CS 280

Risk and Reward in the Information Society

HCI2 - Design

User interface design

- Stakeholders
- Requirements Analysis
- Prototyping
- Evaluation
- Documentation

Requirements Analysis

- Analyze the needs of clients with respect to a proposed project or technology
  - But: Clients often cannot appreciate their real needs until they see what options are available
- Work practices evolve as new technology is introduced, and requirements may evolve for many reasons
- study how work currently takes place to identify problems and opportunities that might be addressed by new technology

Requirements Analysis

- Not just what the user says they need. Consider
  - The ***activities*** of the workplace
  - The ***artifacts*** of the workplace
  - The ***social context*** of the workplace
- When the user asks for a specific product, what are the activities, artifacts, and social context of that request
  - What tacit knowledge might be assumed?
  - What might the user or management expect to be familiar?
  - What training is expected? Is it appropriate?

Stakeholders

- A stakeholder is a person impacted by the system
- multiple stakeholders:
  - Managers; Executives; Workers, Clients, Users
  - Other Employees impacted by the result
  - People outside of the organization
- Each will describe a task differently. Workers often describe typical (or even “official”) versions of a task
- Each will have different implicit and explicit motivations

Stakeholder metrics

- Know your audience.
  - Accountants want “install base” “average daily users”
  - Social directors want “engagement” “session length”
  - Ad executives want “click-through rate”
  - Users want “satisfaction”
  - Developers want “to go home and see my family”

Sources of Task-related Knowledge

- Stakeholders may have:
  - organizational knowledge (explicit)
    - policies and procedures
    - work-flow diagrams
    - instructions from superiors
  - activity knowledge (tacit)
    - unofficial procedures
    - task-specific information
    - Instructions from colleagues

Sources of Knowledge

	Organizational (Explicit) Knowledge	Activity-Oriented (Tacit) Knowledge
Acquisition	training	learning
Action	tasks	know-how
Rank	position in hierarchy	politics, network of contacts
Resourve	procedures and techniques	conceptual understanding
Process	work flow	work practices
Decision	methods and procedures	rules of thumb, judgment
Support	teams	communities
Authority	superior	colleague

Who is responsible for Interactions

- ***People who use software should learn how to use that software***
  - Onus on the user to learn the software
  - Developer is not responsible for interaction?

- ***Software that people use should be learnable by people***
  - Onus on the developer to build learnable software
  - User is not responsible for interaction?

Operators, Users, and Developers

- operator error = system failure
  - you can't attribute a problem to "human error"
  - Operators are part of the system
  - Humane interfaces take operator behavior into account.
  - Operator error is failure of the system designers, not the operator

Beginners versus Experts

- beginners need simplicity, clarity of function, and visibility
- experts need aptness to task, modelessness, and monotony (just one way to accomplish a task)
- People might be at either stage with respect to any one feature

Design

- A structure for good design:
  - ***Understand***: examine the context of relevant technologies, competitors, and potential markets
  - ***Observe***: see what users do, what they are used to, and what is important to them
  - ***Visualize and Predict***: imagine who will be using the products, through scenarios
  - ***Build***, ***Evaluate and Refine***: iteratively fill in details and evaluate the results as you go. Test usability feature by feature and as a whole

