context
what is interaction design?

what the term means
› design of digital products and services
› focus on behavior, and interaction with user

in this nugget, used more narrowly
› navigational structure of a web app
› essential actions and states
the essence of design

The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface.

—Bruce Tonazzini (http://www.asktog.com/basics/firstPrinciples.html)

To design something really well, you have to get it. You have to really grok what it's all about. It takes a passionate commitment to really thoroughly understand something, chew it up, not just quickly swallow it. Most people don't take the time to do that.

—Steve Jobs, interview with Wired, Issue 4.02, Feb 1996
I will contend that conceptual integrity is the most important consideration in system design. It is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas.

—Fred Brooks, Mythical Man Month, 1975

I am more convinced than ever. Conceptual integrity is central to product quality.

tools for designing navigation

object model
› conceptual structure that lies behind the app
› relations give hints to navigations

interaction state machine
› show key pages and paths between them
review: statecharts
examples

- switch
- stop watch
- downloader
- MP3 player
key features

initial

state

event

select

end

select

pause

play

superstate
example: tipster
example: tipster OM

suggests thinking about
› navigating: subject-review, entry-user, subject-category?
› searches: by name, content, rating?
› how/if to support generalization?
a simple first design

which entities?
› focus on subjects and reviews
› no user profiles or category pages
› categories defined in background

subject/review generalization?
› no, but consider later for versioning (tricky)

search
› just for subject, by name and category

operations
› add/edit; for now, no delete (tricky)
resource structure

Each box
- represents one web page
- a ReSTful URI
- state before action occurs

/users/new (result of GET)
Action is /users [POST]
universally accessible resources

defines navigation bar
defining responses
defining errors

history state:
back to last substate
designing internal navigations

unifying edit/new

review edit/new to ...? initially /rid, but prototyping refuted
access control

when to show edit?
when to allow edit?

decisions:
show if user matches by
allow if logged in
digression: remembering return page
strategy

on access violation
› show login page with flash message
› on success, return to page

a common pattern
› redirect to login adds return page to URL
   /login?next=/subjects/1/edit
› login GET preserves return page
   action_url = url_for('login', next=next_url)
   <form class=form action="{{ action_url }}" method=POST>
› login POST redirects to return page
   return redirect(next_url)
def access_denied():
    flash('To access that page, please log in first')
    return redirect(url_for('login', next = request.path))

def requires_login(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        if not 'logged_in' in session:
            return access_denied()
        return f(*args, **kwargs)
    return decorated

@app.route('/subjects/<id>/edit', methods=['GET'])
@requires_login
def edit_subject (id): ...
@app.route('/login', methods=['GET','POST'])
def login():
    error = None
    next_url = request.args['next'] if 'next' in request.args else None
    if request.method == 'POST':
        u = User.get_by_email(request.form['email'])
        if u is not None and u.password == request.form['password']:
            session['logged_in'] = True
            session['user_id'] = u.id
            flash('Welcome back, ' + u.first)
            if next_url is not None:
                return redirect(next_url)
            else:
                return redirect(url_for('index'))
    else:
        error = 'Invalid email or password'
        action_url = url_for('login', next=next_url)
        return render_template('login.html', action_url=action_url, error=error)

<form class=form action="{{ action_url }}" method=POST>
design issues to ponder

uniformity & consistency
› do navigation & resource structures match?
› are similar features represented in similar ways?

access
› are paths to resources short enough?
› are appropriate controls in place?

undo
› can user undo mistakes easily?
challenge

my login access control scheme
  › has a security vulnerability

can you find it?
  › code will be in repo soon