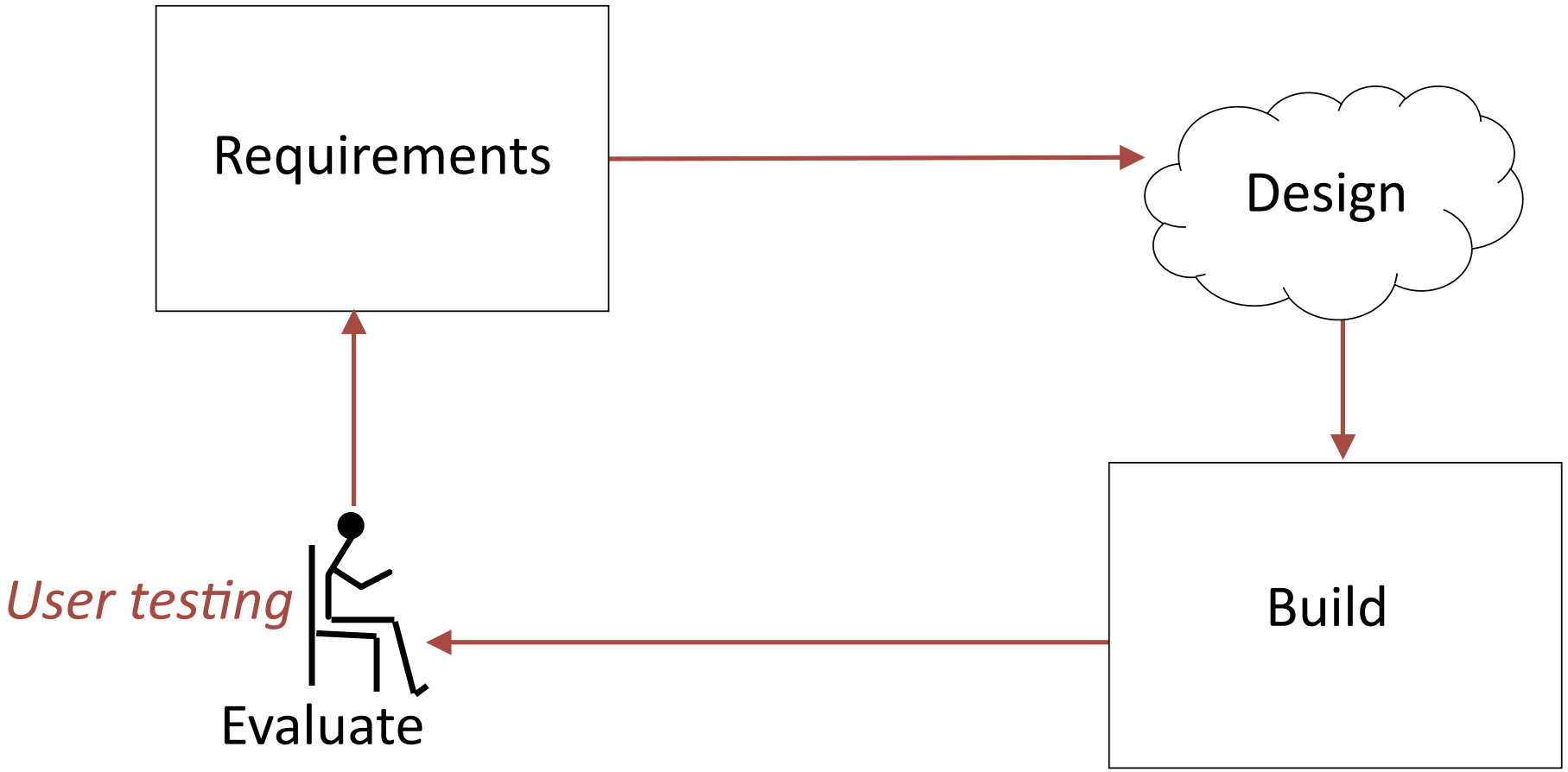
- https://cacm.acm.org/magazines/2020/9/246937-dark-patterns/fulltext

Dark Patterns (Misleading Info)





Analyze/design/build/evaluate loop



Development processes

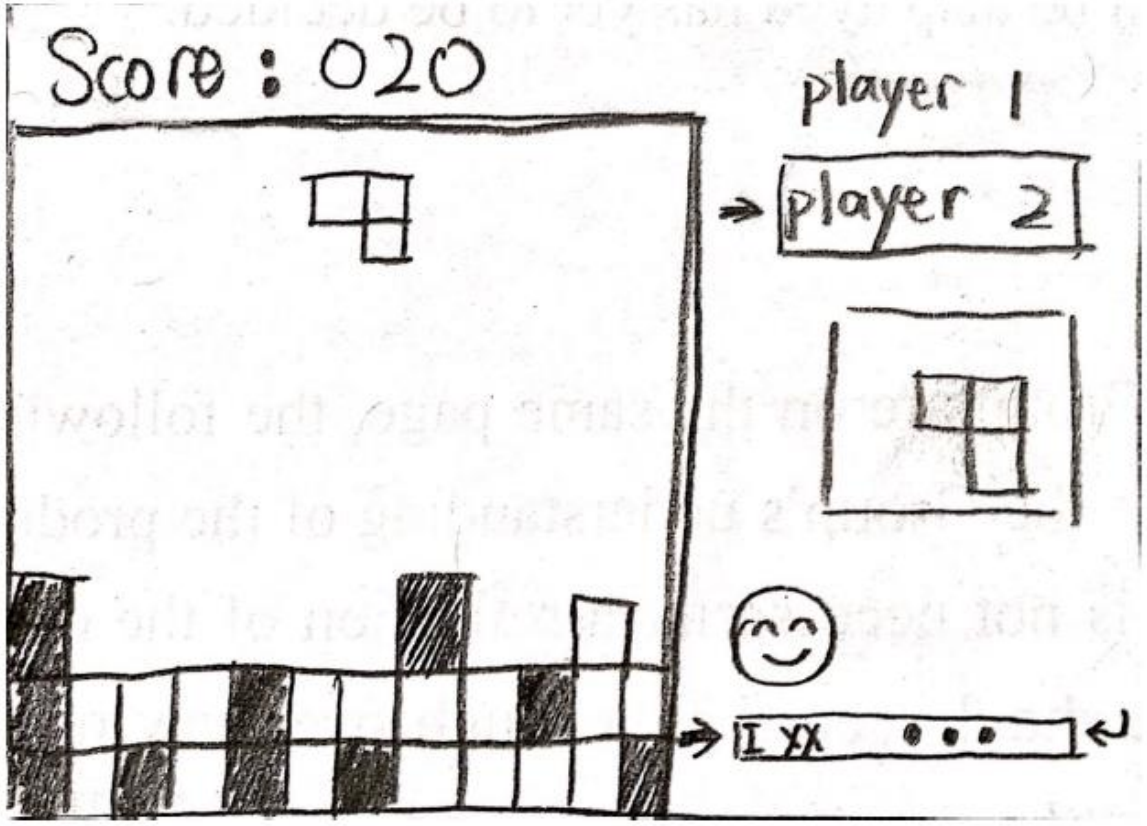
- Written requirements poor fit
  - Requirements benefit from sketches, comparison with existing systems
  - Designs should include graphical elements, benefit from prototypes
- UI must be tested with users; expect requirements and design changes
  - Schedules must include time for testing and time to make changes

UI prototypes

- Preliminary version used to iterate rapidly between requirements and design
  - Minimize polishing effort to maximize iteration speed