Design: Prototypes

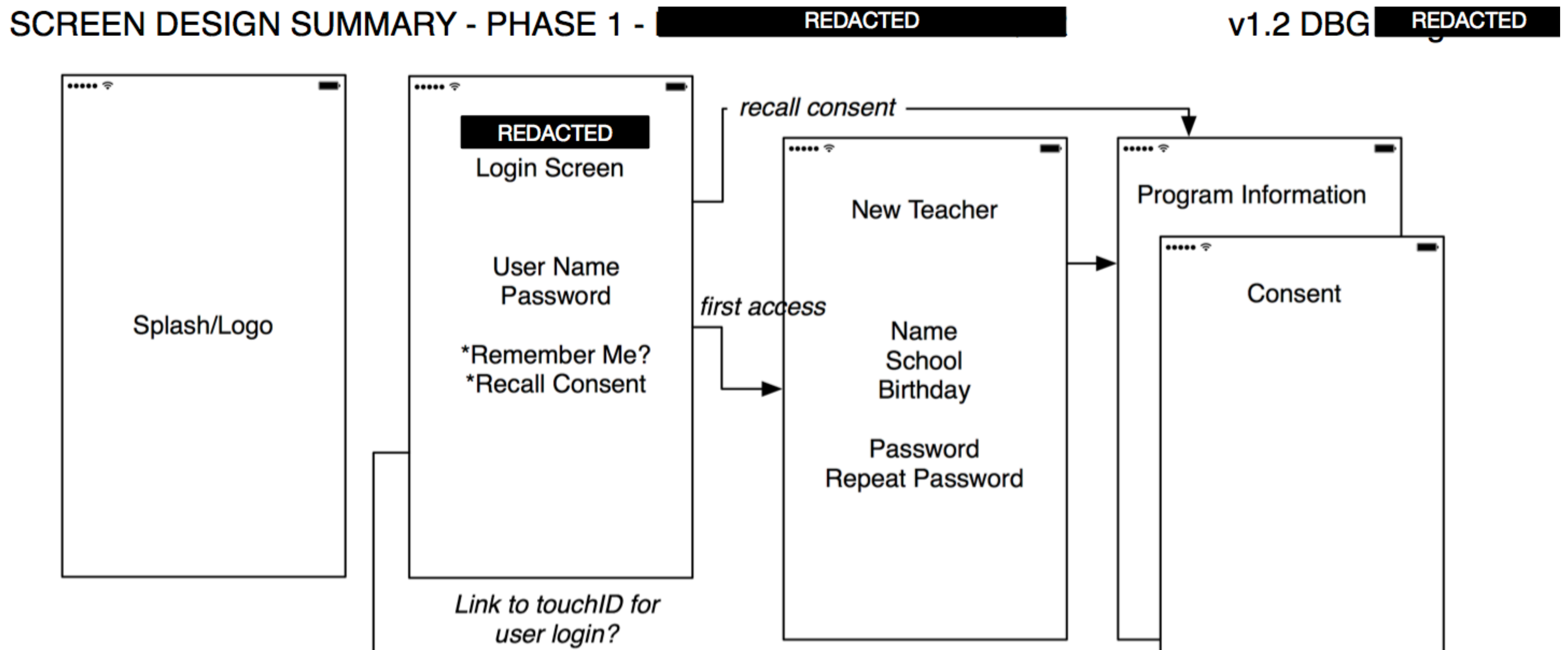
- A ***concrete***, ***partial*** implementation of a system design
  - useful in software development goals:
    - Refine user requirements
    - Explore design ideas
    - Share or co-develop designs with users
    - Test specific issues with stakeholders
- When choosing a prototype technique, consider:
  - cost and effort required to build, and available resources
  - ***fidelity*** and ***generality*** of the resulting prototype
  - goals of the prototype

