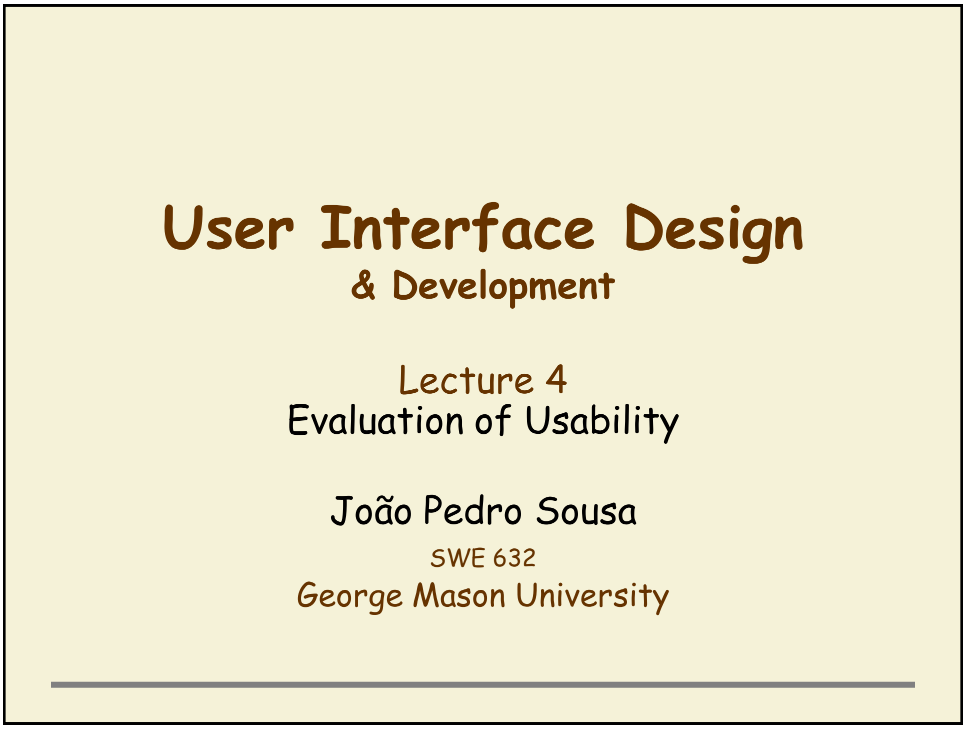
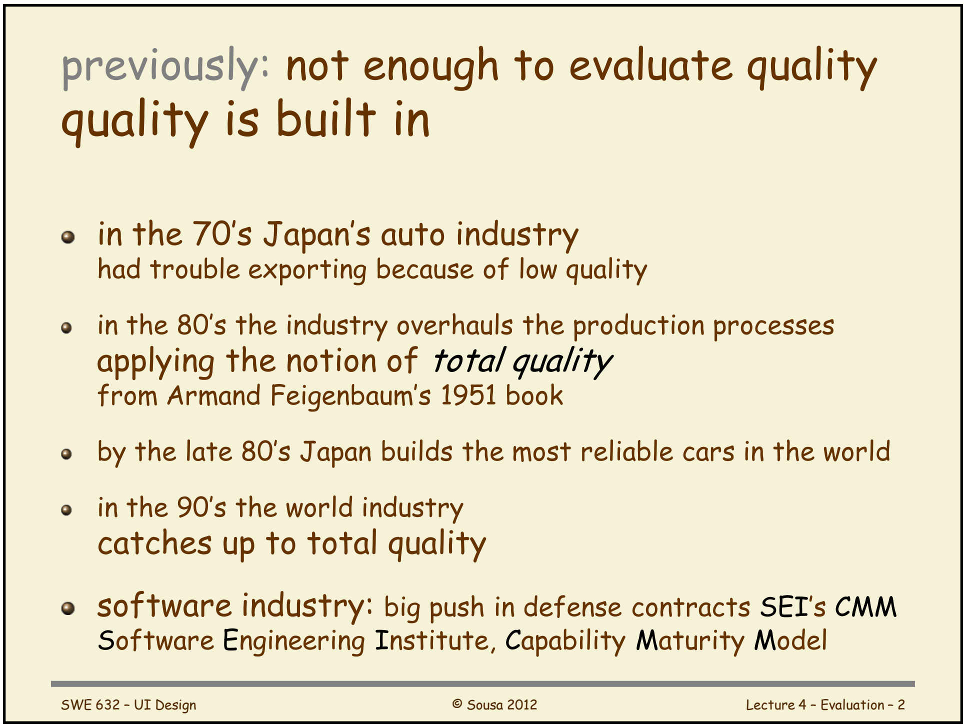


User Interface Design
& Development

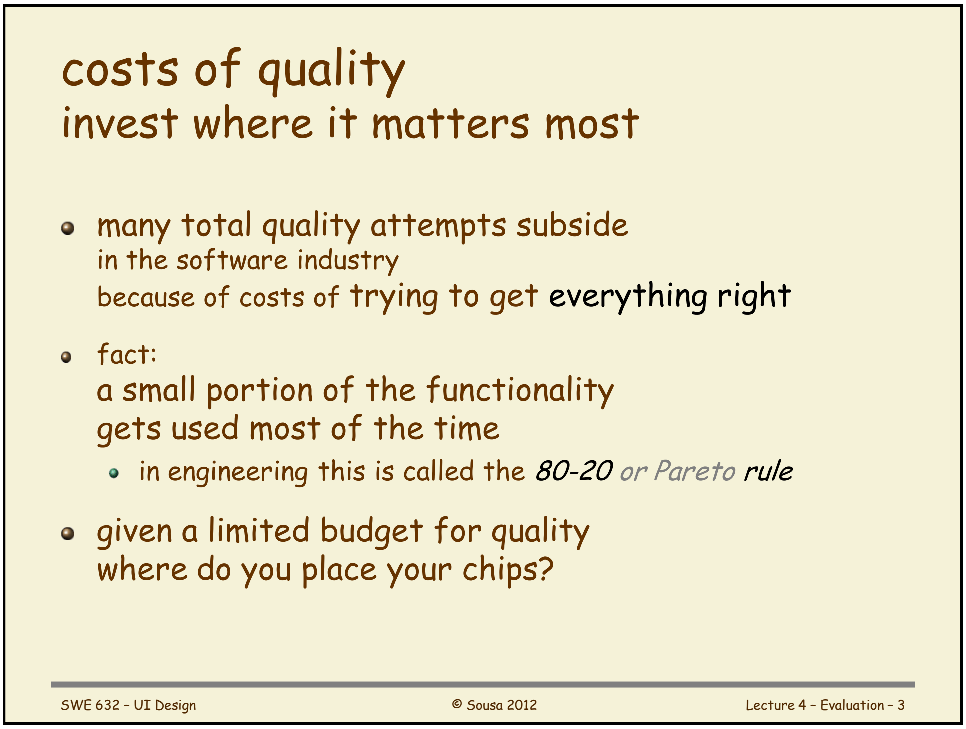
Lecture 4
Evaluation of Usability

João Pedro Sousa
SWE 632
George Mason University



previously: not enough to evaluate quality
quality is built in

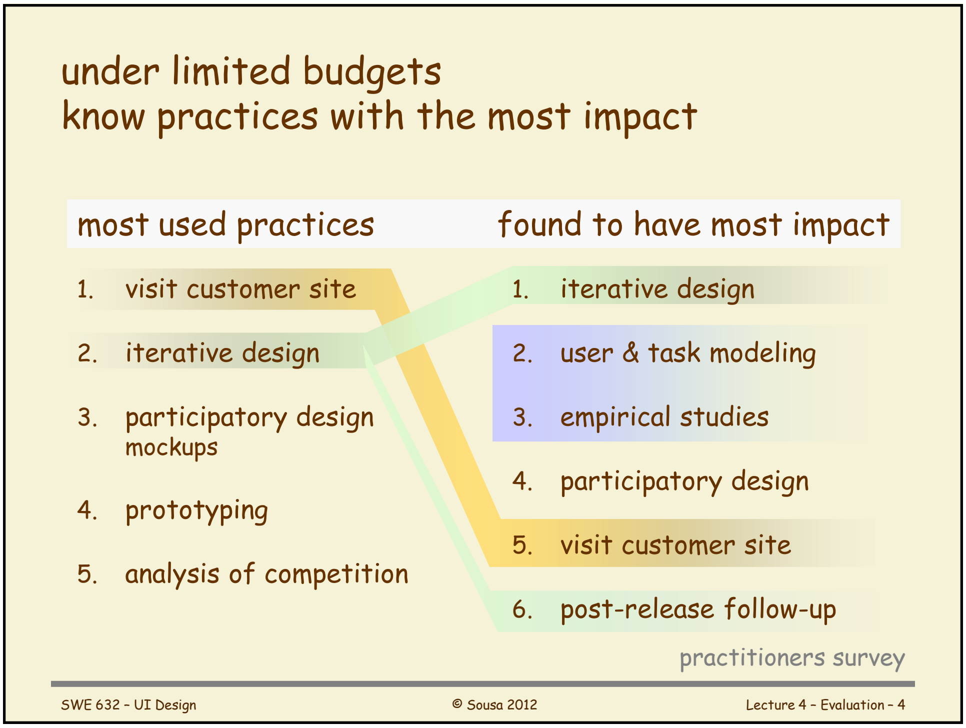
- in the 70's Japan's auto industry
  had trouble exporting because of low quality
- in the 80's the industry overhauls the production processes
  applying the notion of *total quality*
  from Armand Feigenbaum's 1951 book
- by the late 80's Japan builds the most reliable cars in the world
- in the 90's the world industry
  catches up to total quality
- software industry: big push in defense contracts SEI's CMM
  Software Engineering Institute, Capability Maturity Model



costs of quality
invest where it matters most

- many total quality attempts subside
  in the software industry
  because of costs of trying to get everything right
- fact:
  a small portion of the functionality
  gets used most of the time
  - in engineering this is called the *80-20 or Pareto rule*
- given a limited budget for quality
  where do you place your chips?

