concept design
part 1: concepts

Daniel Jackson · Autodesk Online Workshop · June 2025

introducing myself

my career in buildings

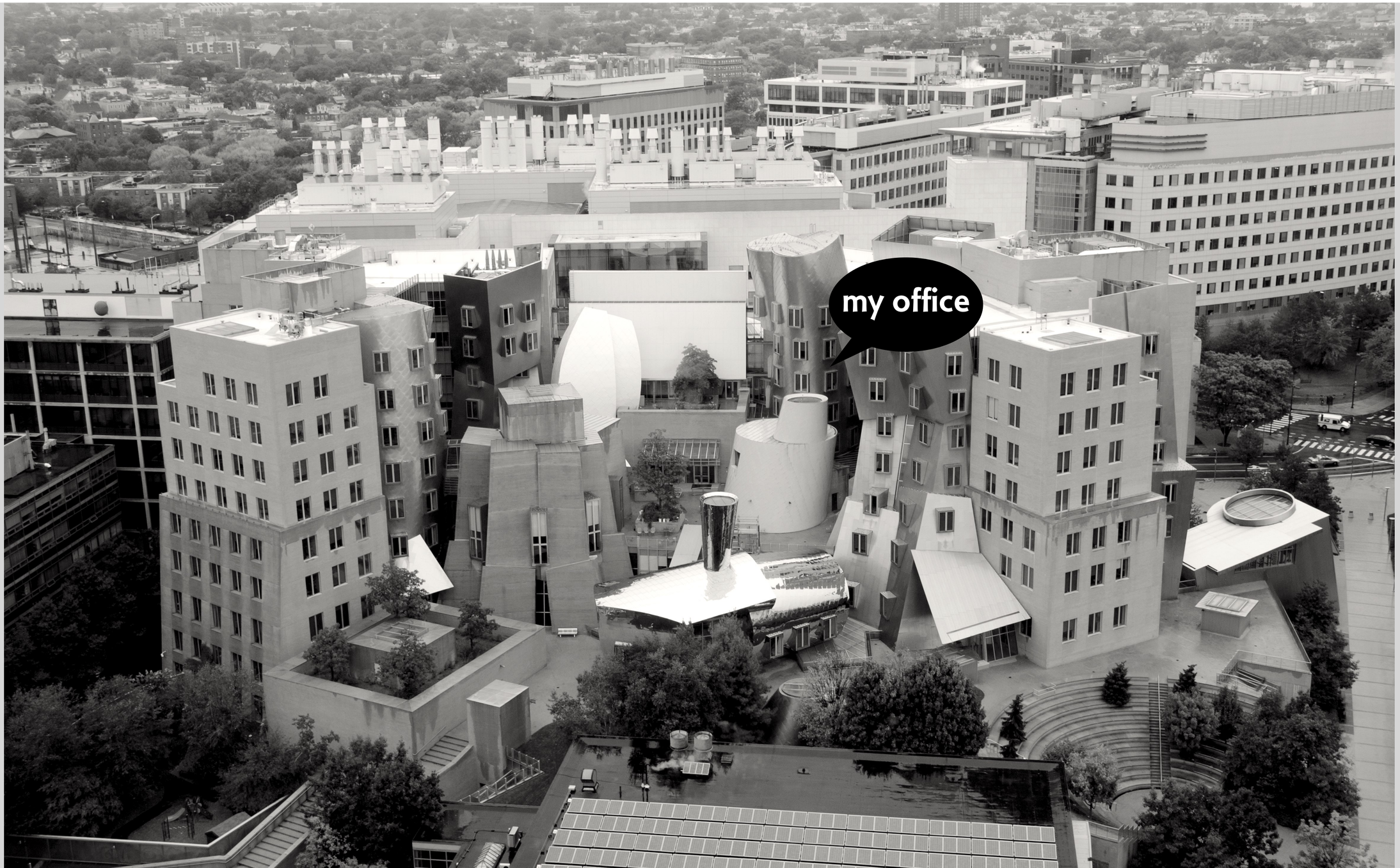
Physics at Oxford

Programmer for Logica UK

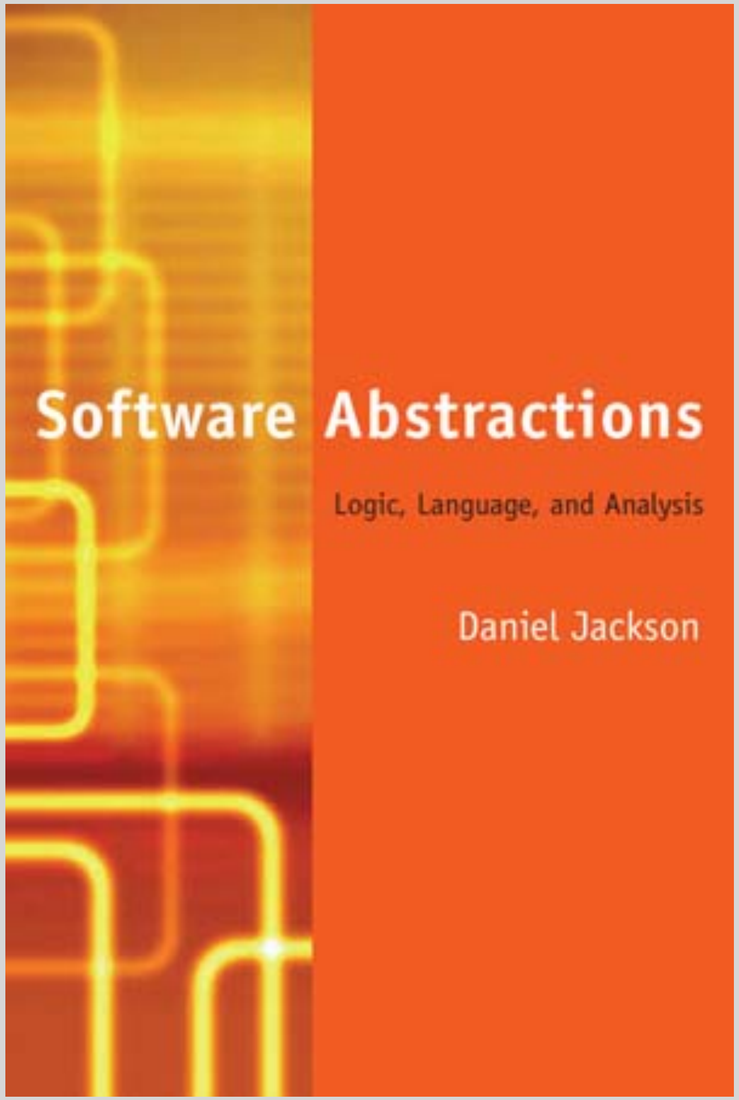
Computer science PhD at MIT

Assistant prof at CMU

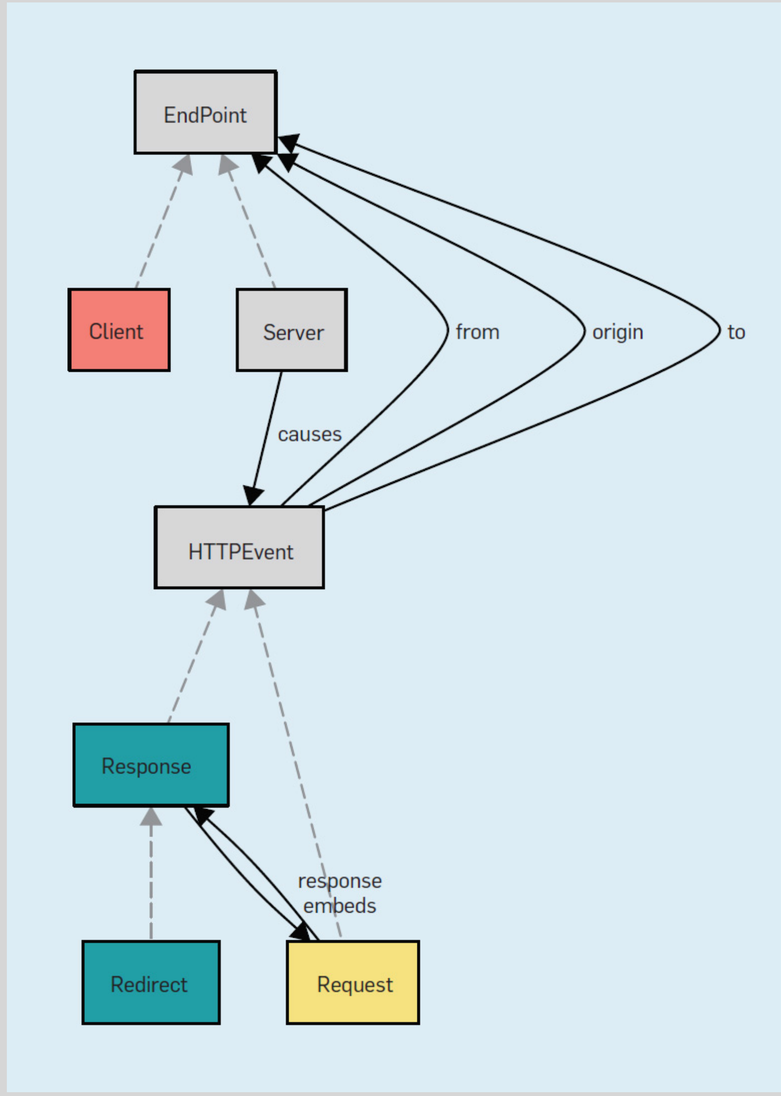
where I work now



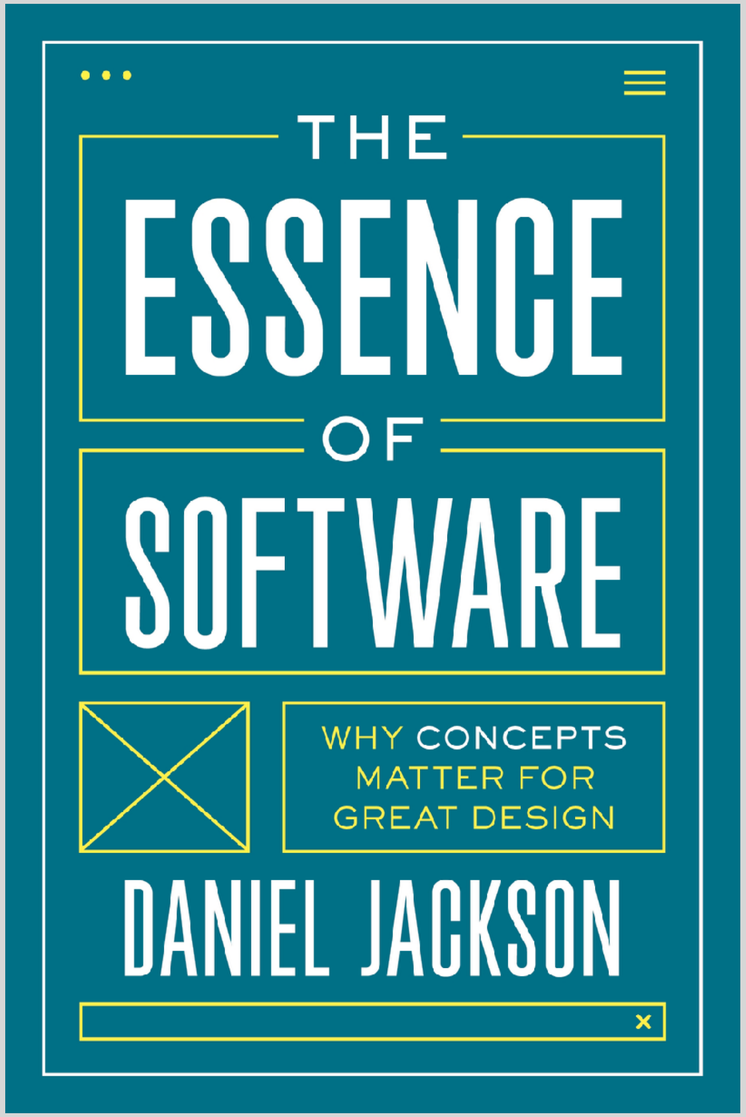