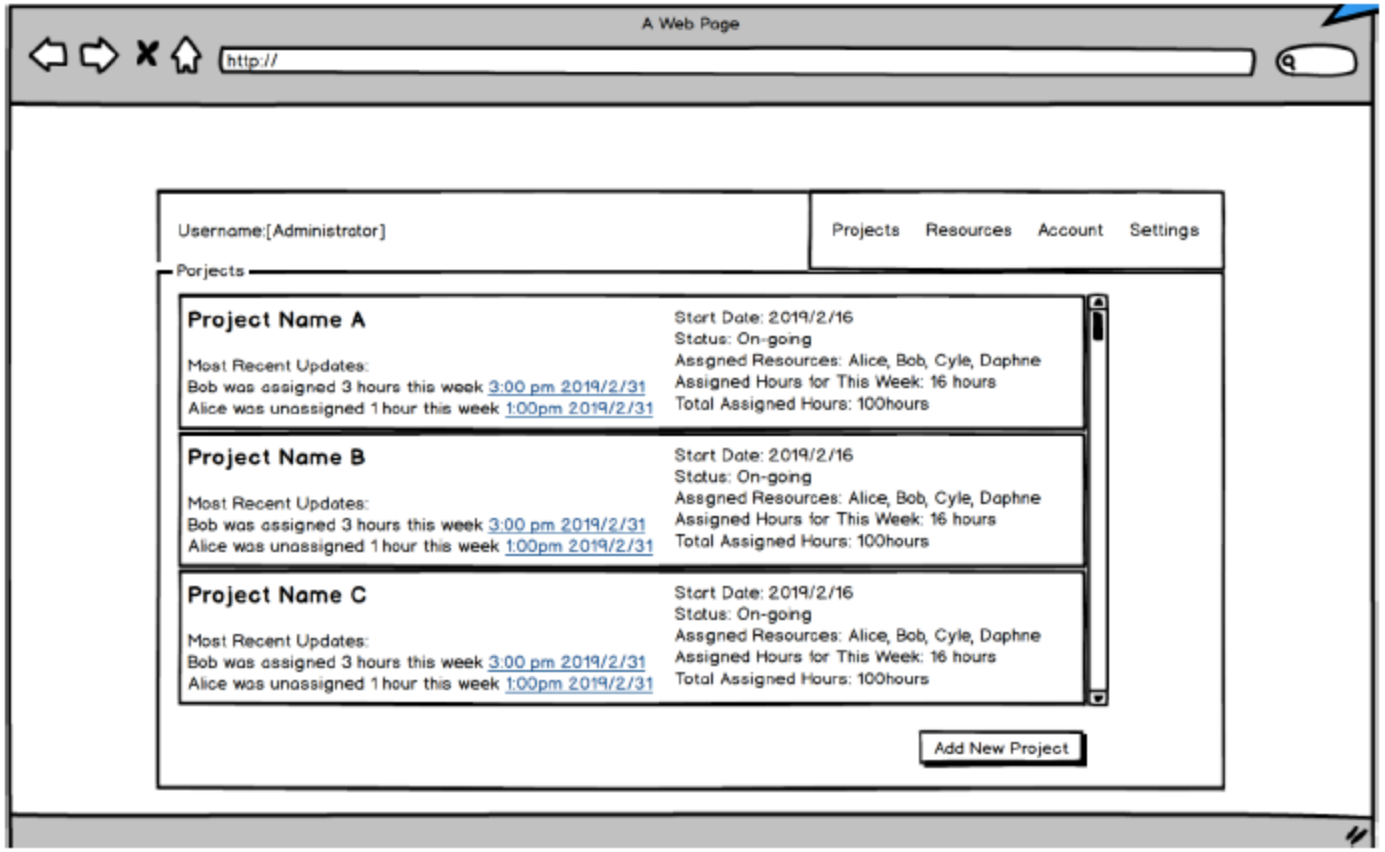
- Paper sketches
  - Lowest effort, so amenable to major changes
- Wireframe
  - Outline layout
- Mock-up
  - Graphic designs with detailed layout, color
- Operational prototype
  - Enables interaction and navigation

Example: paper prototype



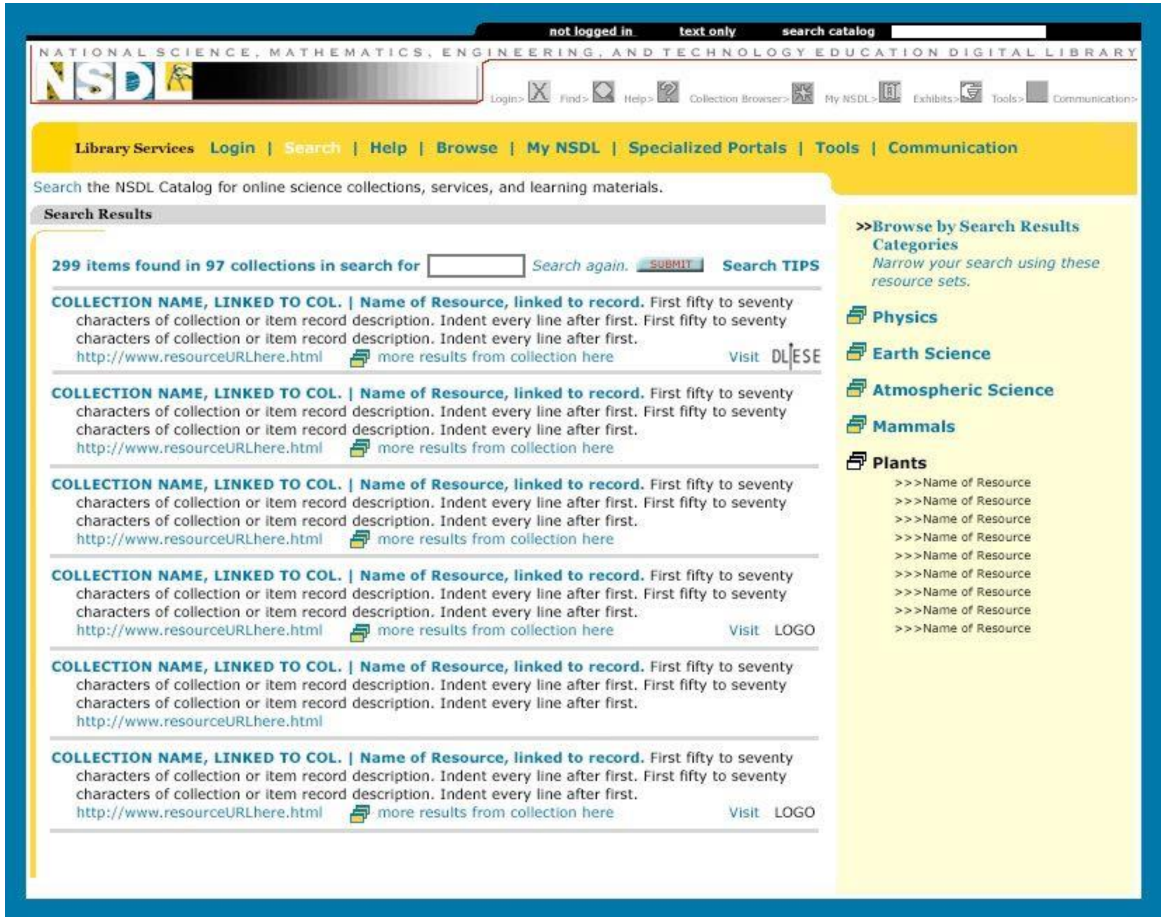
From past CS 5150 project

Example: wireframe



From past CS 5150 project

Example: mock-up



Interactive prototypes

- "Clickable" - responds in limited ways to user interactions
  - Illustrate time-dependent design
    - Animations
    - Drag-and-drop
  - Navigate between pages, dialogs
- Not production code
  - Does not update model data, trigger external events
  - Make sure client understands limitations

- Collaborative tools:
  - Figma
  - Adobe XD
  - (many others)

Models

Relating user and system models

Mental model

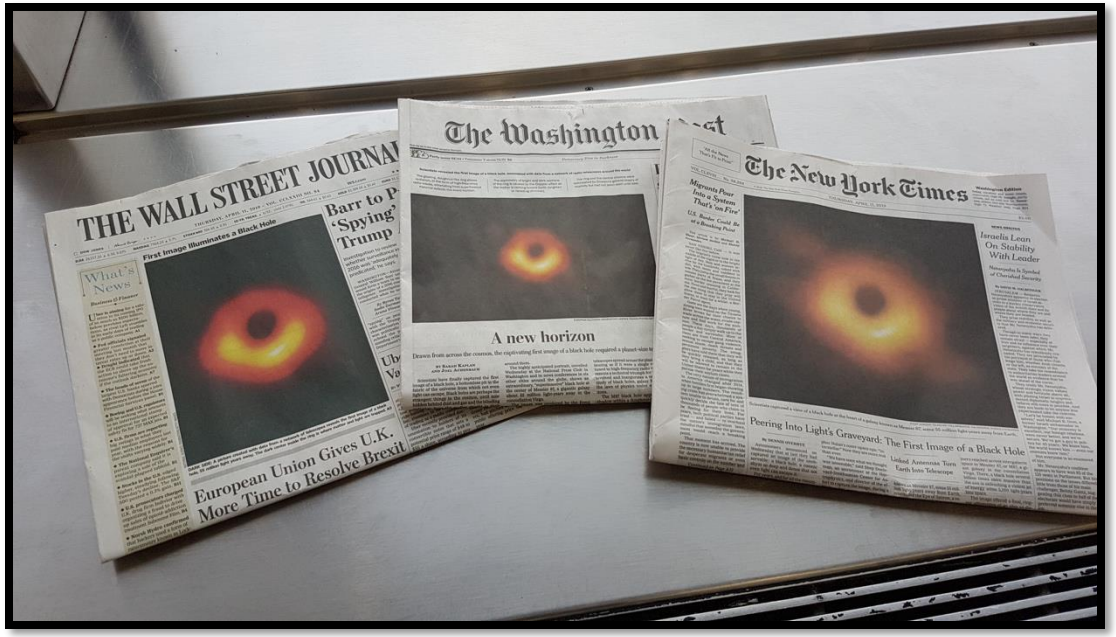
- User's view of system and the UX it provides
- May include physical metaphors for digital interactions
- Examples:
  - Pieces on a game board
  - File folders and desks

Program model

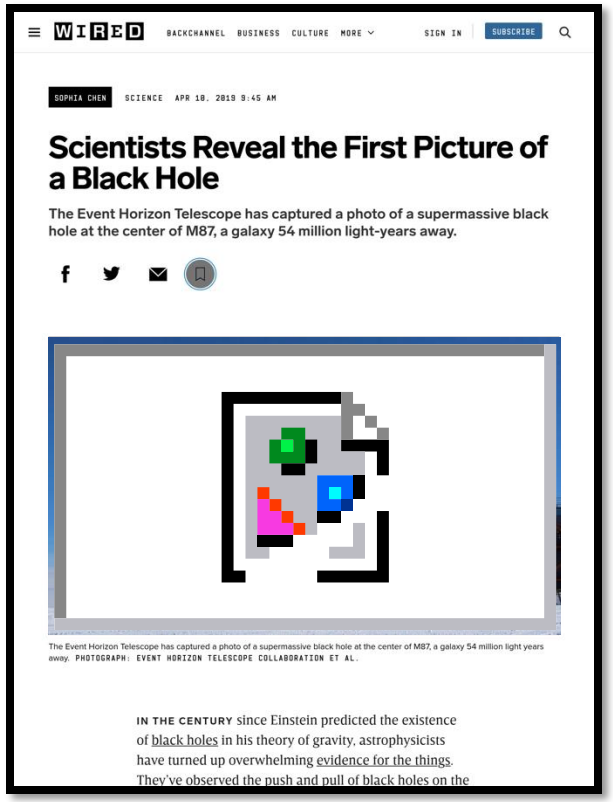
- Data, relationships, and functions making up the system
- Examples:
  - Object identity & coordinates, rules constraining movement
  - Tree of data units with metadata