SWE 632 – UI Design © Sousa 2012 Lecture 4 – Evaluation – 3



under limited budgets
know practices with the most impact

most used practices

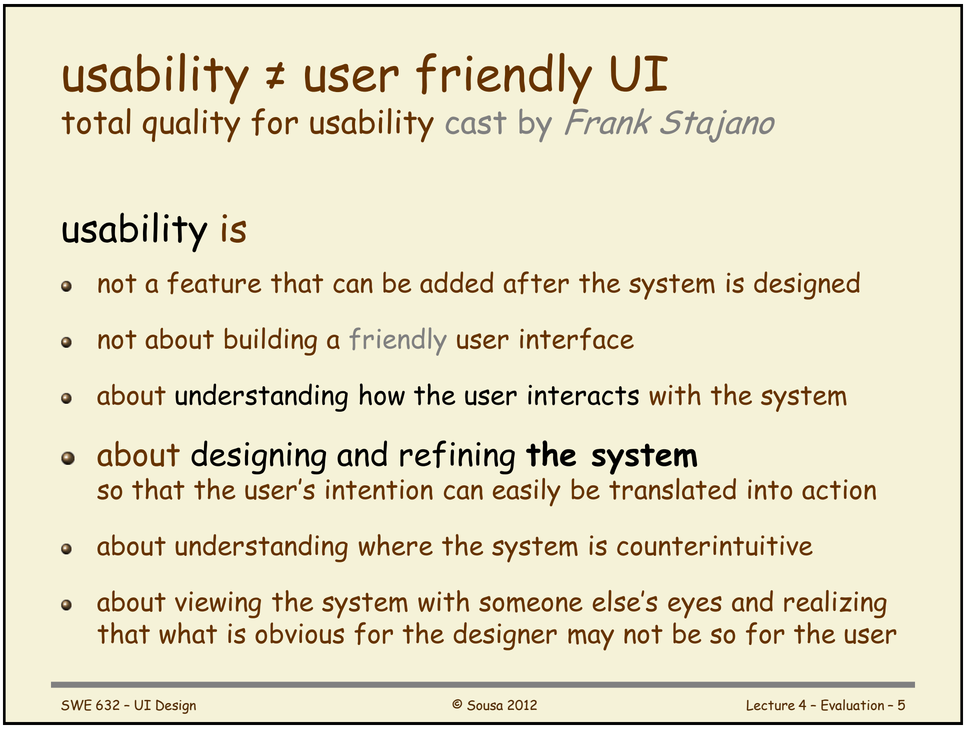
1. visit customer site
2. iterative design
3. participatory design
   mockups
4. prototyping
5. analysis of competition

found to have most impact

1. iterative design
2. user & task modeling
3. empirical studies
4. participatory design
5. visit customer site
6. post-release follow-up

practitioners survey

SWE 632 – UI Design © Sousa 2012 Lecture 4 – Evaluation – 4

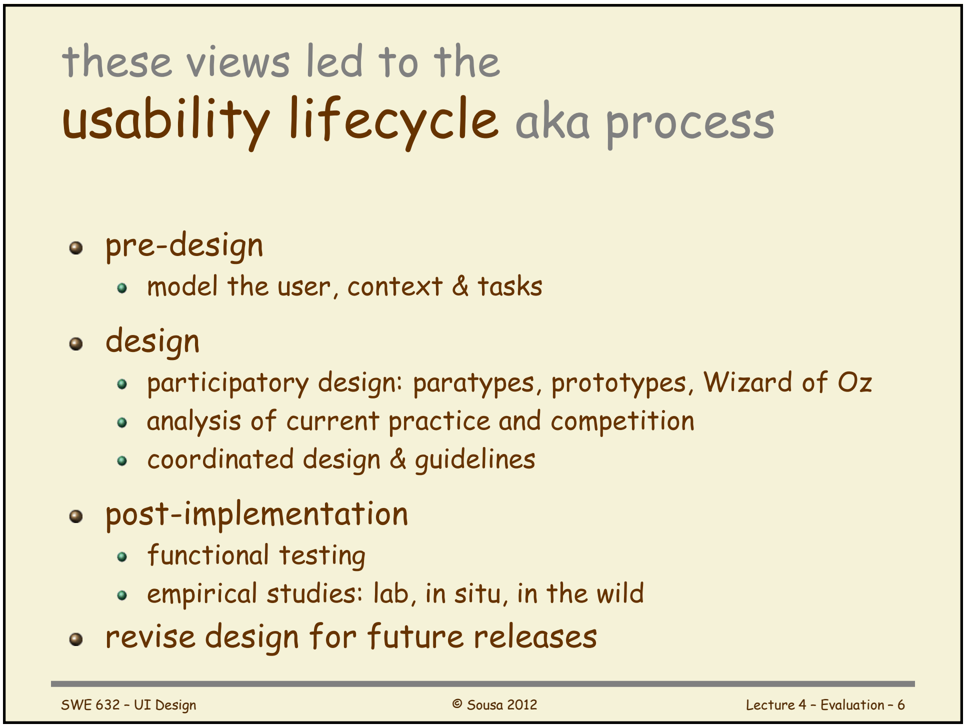


usability ≠ user friendly UI

total quality for usability cast by *Frank Stajano*

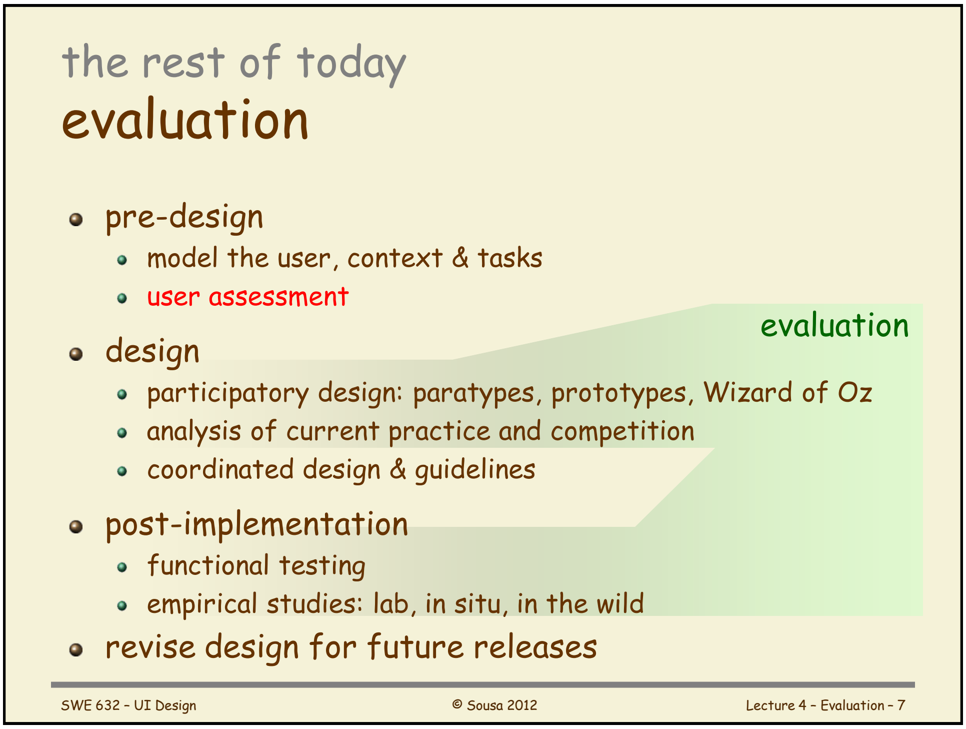
usability is

- not a feature that can be added after the system is designed
- not about building a friendly user interface
- about understanding how the user interacts with the system
- about designing and refining **the system**
  so that the user's intention can easily be translated into action
- about understanding where the system is counterintuitive
- about viewing the system with someone else's eyes and realizing that what is obvious for the designer may not be so for the user



these views led to the usability lifecycle aka process

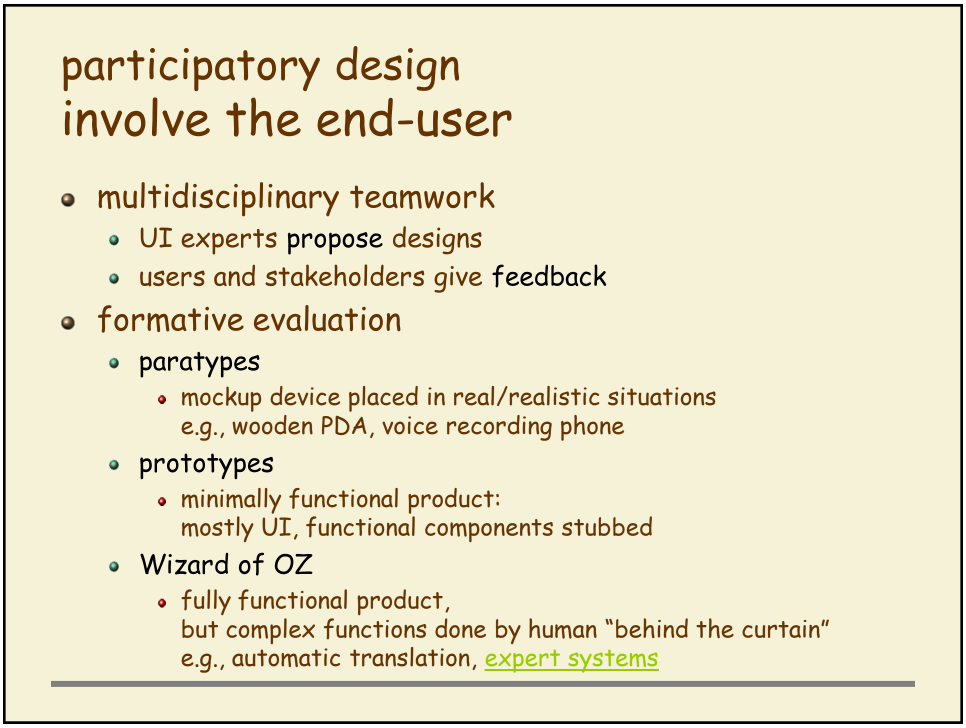
- pre-design
  - model the user, context & tasks
- design
  - participatory design: paratypes, prototypes, Wizard of Oz
  - analysis of current practice and competition
  - coordinated design & guidelines
- post-implementation
  - functional testing
  - empirical studies: lab, in situ, in the wild
- revise design for future releases