Prototype Techniques

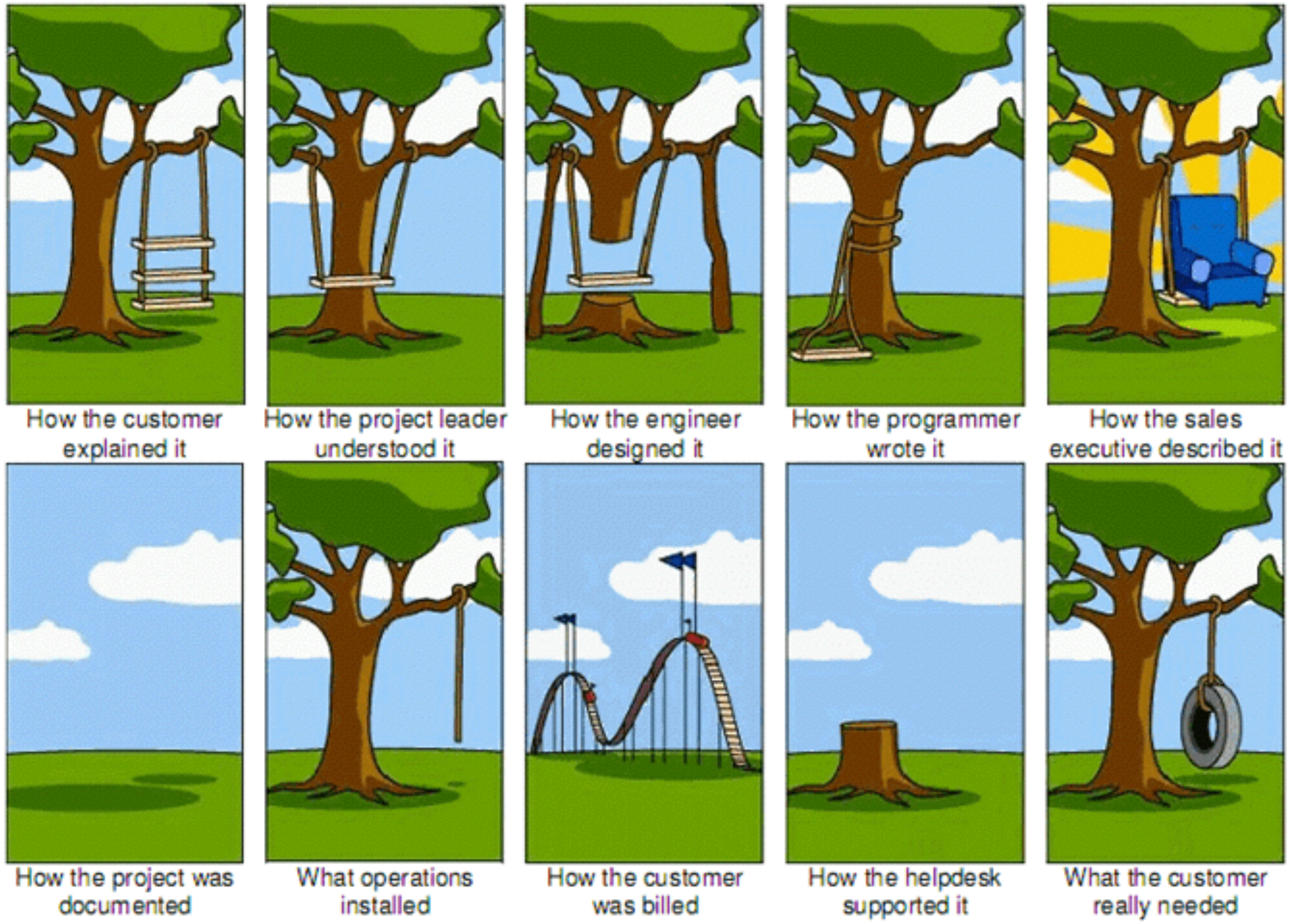
Storyboard	Sketches / screenshots with transitions indicated
cardboard mock-up	Fabricated device with simulated controls / display
"Wizard of Oz"	Hidden human assistant simulates functionality
Video Prototype	Video recording of persons enacting envisioned tasks
Computer Animation	Animated screens illustrate input and output events
Scenario Machine	Interactive mockup implementing one scenario
Rapid Prototype	Interactive mockup created with prototyping tools
Working Partial System	System with a subset of intended functionality

Rough prototypes

- Avoid photorealistic sketches - easy to produce in drawing programs but can impart a false sense of polish



Prototyping and stakeholders


How the customer explained it
How the project leader understood it
How the engineer designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

- Underpromise, overdeliver.
- Don’t prototype something you can’t build for real.