Model mismatches

Mental model

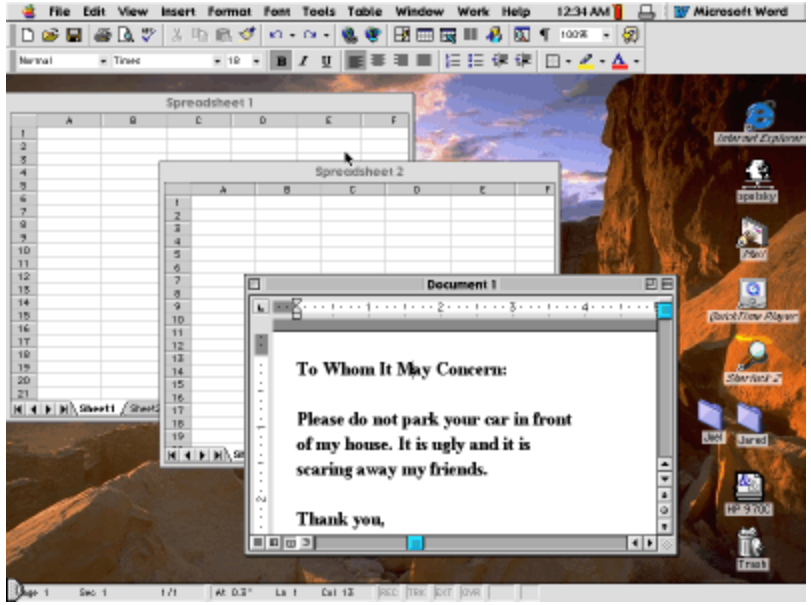


Program model



Model mismatches

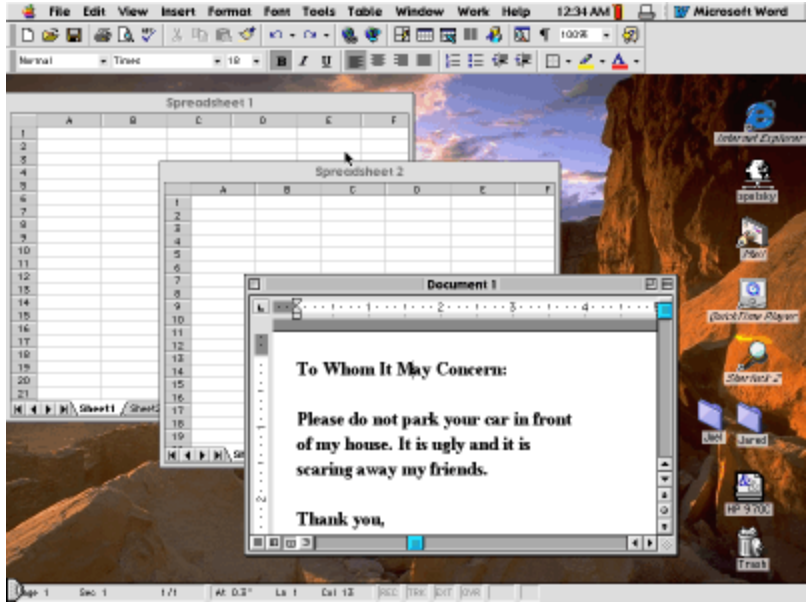
PollEv.com/cs5150sp25

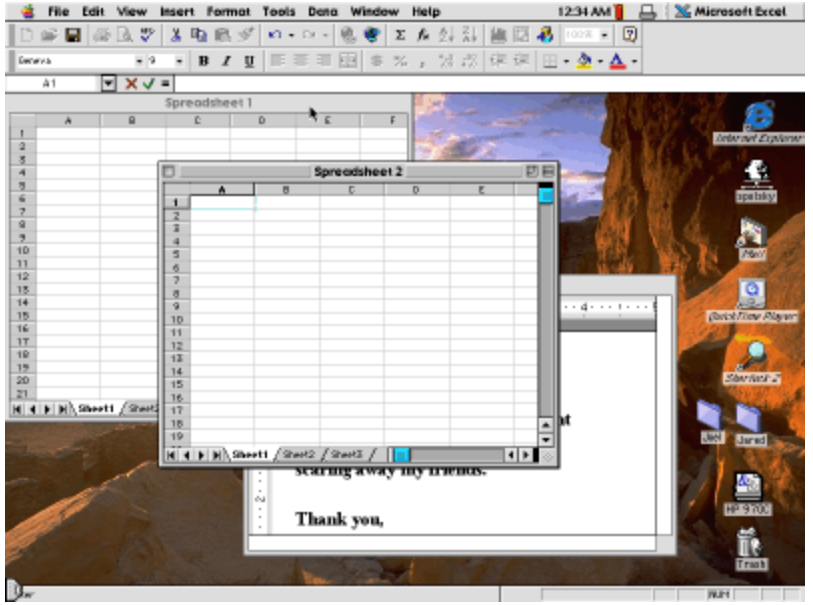


https://www.joelonsoftware.com/2000/04/11/figuring-out-what-they-expected/

Model mismatches

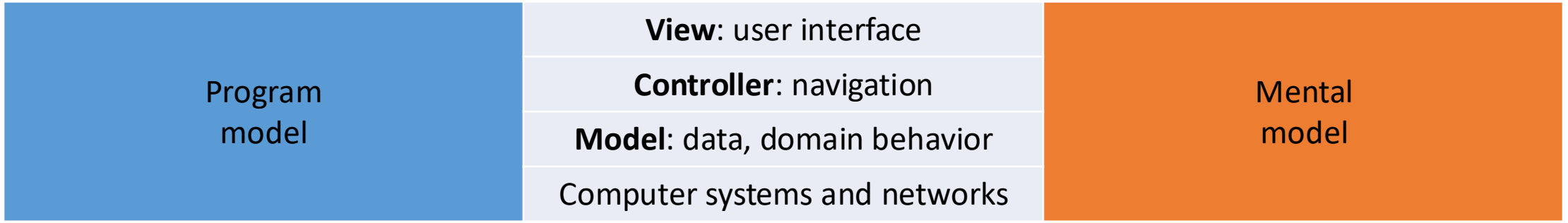
PollEv.com/cs5150sp25





https://www.joelonsoftware.com/2000/04/11/figuring-out-what-they-expected/

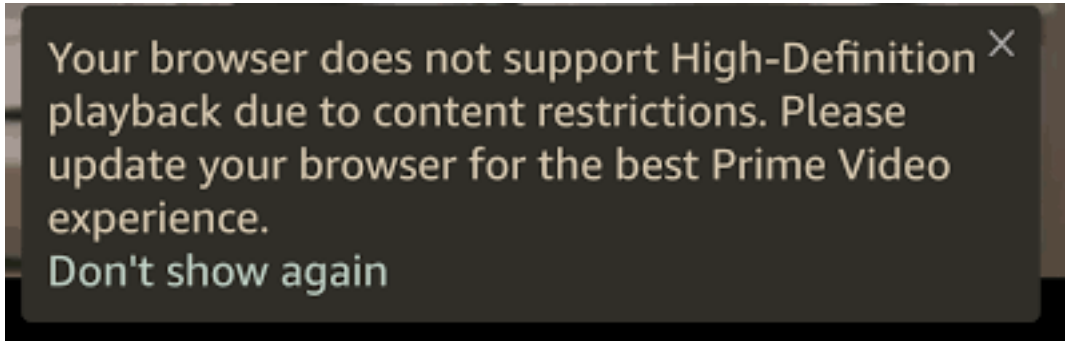
Model-view-controller (as a "model")

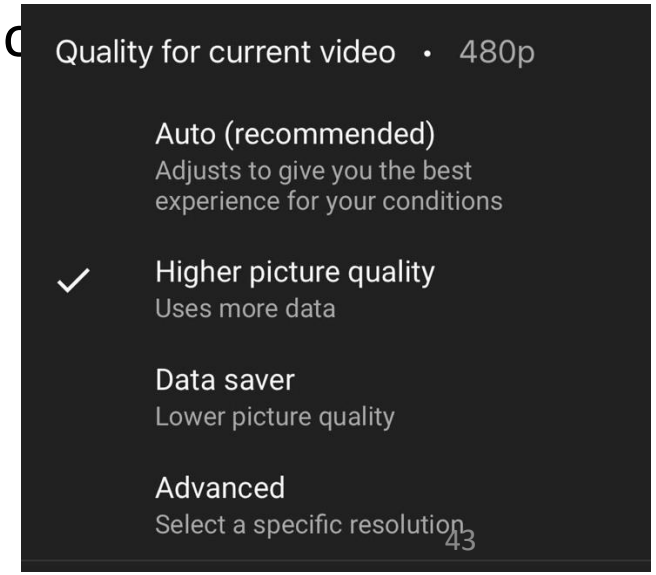


Layers correspond with most users' mental models of computing systems.