the rest of today
evaluation

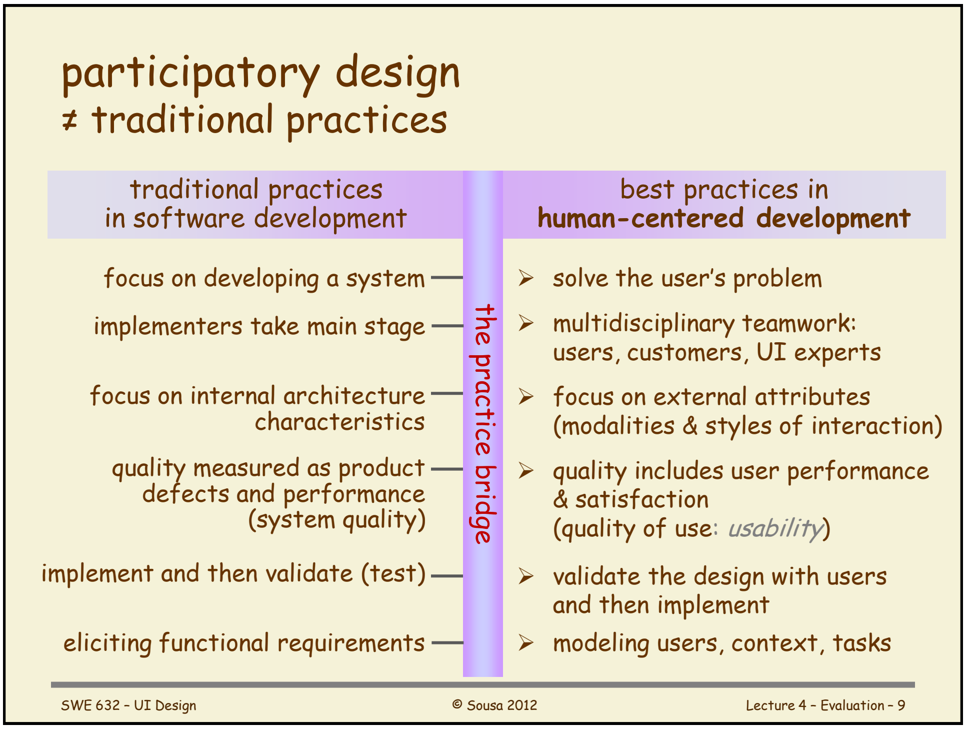
- pre-design
  - model the user, context & tasks
  - user assessment
- design
  - participatory design: paratypes, prototypes, Wizard of Oz
  - analysis of current practice and competition
  - coordinated design & guidelines
- post-implementation
  - functional testing
  - empirical studies: lab, in situ, in the wild
- revise design for future releases

evaluation



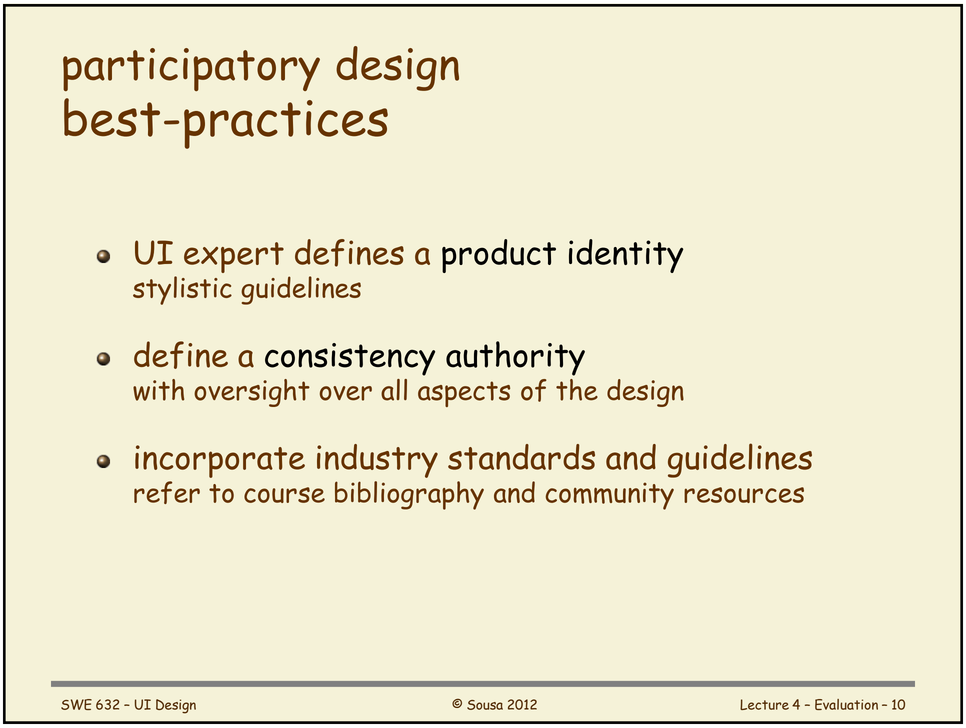
participatory design
involve the end-user

- multidisciplinary teamwork
  - UI experts propose designs
  - users and stakeholders give feedback
- formative evaluation
  - paratypes
    - mockup device placed in real/realistic situations
      e.g., wooden PDA, voice recording phone
  - prototypes
    - minimally functional product:
      mostly UI, functional components stubbed
  - Wizard of OZ
    - fully functional product,
      but complex functions done by human “behind the curtain”
      e.g., automatic translation, expert systems



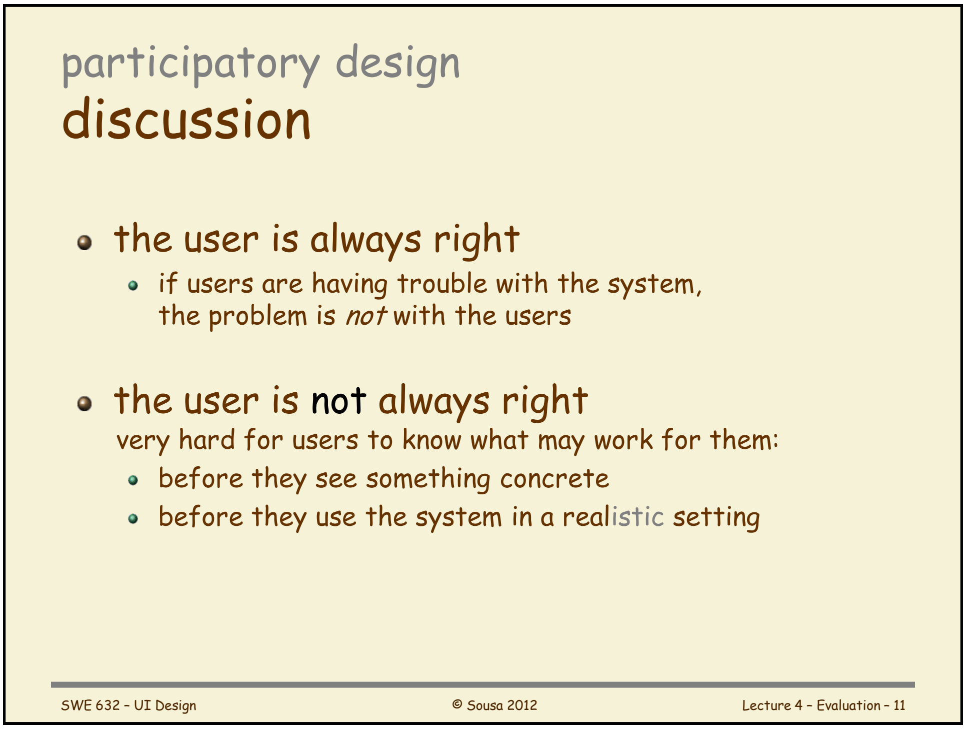
participatory design ≠ traditional practices

traditional practices in software development		best practices in **human-centered development**
focus on developing a system	the practice bridge	solve the user's problem
implementers take main stage		multidisciplinary teamwork: users, customers, UI experts
focus on internal architecture characteristics		focus on external attributes (modalities & styles of interaction)
quality measured as product defects and performance (system quality)		quality includes user performance & satisfaction (quality of use: *usability*)
implement and then validate (test)		validate the design with users and then implement
eliciting functional requirements		modeling users, context, tasks



participatory design best-practices

- UI expert defines a product identity
  stylistic guidelines
- define a consistency authority
  with oversight over all aspects of the design
- incorporate industry standards and guidelines
  refer to course bibliography and community resources