Evaluation

- Assess a prototype (or system) against requirements
  - Software engineering model will dictate frequency and depth of evaluation at each stage of development
- ***Formative*** evaluation: during design
- ***Summative*** evaluation: after production

- ***Analytical*** evaluation: using theoretical models
  - ie. without users.
- ***Empirical*** evaluation: using experiments
  - ie. with users

Analytical measures

- Evaluation by user-interface experts
  - multiple experts, should have rubrics they follow.
- Model-based analysis, e.g. KLM-GOMS (as before)
  - Keyboard-level model; keyboard, mouse, other actions have predicted times.
  - Analyse an interface without needing users
  - Doesn’t take into account user frailties; assumes expert users with perfect understanding.

KLM-GOMS example

Description	**Operation**	**Time (sec)**
Reach for mouse	H[mouse]	0.40
Move pointer to "Replace" button	P[menu item]	1.10
Click on "Replace" command	K[mouse]	0.20
Home on keyboard	H[keyboard]	0.40
Specify word to be replaced	M, 4*K[word]	2.15
Reach for mouse	H[mouse]	0.40
Point to correct field	P[field]	1.10
Click on field	K[mouse]	0.20
Home on keyboard	H[keyboard]	0.40
Type new word	M, 4*K[word]	2.15
Reach for mouse	H[mouse]	0.40
Move pointer on Replace-all	P[replace-all]	1.10
Click on field	K[mouse]	0.20
Total		**10.2**

User Study

- Observe users interacting with the system
- Users can be squishy and unpredictable
- May need special permission to test on users
- University / hospital requires approval from the ***Research Ethics Board***
  - They evaluate all research involving human subjects
  - User studies are usually pretty straightforward.
- Requires good research methods design in general
  - Take a research methods course

Designing evaluation tasks (emperical)

- Relate to requirements analysis
- Consider training
  - Test before and after training
- Consider anchoring effect
  - Test old and new interfaces in different order for different people
- Consider objective and subjective measures
  - How long did it take vs how did it make you feel

User Study: Think-Aloud Protocols

- Encourage the user to narrate all observations, plans, and reactions
- Trying to make norman’s gulfs more overt
- Narration before / after a problem can expose problematic design elements
- Not a natural behaviour, for most users, may change behaviour
  - Try it while you’re driving some time.