Layer 0: Computer systems and networks

- Performance, reliability, predictability of systems have a large impact on user experience
- Interfaces may be designed for specific hardware capabilities and constraints
  - Screen sizes, input devices, sensors, graphics/multimedia processing
  - Later: Adapting to constraints of web browsers and smartpho
  - Example: youtube video quality, amazon prime video



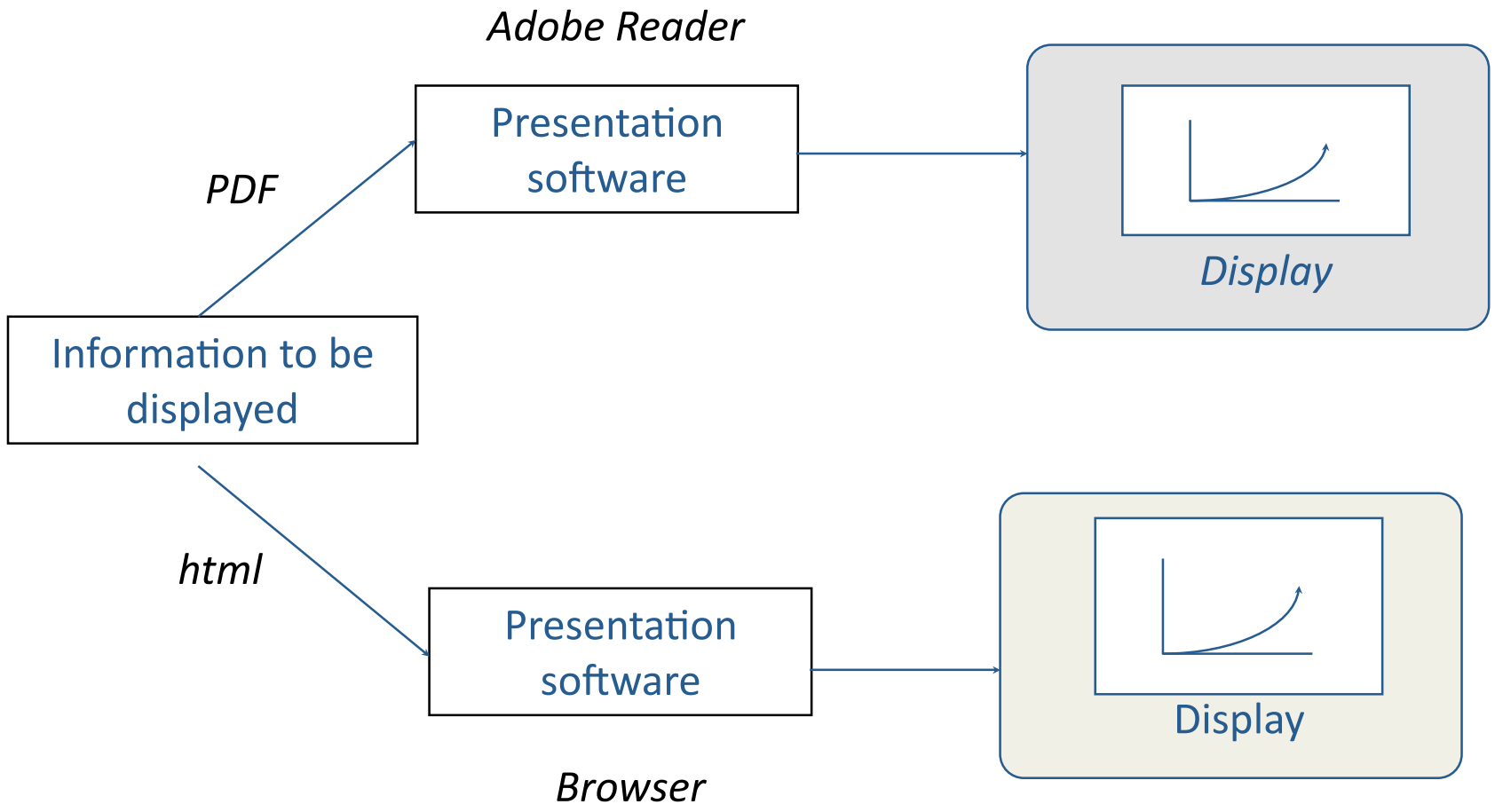


Layer 1: Model

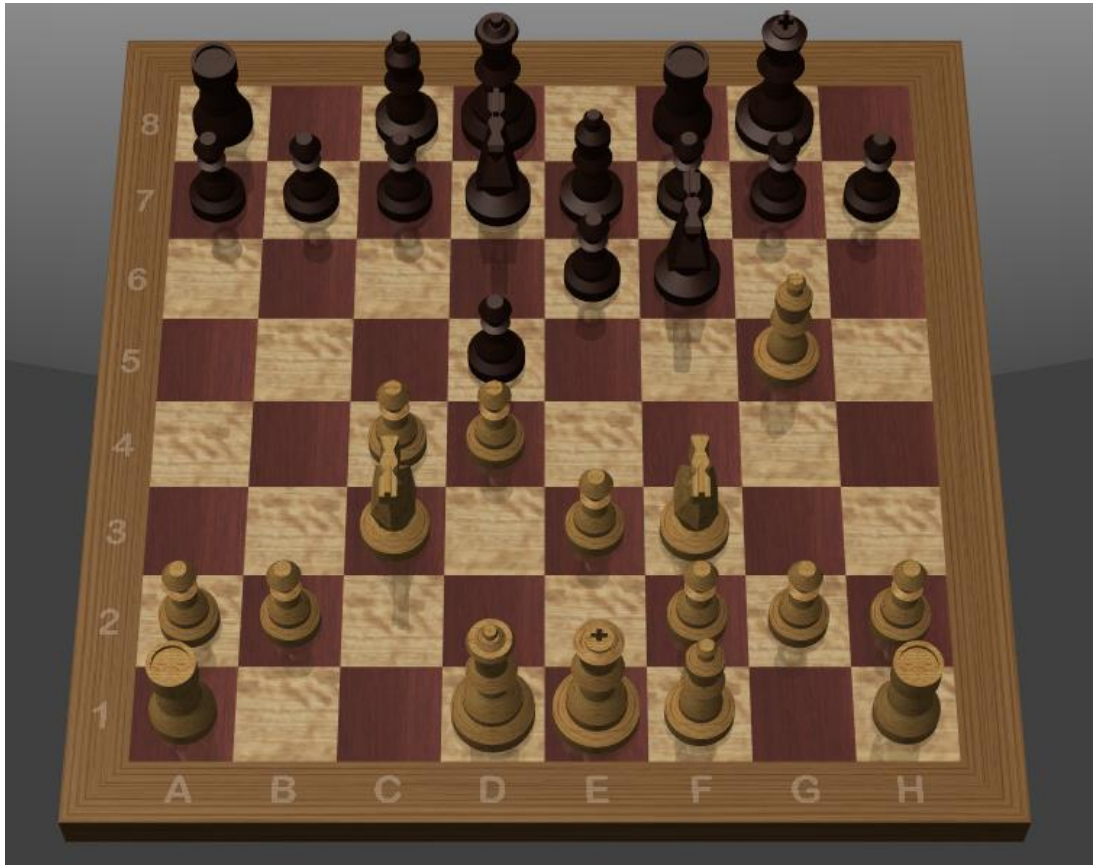
- Provides all functionality of program *except* for user interaction
  - Program logic, services
  - Data structures, file systems
  - Content (text, graphics, audio, metadata, etc.)

- Beware: easy for clients, designers to specify new behavior that is not supported by existing model

Separation of content from view



Separation of content from view



William Arms

—

Auto-Match Player

Edit Game Info...

1.	d2 - d4	d7 - d5
2.	c2 - c4	e7 - e6
3.	Nb1 - c3	Ng8 - f6
4.	Bc1 - g5	Bf8 - e7
5.	e2 - e3	0 - 0
6.	Ng1 - f3	Nb8 - d7

Layer 2: Control (navigation)

- Controller manages flow of application
  - Controls navigation between various "displays"
    - Web pages, window forms, pop-up dialogs, app screens
  - Updates model, view in response to user interaction

- Controller role varies between implementations

Layer 3: View (user interface)

- Appearance of displays and facilities for interaction
  - Graphical elements (fonts, colors, icons, images, animations)
  - Control widgets (text boxes, menus, buttons, sliders)
  - User input (touchscreen, gamepad, keyboard & mouse, buttons & knobs)

- For a quality user interface, teams need someone skilled in graphic design

Design principles

UI design principles

- UI design is partly an art, but some general principles apply:
  - Consistency (in appearance, interaction, function)
  - Feedback (what is the system doing? why does the user see what they do? what is about to happen?)
  - Ability to interrupt or reverse actions
  - Comprehensible and non-destructive error handling
- The user should feel in control (not like they're being controlled)

Example considerations: navigation menus

Advantages

- Easy for users to learn and use
- Avoids certain categories of error

Challenges

- How to handle large number of choices?
  - Scrolling menu (e.g. lists of countries or states)
  - Hierarchical
  - Filtered based on typing
- Users typically prefer menu systems that are broad and shallow (rather than deep)

Design choices: text vs. graphics

Text

- Precise, unambiguous (hopefully)
- Fast to compute, transmit

Graphics

- Quick to comprehend, learn
  - But icons may be difficult to recognize
- Variations can show different cases

Command line interfaces

- Limitations of GUIs
  - Only suitable for human users (difficult to automate)
  - Awkward to control complex interactions (difficult to compose)