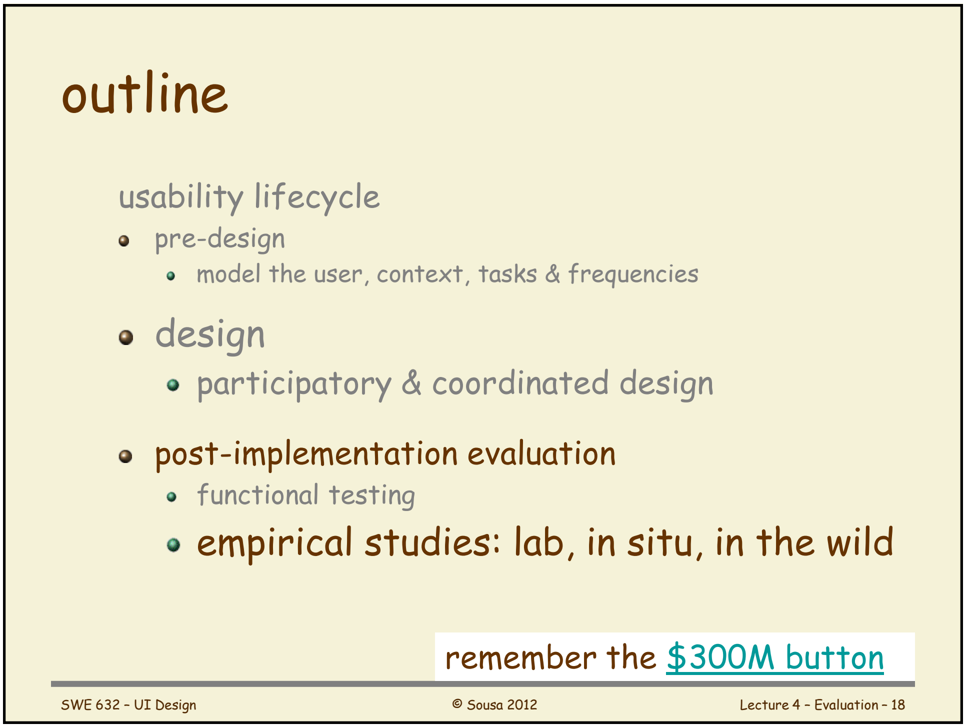
participatory design

discussion

- the user is always right
  - if users are having trouble with the system, the problem is *not* with the users
- the user is not always right

  very hard for users to know what may work for them:
  - before they see something concrete
  - before they use the system in a realistic setting

SWE 632 – UI Design © Sousa 2012 Lecture 4 – Evaluation – 11



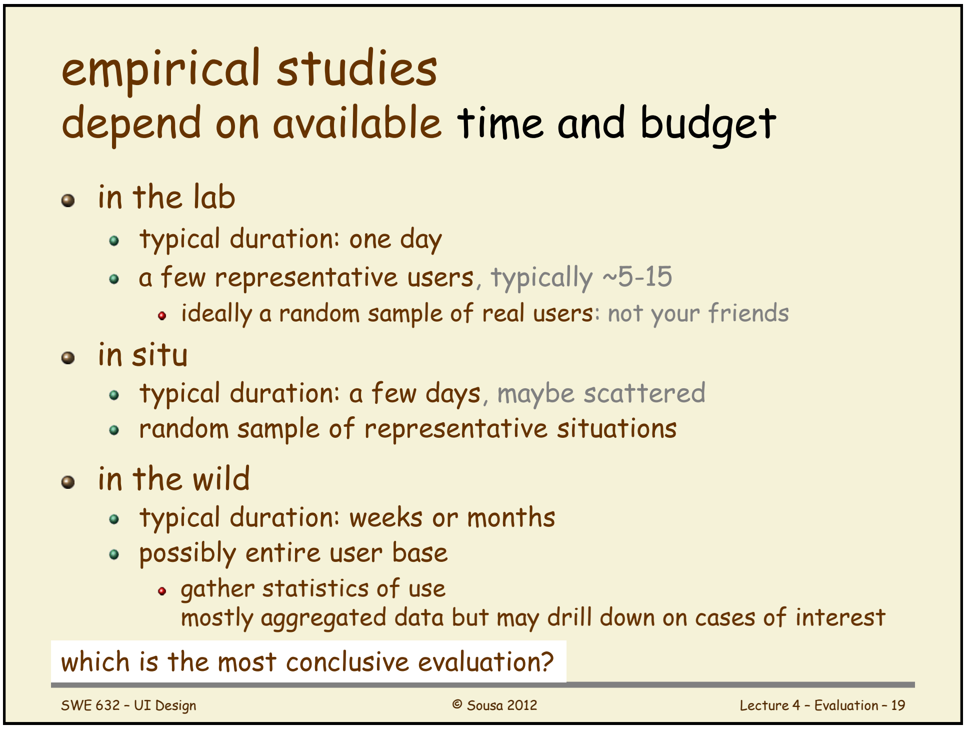
outline

usability lifecycle

- pre-design
  - model the user, context, tasks & frequencies
- design
  - participatory & coordinated design
- post-implementation evaluation
  - functional testing
  - empirical studies: lab, in situ, in the wild

remember the $300M button

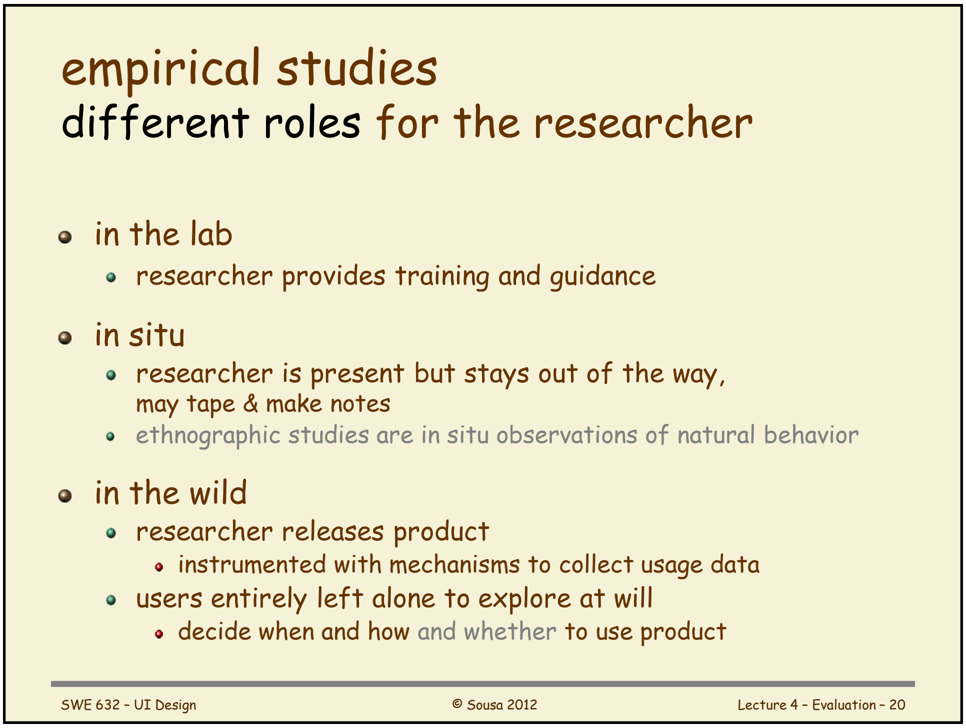
SWE 632 – UI Design © Sousa 2012 Lecture 4 – Evaluation – 18



empirical studies
depend on available time and budget

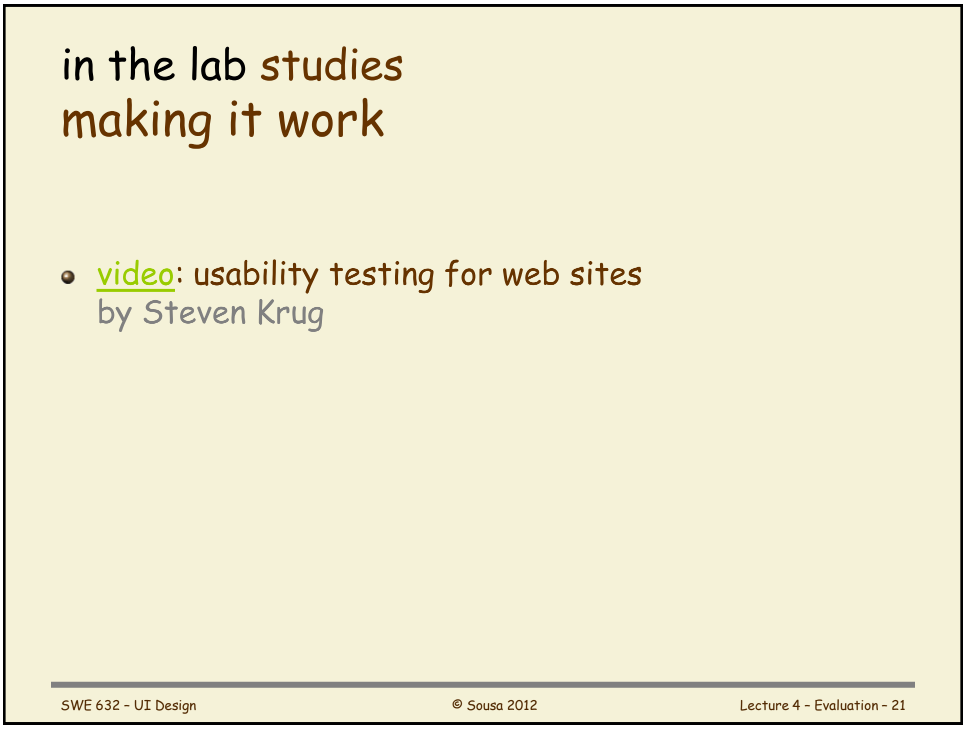
- in the lab
  - typical duration: one day
  - a few representative users, typically ~5-15
    - ideally a random sample of real users: not your friends
- in situ
  - typical duration: a few days, maybe scattered
  - random sample of representative situations
- in the wild
  - typical duration: weeks or months
  - possibly entire user base
    - gather statistics of use
      mostly aggregated data but may drill down on cases of interest

which is the most conclusive evaluation?