Documentation

- Stored information about system use
  - Not code documentation

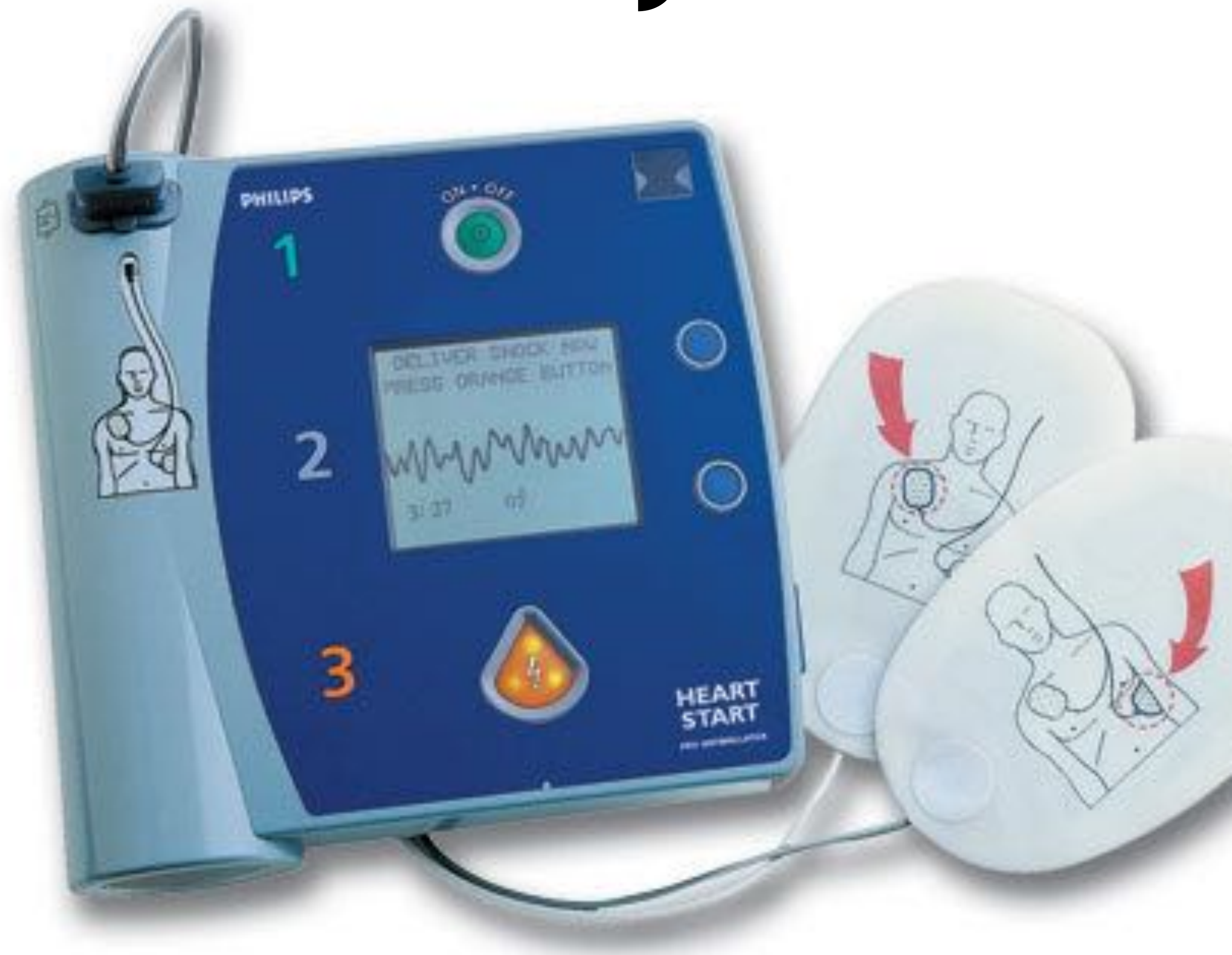
- Standalone
  - User manuals, tutorials, training, walkthroughs
- Integrated
  - Prompts, error messages, tooltips, help menus, first-run protocols

Documentation paradox

- Nobody wants to waste time reading manuals, people want to get started
  - Should we force users to read docs? If so, how?
- Nobody reads documentation. Nobody wants to write documentation. But when you need it, it had better be there and be good
- Nobody reads error messages either

Documentation Notes

- People learn in different ways. Good documentation should reflect that
  - Showing, doing etc rather than just reading
- Documentation is often translated from the language of the manufacturer, so written manuals can be very bad
- ***Walk-up-and-use*** systems must assume no documentation, and still be usable
  - Affordances, abstractions, familiarity.