Internal projects

- Gerrit: Use Git CLI to create, update reviews
- Review Board: Use `rbt` CLI to create, update reviews

- Command line interfaces (CLI)
  - User interacts with system by typing commands
  - Composable
  - Scriptable
  - Can be adapted for users with disabilities
  - Amenable to formal specification
  - Usually requires learning or training

Web and mobile interfaces

Device-aware interfaces

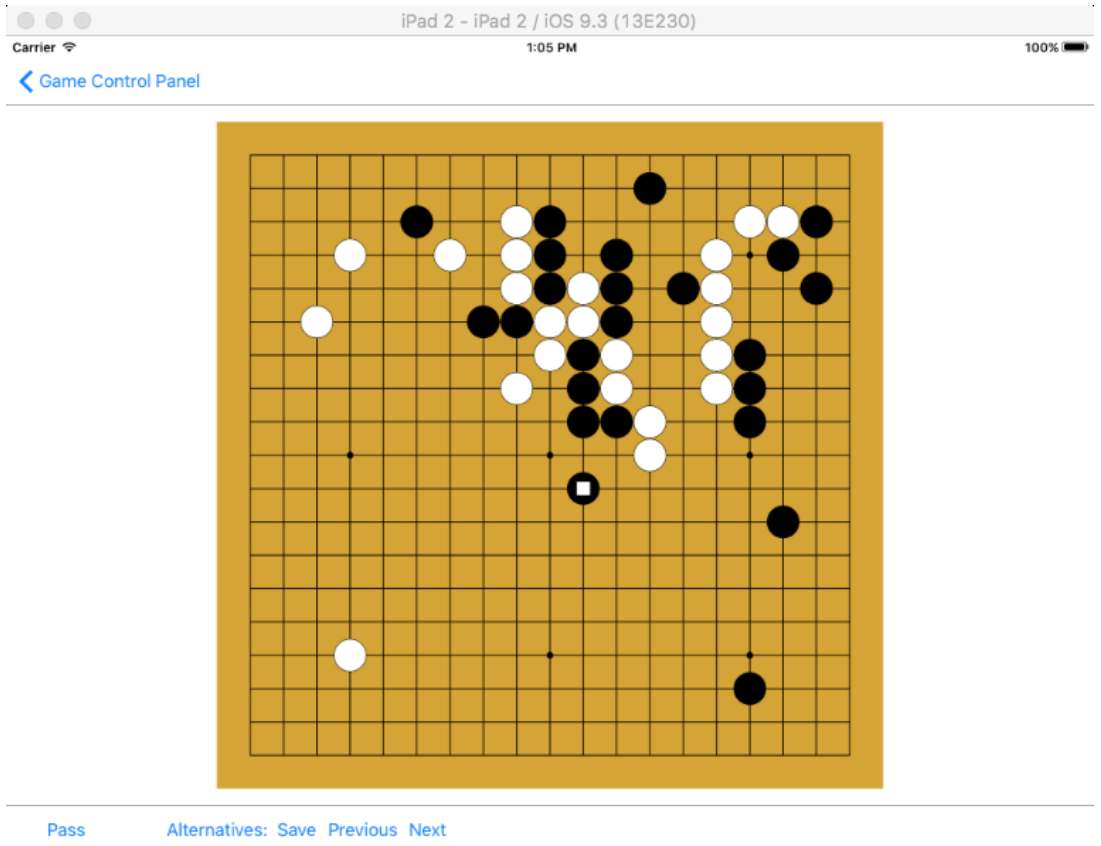
- How does a laptop computer differ from a desktop?
- What is special about a smartphone?

Web and mobile apps

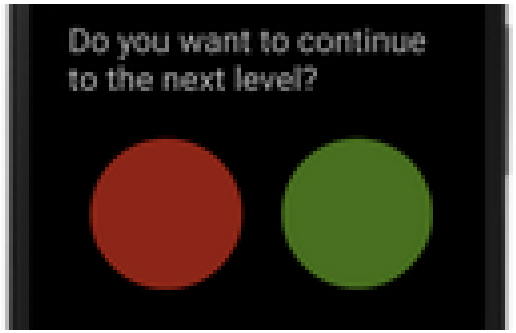
- Must consider network
  - Transfers may need to be asynchronous to hide latency
    - Need visual feedback that operation is in progress
    - Should support cancellation
  - Connections may be unreliable
    - Should be robust to duplication

Leverage simulation

- App development environments (e.g. Xcode, Android Studio) allow you to simulate screen sizes, touch events
- Web browser developer tools allow you to simulate screen sizes, network speed



Test for accessibility



Responsive design

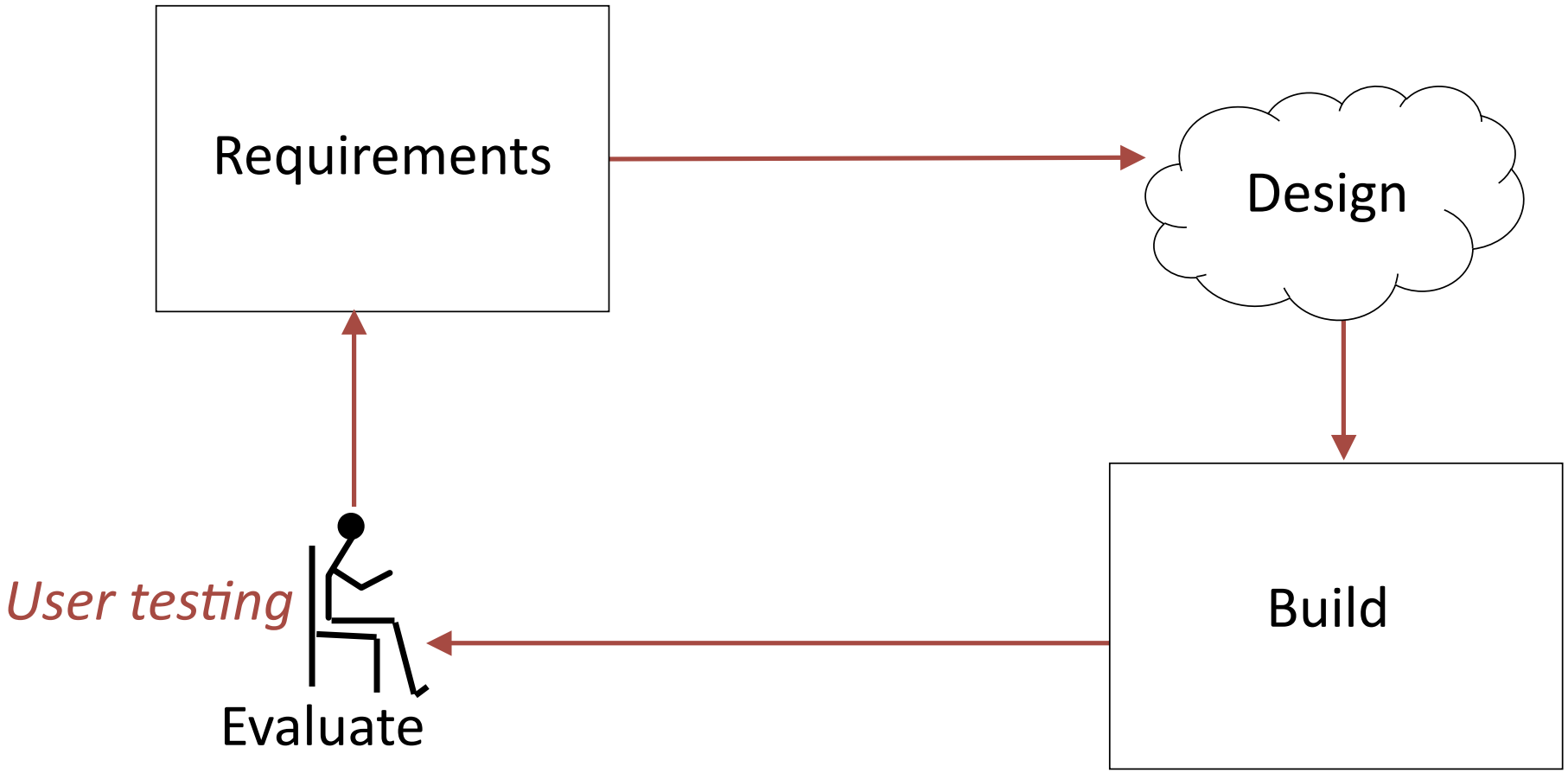
- Automatically adjust user interface based on size of screen (or other device properties)
  - Beyond simple layout scaling – can completely change layout to accommodate device
  - Use CSS media queries to select different style rules in different situations

Poll: Progressive enhancement

PollEv.com/cs5150sp25

Evaluation and user testing

Analyze/design/build/evaluate loop



Evaluation

- Design and evaluation should be done by different people
- Schedule must include time to conduct tests *and* make changes
- Evaluation should be ongoing
  - Iterative refinements during development
  - Quality assurance before deployment
  - Improvements after launch
- Methods of evaluation
  - Empirical (user testing)
  - Quantitative (measurements on operational systems)
  - Analytical (sans users; not in CS 5150)