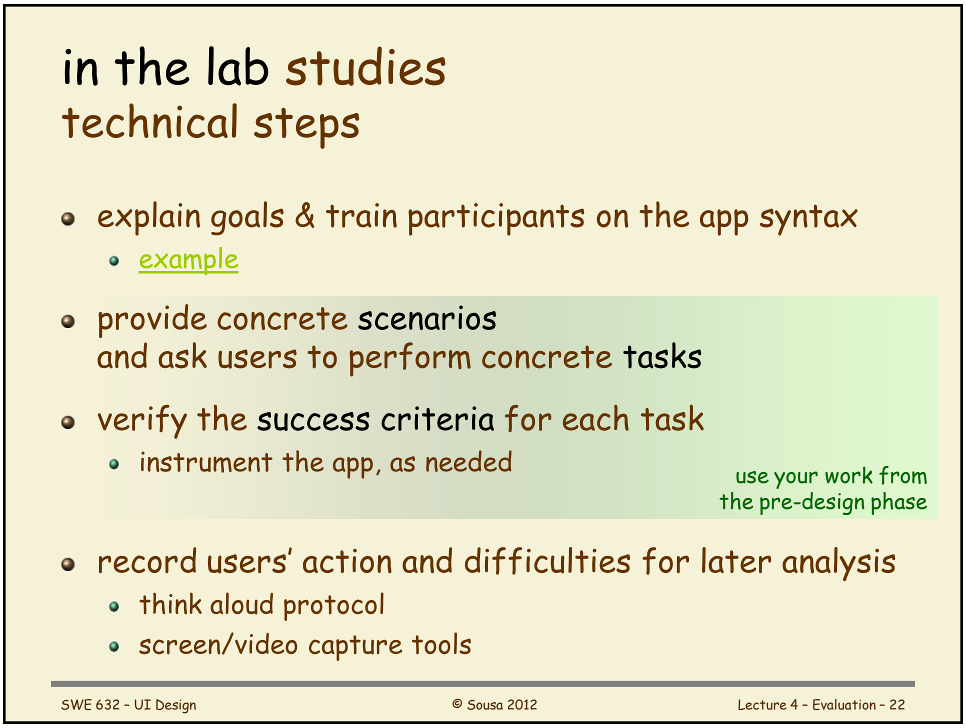
empirical studies
different roles for the researcher

- in the lab
  - researcher provides training and guidance
- in situ
  - researcher is present but stays out of the way,
    may tape & make notes
  - ethnographic studies are in situ observations of natural behavior
- in the wild
  - researcher releases product
    - instrumented with mechanisms to collect usage data
  - users entirely left alone to explore at will
    - decide when and how and whether to use product



in the lab studies
making it work

- video: usability testing for web sites
  by Steven Krug



in the lab studies
technical steps

- explain goals & train participants on the app syntax
  - example
- provide concrete scenarios
  and ask users to perform concrete tasks
- verify the success criteria for each task
  - instrument the app, as needed

use your work from
the pre-design phase

- record users' action and difficulties for later analysis
  - think aloud protocol
  - screen/video capture tools

what to measure
usability metrics

remember: you are not a typical user
measure these for real users

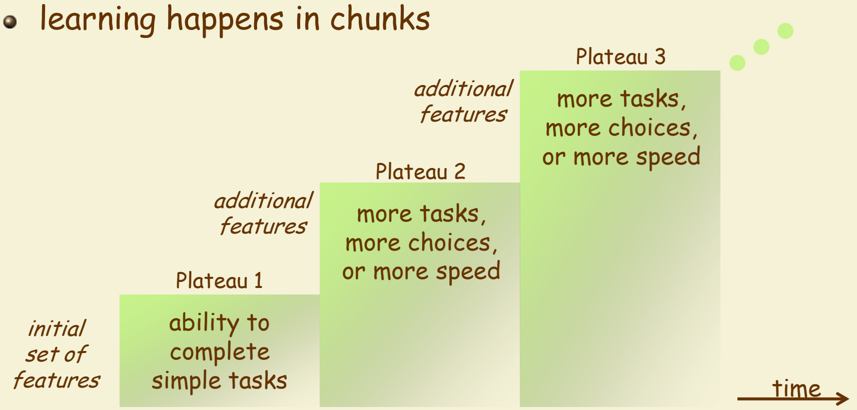
1. time to learn
2. speed of user performance
3. rate of errors by users
4. retention over time
5. subjective satisfaction

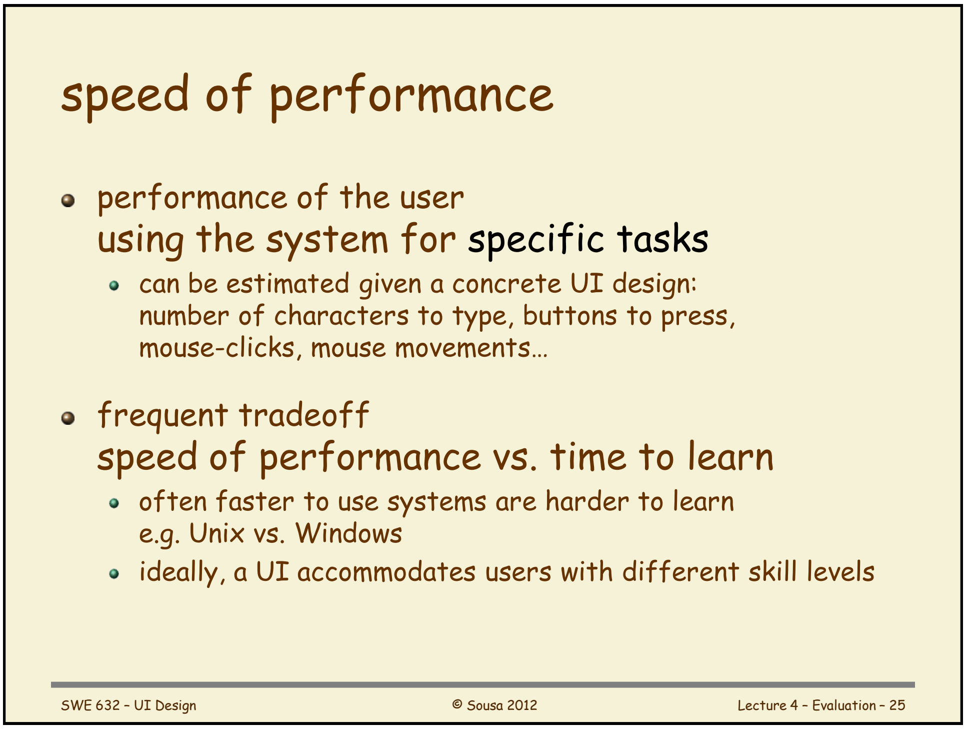
measurable quantitatively

let's look at these in turn

time to learn

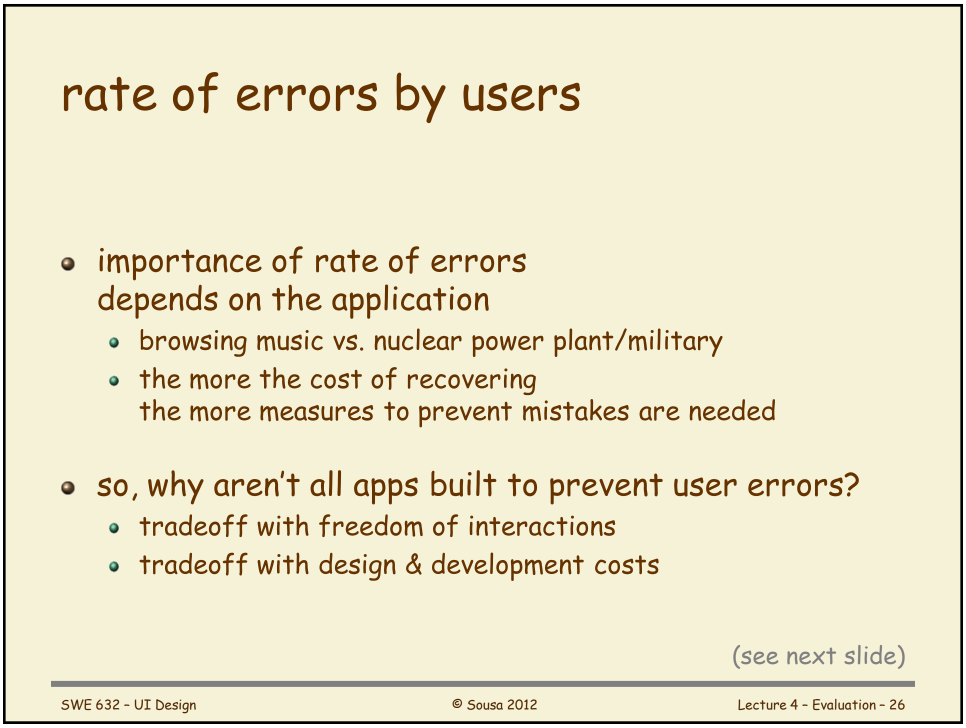
- how long does it take to be able to use an interface to carry out a (set of) task(s)
- learning happens in chunks


initial set of features
Plateau 1
ability to complete simple tasks
additional features
Plateau 2
more tasks, more choices, or more speed
additional features
Plateau 3
more tasks, more choices, or more speed
time



speed of performance

- performance of the user
  using the system for specific tasks
  - can be estimated given a concrete UI design:
    number of characters to type, buttons to press,
    mouse-clicks, mouse movements…
- frequent tradeoff
  speed of performance vs. time to learn
  - often faster to use systems are harder to learn
    e.g. Unix vs. Windows
  - ideally, a UI accommodates users with different skill levels



rate of errors by users

- importance of rate of errors
  depends on the application
  - browsing music vs. nuclear power plant/military
  - the more the cost of recovering
    the more measures to prevent mistakes are needed
- so, why aren't all apps built to prevent user errors?
  - tradeoff with freedom of interactions
  - tradeoff with design & development costs