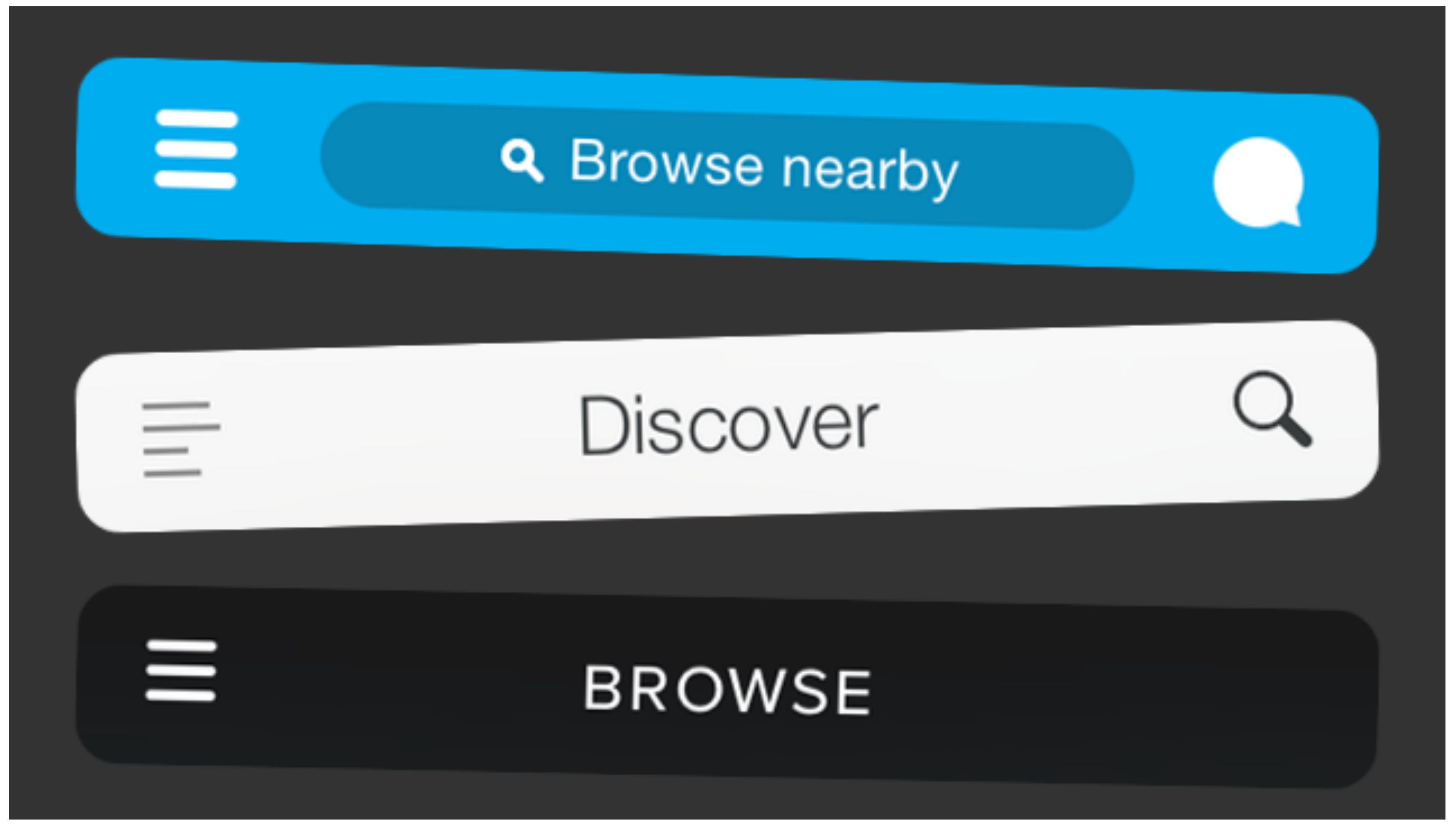


Information Architecture

- System information can be documentation, but can reinforce (or be the cause of) UI problems
  - Redesign the IA if you start to get UI problems
  - If you let the IA dictate the UI, it won't be as good as it can be