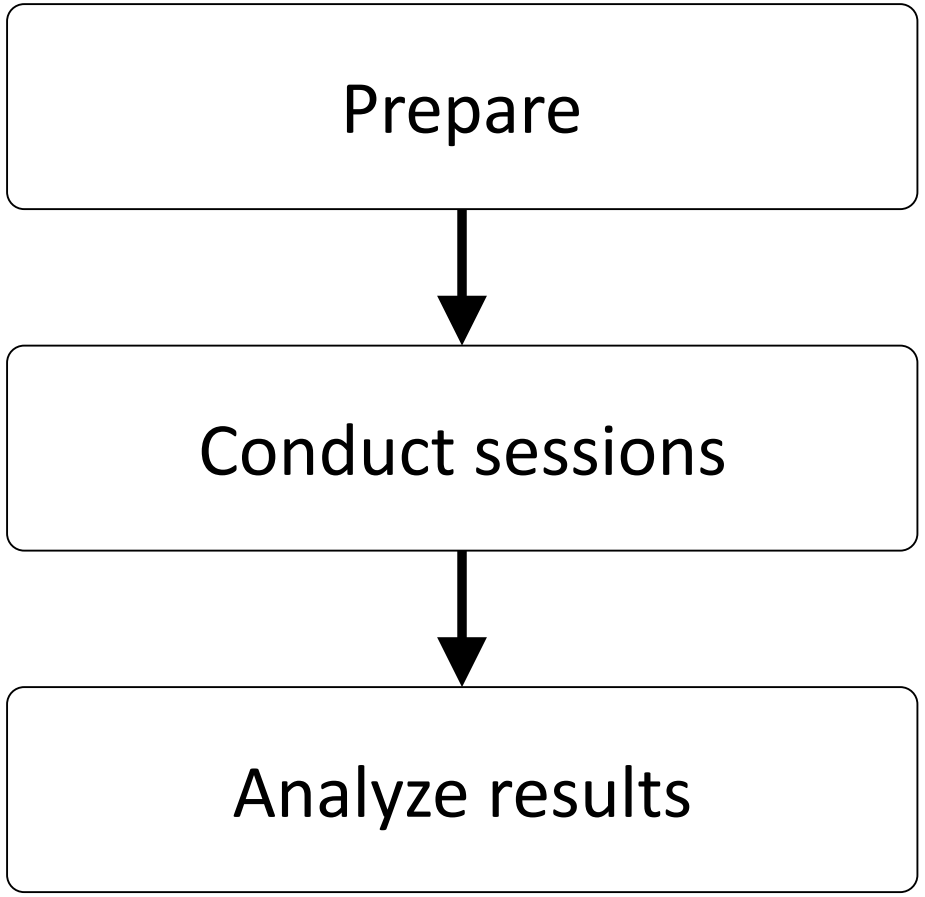
Standards for usability: ISO 9241:11

- Effectiveness
  - The accuracy and completeness with which users achieve certain goals
  - Measures: quality of solution, error rates
- Efficiency
  - The relationship between the effectiveness and the resources expended in achieving them
  - Measures: task completion time, learning time, number of clicks
- Satisfaction
  - The users' comfort with and positive attitudes towards the use of the system
  - Measures: attitude rating scales

Poll: Measuring usability

PollEv.com/cs5150sp25

User testing stages



- User testing is time-consuming, expensive, and *critical*

Preparation

- Determine goals of usability testing
  - *"Can a user find the required information in no more than two minutes?"*
- Write the user tasks
  - *"Given a new customer application form, add a new customer to the customer database"*
- Recruit participants
  - Use the descriptions of users from the requirements phase to determine categories of potential users and user tasks

Participants

- Don't need many (per feature)
  - Diminishing returns after 5-6 users
  - Look for diversity (age, experience, ability)
- Combine structured tests with free-form interviews
- Have at least two evaluators per test
  - Should *not* include designers
- Advice: it's not a race!
  - Example: user testing for arXiv

Conducting sessions

- Environment
  - Informal
  - Simulated work environment
  - Usability lab
- Give the user their task
- Observe the user
  - Human observer(s)
  - Recording (with permission)
- Query satisfaction

Analyzing results

- Test the system, not the users
  - Respect the data and the user's responses
  - Do not make excuses for designs that failed
  - If possible, use statistical summaries
- Pay close attention to instances where users:
  - Were frustrated
  - Took a long time
  - Could not complete tasks
- Also note aspects of the design that *did* work
  - Ensures they are maintained / do not regress in final product

Example: Past CS 5150 methodology

How we're user testing:

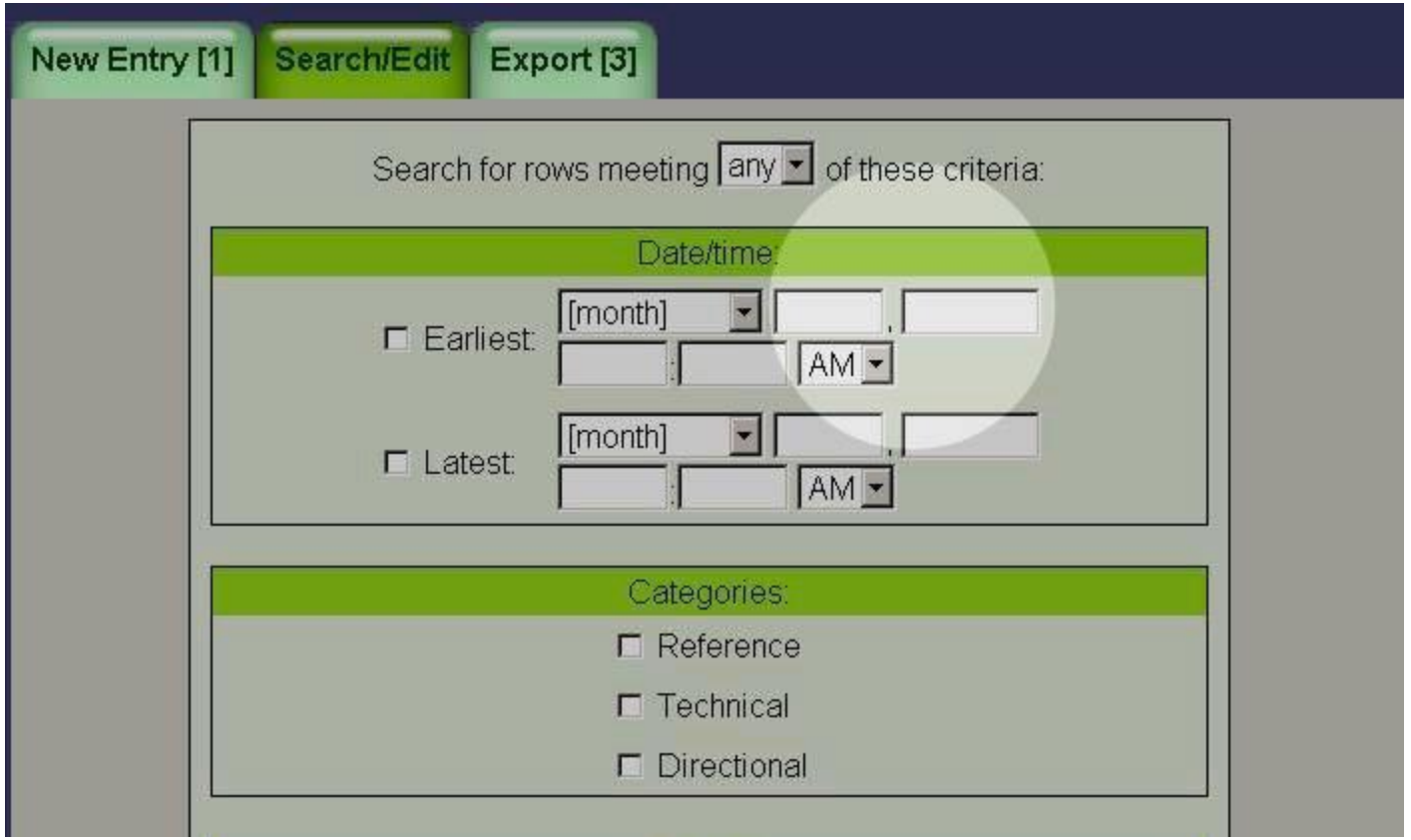
- One-on-one, 30-45 min user tests with staff levels
- Specific tasks to complete
- No prior demonstration or training
- Pre-planned questions designed to stimulate feedback
- Emphasis on testing system, not the stakeholder!
- Standardized tasks / questions among all testers