(see next slide)

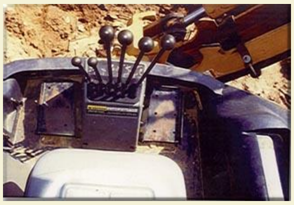
rate of errors by users

- tradeoff freedom/errors
  - the more freedom a UI provides
    the more likely are users to make mistakes
  - the more guidance, the more constraints, the less mistakes
  - different styles offer different tradeoff
    - e.g., command line versus GUI
- tradeoff D&D cost/errors
  rate of errors also affected by factors such as:
  - adequacy of design & instructions to user tasks & profile
  - consistency of interactions
  - organization of interactions
    e.g. how much a user has to remember/transfer
    from one interaction to another

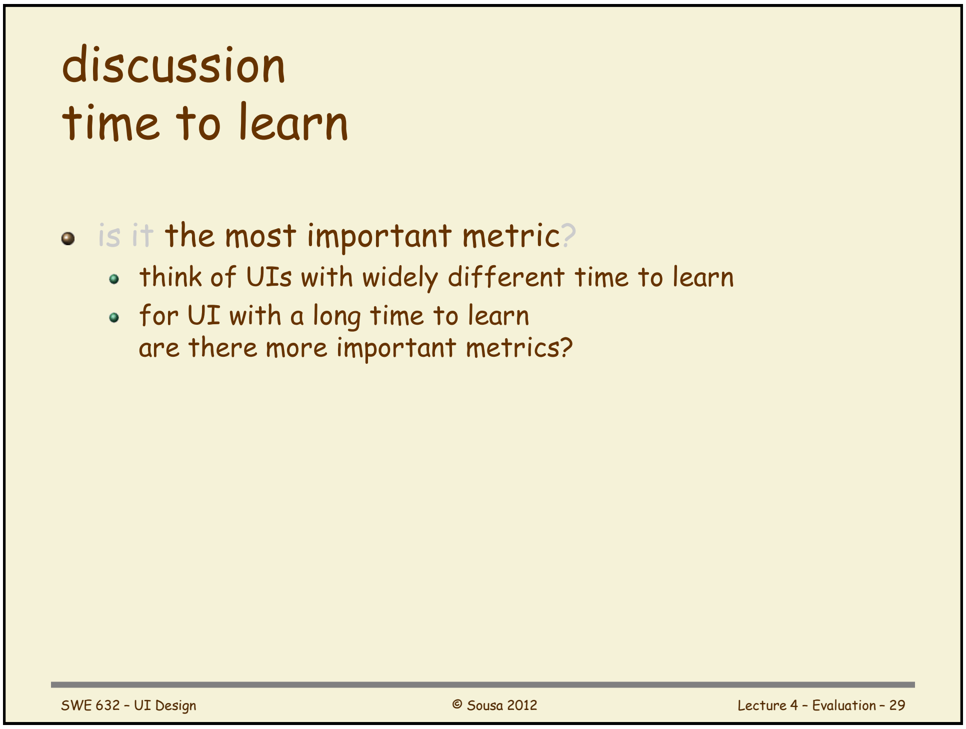
making a good fit, high-quality UI is hard work

retention over time

- related with time to learn
  - retention is more important if learning is costly
- UIs are easier to learn & remember if operations match user intuitions
  - e.g., using a cooking stove vs. controlling a backhoe

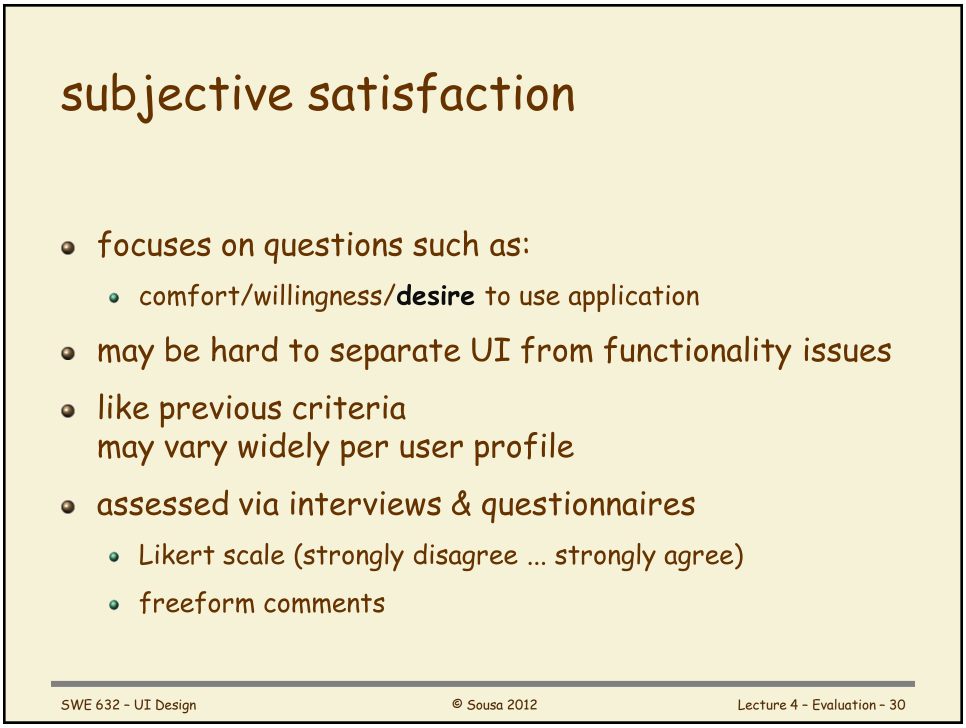


challenge:
what would be
an easy-to-learn
UI for the hoe?



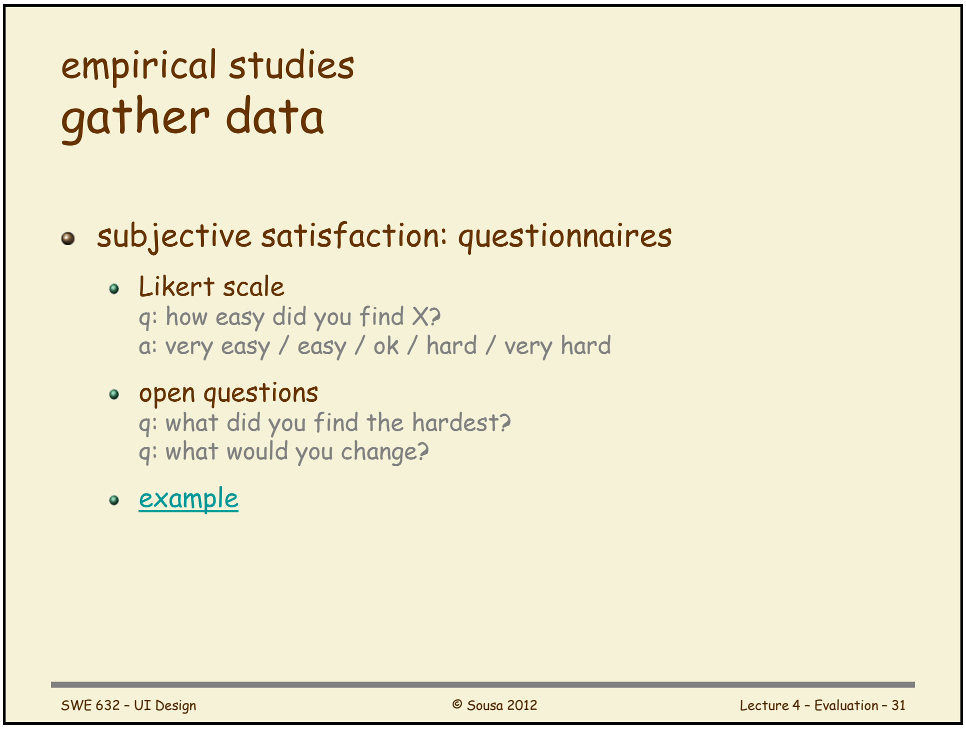
discussion
time to learn

- is it the most important metric?
  - think of UIs with widely different time to learn
  - for UI with a long time to learn
    are there more important metrics?