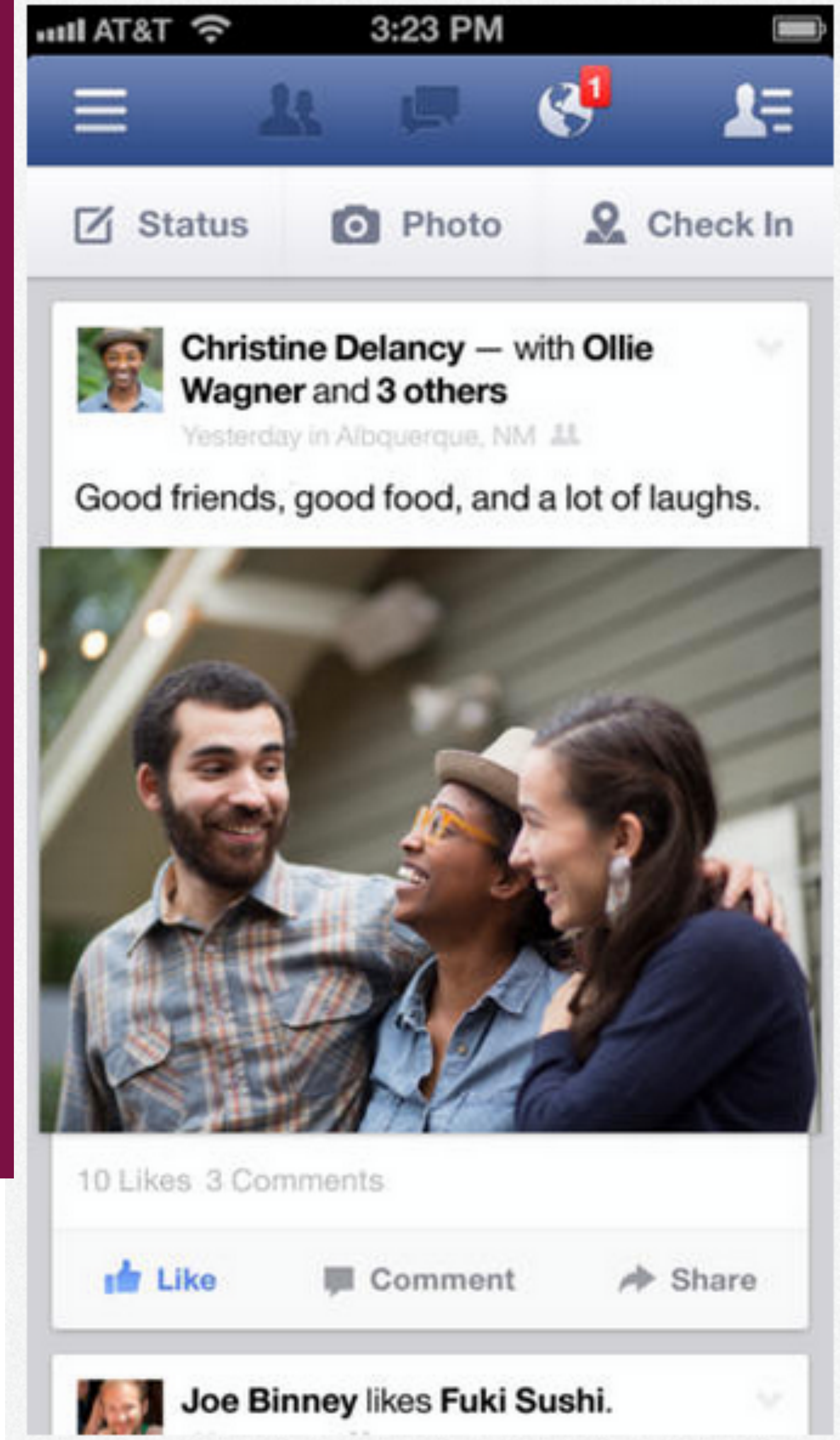
  - https://lmjabreu.com/post/why-and-how-to-avoid-hamburger-menus/
  - https://www.usability.gov/what-and-why/information-architecture.html