Example

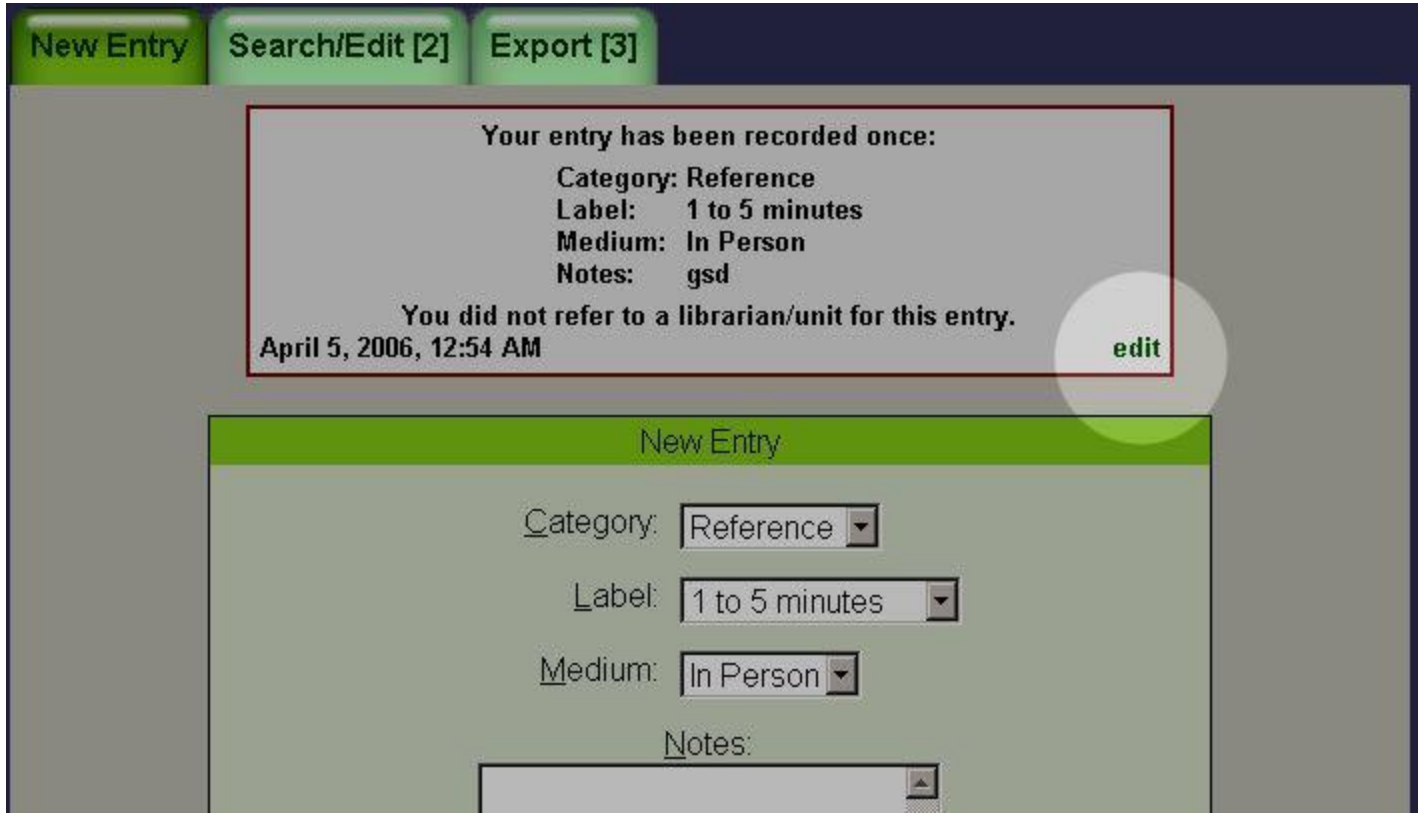
Types of questions we asked:

- Which labels, keywords were confusing?
- What was the hardest task?
- What did you like, that should not be changed?
- If you were us, what would you change?
- How does this system compare to your paper based system
- How useful do you find the new report layout? (admin)
- Do you have any other comments or questions about the system? (open ended)

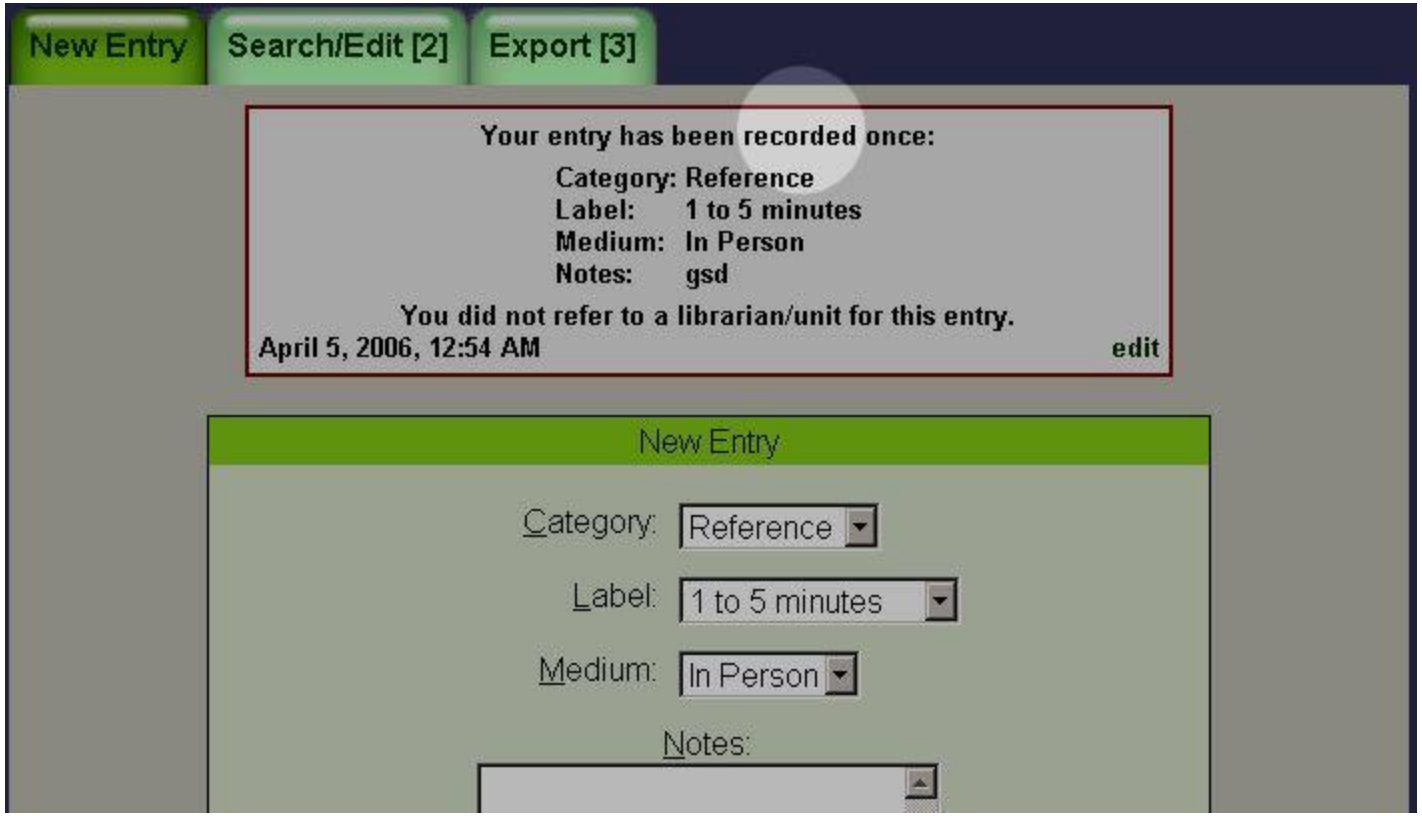
What we've found:
Issue #1, Search Form Confusion!



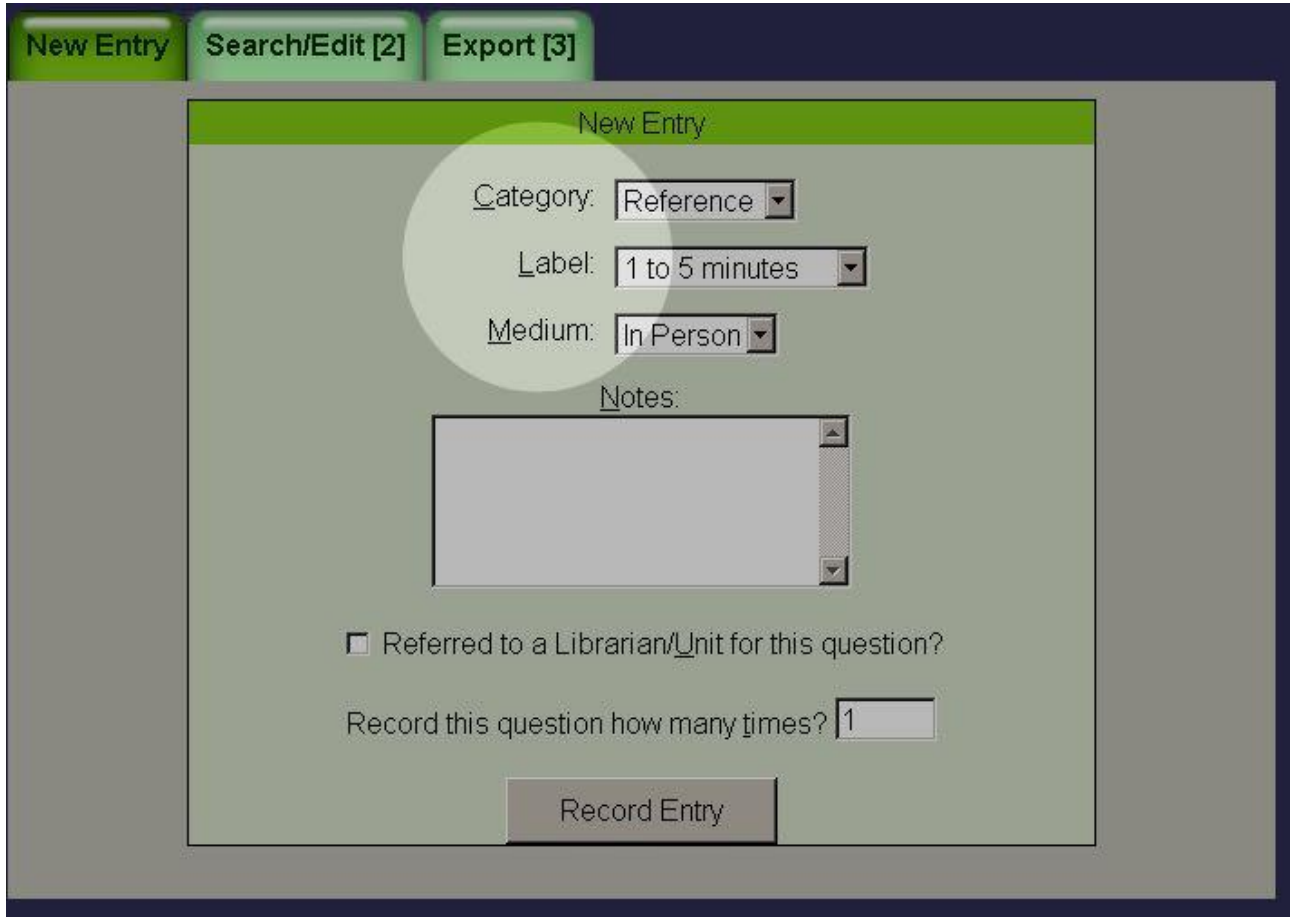
What we've found:
Issue #2, Inconspicuous Edit/Confirmations!