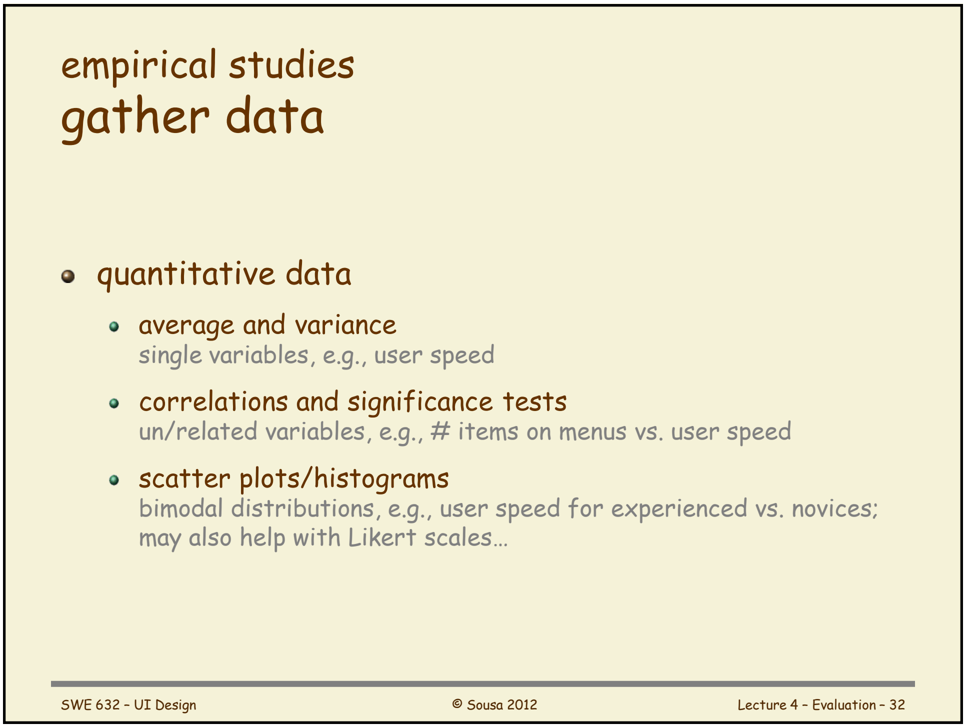
subjective satisfaction

- focuses on questions such as:
  - comfort/willingness/**desire** to use application
- may be hard to separate UI from functionality issues
- like previous criteria
  may vary widely per user profile
- assessed via interviews & questionnaires
  - Likert scale (strongly disagree … strongly agree)
  - freeform comments



empirical studies
gather data

- subjective satisfaction: questionnaires
  - Likert scale
    q: how easy did you find X?
    a: very easy / easy / ok / hard / very hard
  - open questions
    q: what did you find the hardest?
    q: what would you change?
  - example



empirical studies
gather data

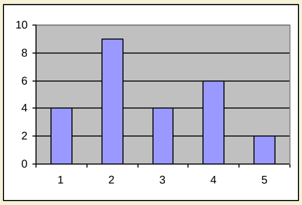
- quantitative data
  - average and variance
    single variables, e.g., user speed
  - correlations and significance tests
    un/related variables, e.g., # items on menus vs. user speed
  - scatter plots/histograms
    bimodal distributions, e.g., user speed for experienced vs. novices; may also help with Likert scales…

discussion
gathering data

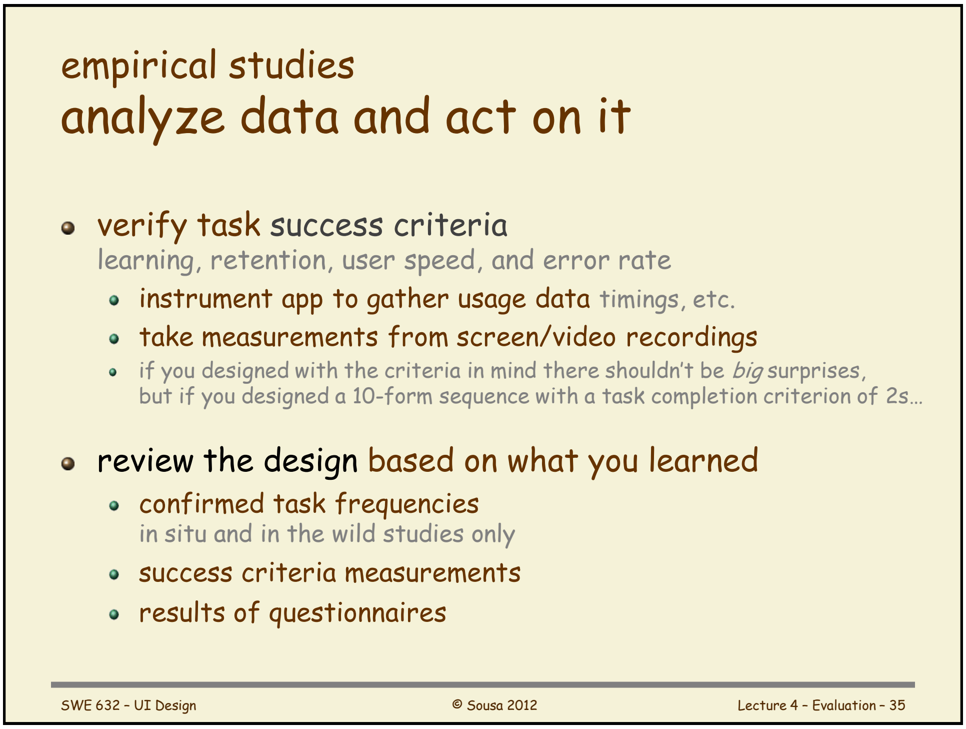
- suppose your team is debating two design alternatives
  - you evaluate one with user A and the other with user B
  - A performed much better than B, what do you conclude?
    - difference may be due to user variability as much as 10x
    - have users (prefb. more users) test both designs and compare performance *diff for each* user
- suppose you evaluate some x of interest and the average x for a group of users is much worse than you expected, what do you conclude?

example
survey on context-aware reminders

- question: would you like to have the app remind you to take your laptop if you'll need it during the day, before leaving home?
  - answers: 3 4 2 4 2 5 1 2 2 4 4 3 2 3 2 3 1 1 1 4 2 4 5 2 2 (1 - no, 2 - not really, 3 - maybe, 4 - yes, 5 - absolutely)
  - 25 respondents, average 2.72, mode 2
  - how do you interpret the results?
  - do an histogram:
    - subgroups of users with diff reactions personae

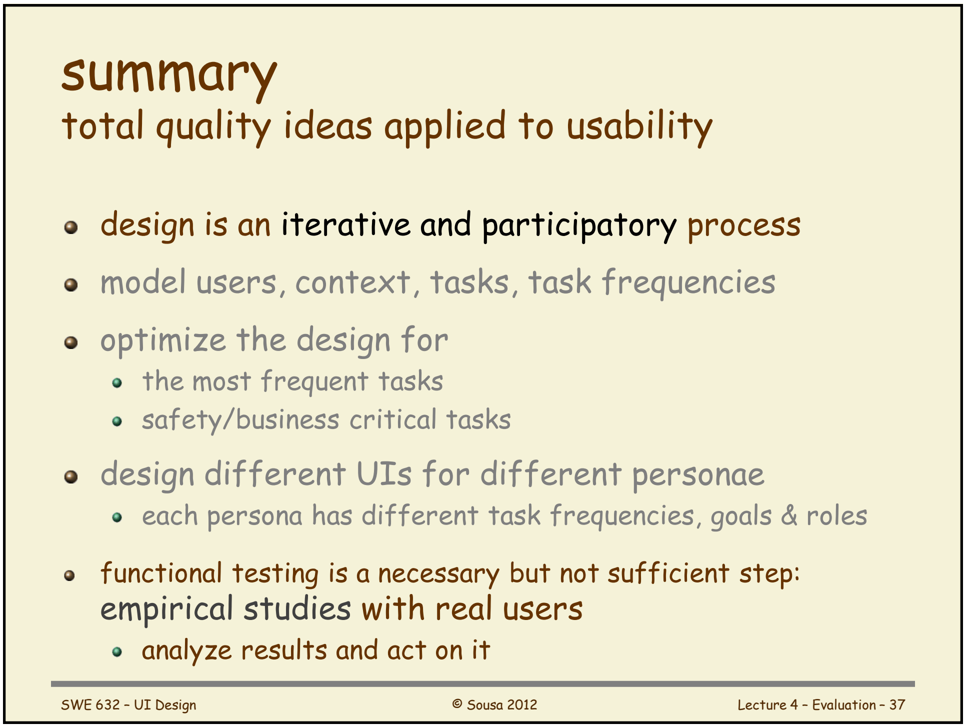


- also: why did you get those reactions? use disambiguation questions
  - do you normally take your laptop to work/school?
  - are you ok with always taking the laptop, even if you don't need it?
  - would you like to get a reminder…?



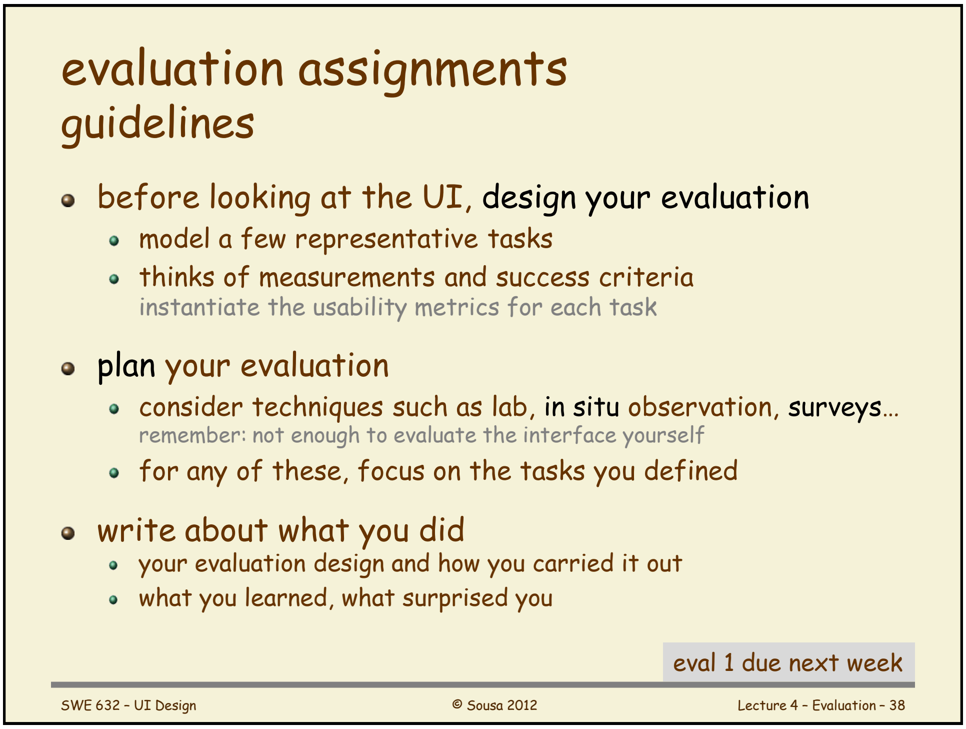
empirical studies
analyze data and act on it