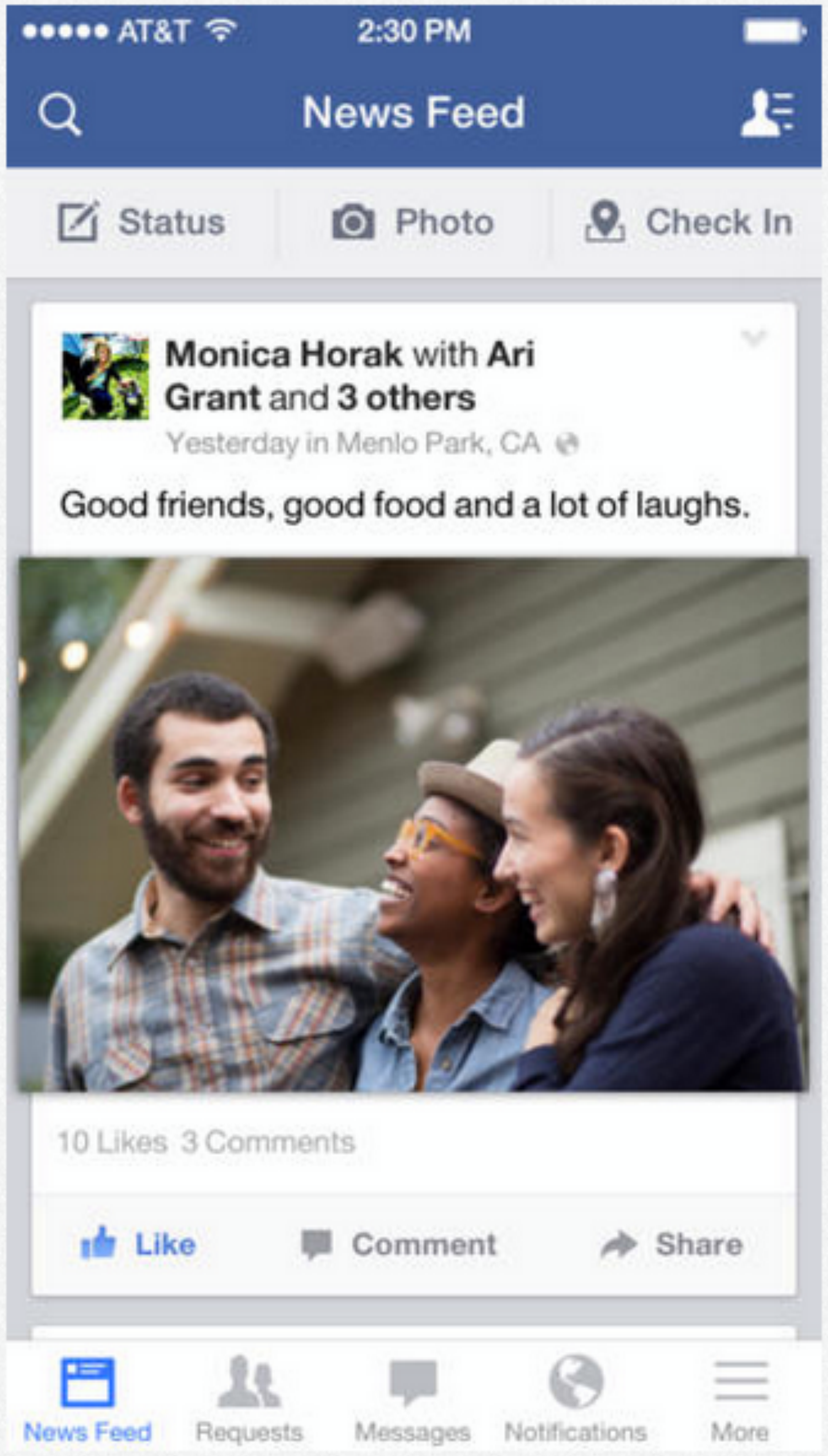
Hamburger Menus



- motivated from a style perspective, not a usability perspective
  - Clean, uncluttered "look"
- ***Hidden Information***: Users shouldn't have to take an action to see what actions they can take
  - Exercise: find an app with a hamburger menu and predict what will be shown when activated.
- ***Reduced efficiency***: Choosing an option fro the hamburger menu usually closes the menu, so every menu item now takes two touches at least

Ditching the hamburger menu in favour of a menu bar


AT&T
3:23 PM
1
Status
Photo
Check In
Christine Delancy — with Ollie Wagner and 3 others
Yesterday in Albuquerque, NM
Good friends, good food, and a lot of laughs.
10 Likes 3 Comments
Like
Comment
Share
Joe Binney likes Fuki Sushi.


AT&T
2:30 PM
News Feed
Status
Photo
Check In
Monica Horak with Ari Grant and 3 others
Yesterday in Menlo Park, CA
Good friends, good food and a lot of laughs.
10 Likes 3 Comments
Like
Comment
Share
News Feed
Requests
Messages
Notifications
More