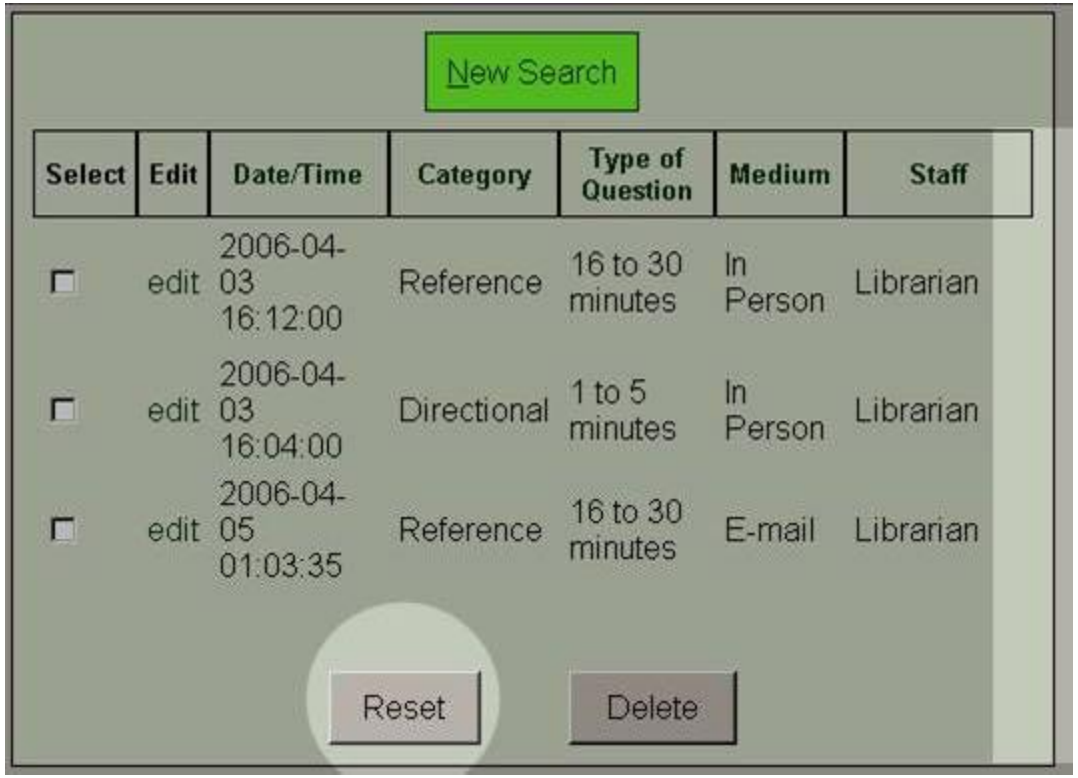
What we've found:
Issue #3, Confirmation Terms



What we've found:
Issue #4, Entry Semantics



What we've found: Issue #5, Search Results Disambiguation & Semantics

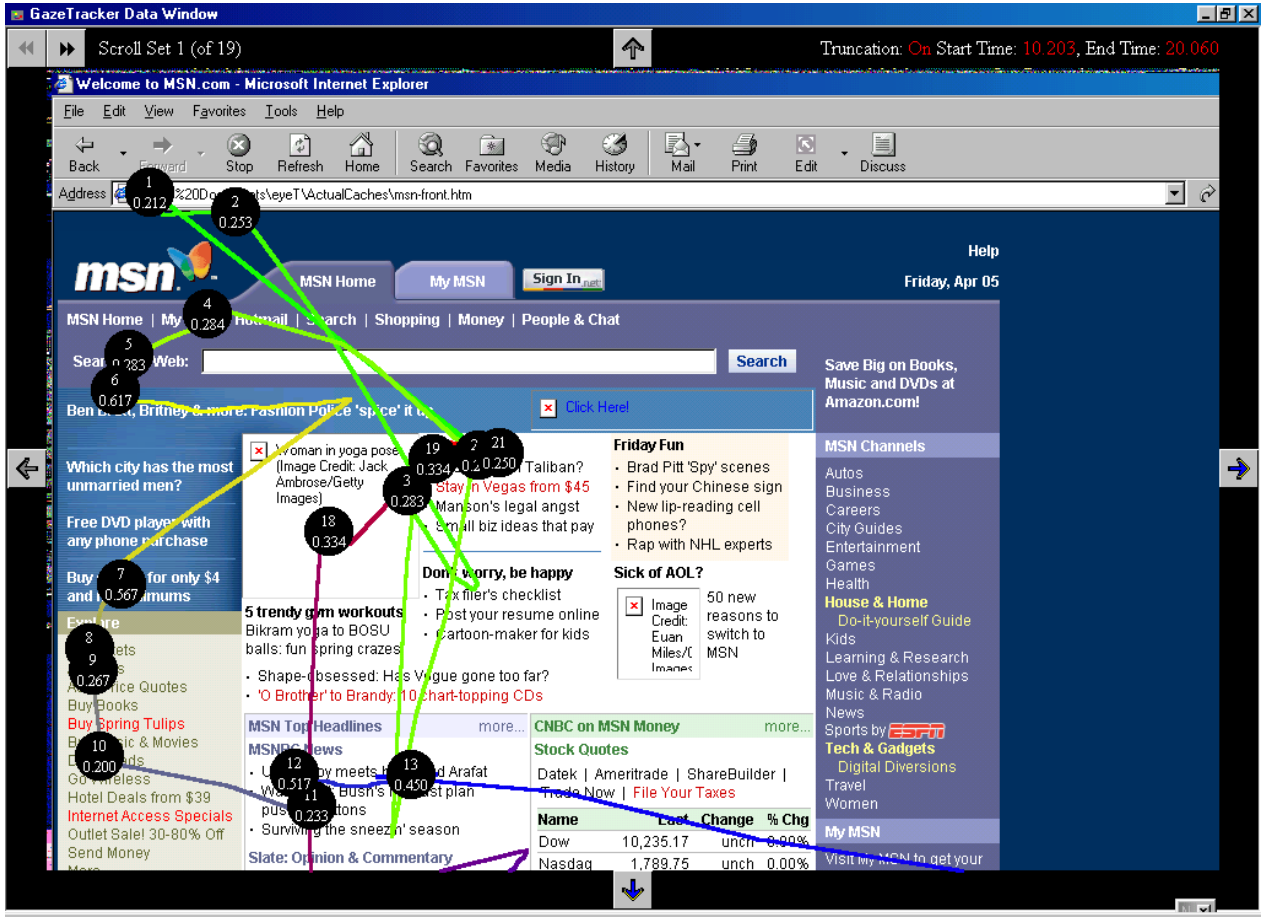


Measurement-based evaluations

- User testing can be done with (non-functional) prototypes
  - Requires more interaction with evaluator (risk of bias)
- Measurements require an operational system

- Log events in users' interactions with system
  - Clicks (when, where)
  - Navigation (from page to page)
  - Keystrokes
  - Use of help system
  - Errors encountered
  - Eye tracking
- May be used for statistical analysis or for detailed study of an individual user

Eye tracking



Analyzing measurements

- Which interface options were used?
- When was the help system consulted?
- What errors occurred? From where and how often?
- Which links were followed? (clickthrough data)

- Human feedback (less structured)
  - Complaints and praise in feedback forms
  - Bug reports
  - Calls to customer service

Refining designs

- Do not allow test evaluators to become designers
  - Designers are poor evaluators of their own work,
  - But designers know requirements, constraints, context of design
    - Know which problems might be addressed with small changes
    - Know which problems require major changes that should be escalated
    - Know which user requests are mutually incompatible
      - Balance between configurability and simplicity (designer's job)
- Designers and evaluators must work as a team
  - But not try to do each other's work

User testing in CS 5150

- All projects must conduct user testing of user interfaces you design
  - Internal projects: recruit classmates from other teams
  - Decide how much training users should have
    - They should probably be familiar with existing system
    - You can provide training (but don't "teach to the test"), or a user manual
  - Design tasks & metrics
    - "Which files has your reviewer read so far?"
    - "Which, if any, of your commit messages has your reviewer left a comment on?"
    - "Add a reviewer comment to this file that was not modified"
  - Design survey

Code tracing

Techniques

- Monitor application logs
- Developer tools network view
  - Look for mutating methods (POST, PUT, DELETE, vs. GET); ignore static resources
  - Look at initiator stack trace
    - Ignore framework methods (jQuery, etc.)
    - Look for promising files, then read them
- Search source code
  - Filter results (ignore static, tests, docs)