- verify task success criteria
  learning, retention, user speed, and error rate
  - instrument app to gather usage data timings, etc.
  - take measurements from screen/video recordings
  - if you designed with the criteria in mind there shouldn't be *big* surprises, but if you designed a 10-form sequence with a task completion criterion of 2s…
- review the design based on what you learned
  - confirmed task frequencies
    in situ and in the wild studies only
  - success criteria measurements
  - results of questionnaires



summary
total quality ideas applied to usability

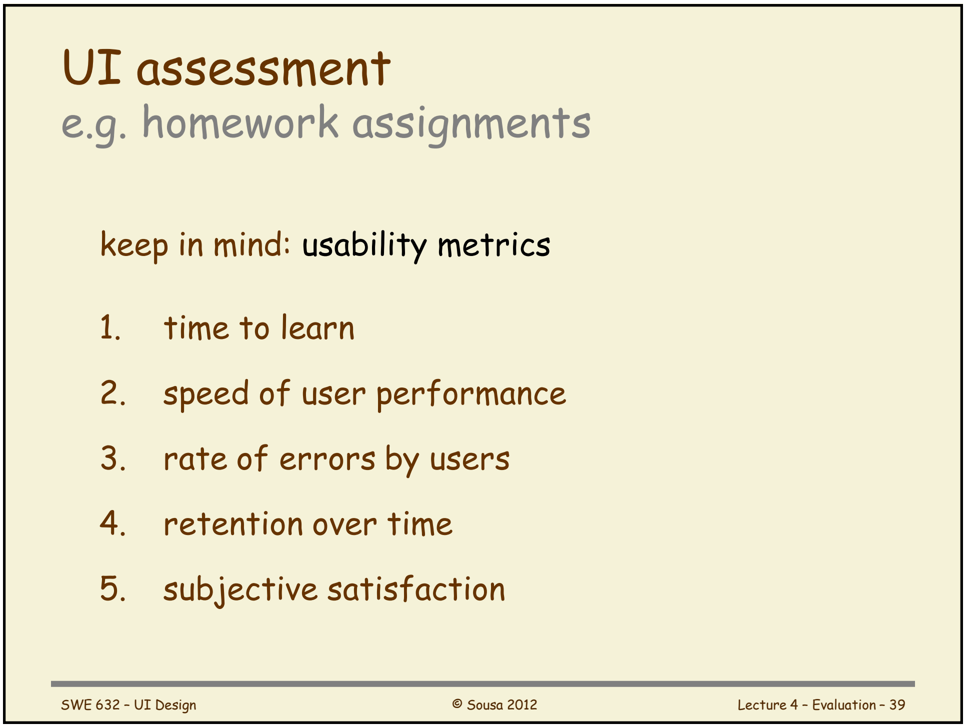
- design is an iterative and participatory process
- model users, context, tasks, task frequencies
- optimize the design for
  - the most frequent tasks
  - safety/business critical tasks
- design different UIs for different personae
  - each persona has different task frequencies, goals & roles
- functional testing is a necessary but not sufficient step:
  empirical studies with real users
  - analyze results and act on it



evaluation assignments
guidelines

- before looking at the UI, design your evaluation
  - model a few representative tasks
  - thinks of measurements and success criteria
    instantiate the usability metrics for each task
- plan your evaluation
  - consider techniques such as lab, in situ observation, surveys…
    remember: not enough to evaluate the interface yourself
  - for any of these, focus on the tasks you defined
- write about what you did
  - your evaluation design and how you carried it out
  - what you learned, what surprised you

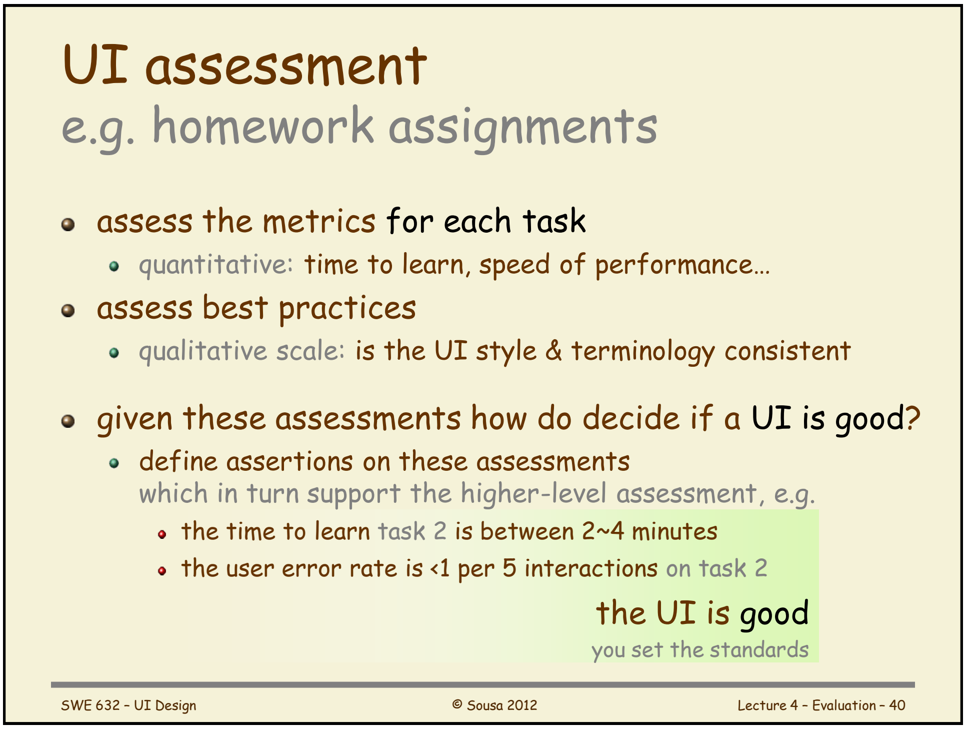
eval 1 due next week



UI assessment
e.g. homework assignments

keep in mind: usability metrics

1. time to learn
2. speed of user performance
3. rate of errors by users
4. retention over time
5. subjective satisfaction



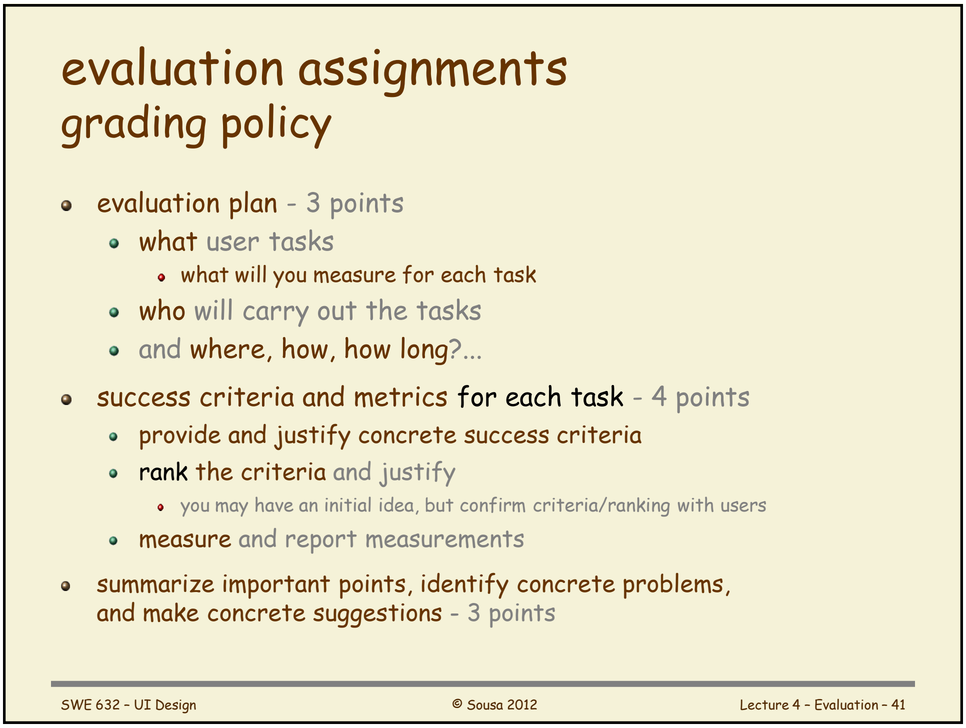
UI assessment

e.g. homework assignments

- assess the metrics for each task
  - quantitative: time to learn, speed of performance…
- assess best practices
  - qualitative scale: is the UI style & terminology consistent
- given these assessments how do decide if a UI is good?
  - define assertions on these assessments
    which in turn support the higher-level assessment, e.g.
    - the time to learn task 2 is between 2~4 minutes
    - the user error rate is <1 per 5 interactions on task 2

the UI is good

you set the standards



evaluation assignments

grading policy

- evaluation plan - 3 points
  - what user tasks
    - what will you measure for each task
  - who will carry out the tasks
  - and where, how, how long?…
- success criteria and metrics for each task - 4 points
  - provide and justify concrete success criteria
  - rank the criteria and justify
    - you may have an initial idea, but confirm criteria/ranking with users
  - measure and report measurements
- summarize important points, identify concrete problems, and make concrete suggestions - 3 points