alloy: a lightweight, analyzable modeling language

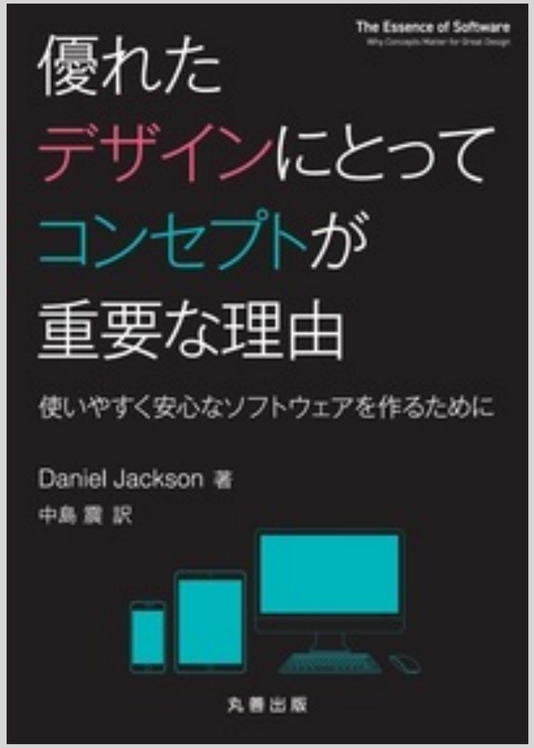



```
1   abstract sig EndPoint { }

2   sig Server extends EndPoint {
3     causes: set HTTPEvent
4     }

5   sig Client extends EndPoint { }

6   abstract sig HTTPEvent {
7     from, to, origin: EndPoint
8     }

9   sig Request extends HTTPEvent {
10    response: lone Response
11    }

12  sig Response extends HTTPEvent {
13    embeds: set Request
14    }

15  sig Redirect extends Response {
16    }
```



concepts: a new approach to software design





when I'm not working

how this project began
(almost 20 years ago)

a simple task: sign and return

Field Trip Permission Form

Dear Parents:

Ms. Frizzle will again be taking her second grade class on an exciting field trip. Please sign and return the permission slip below.

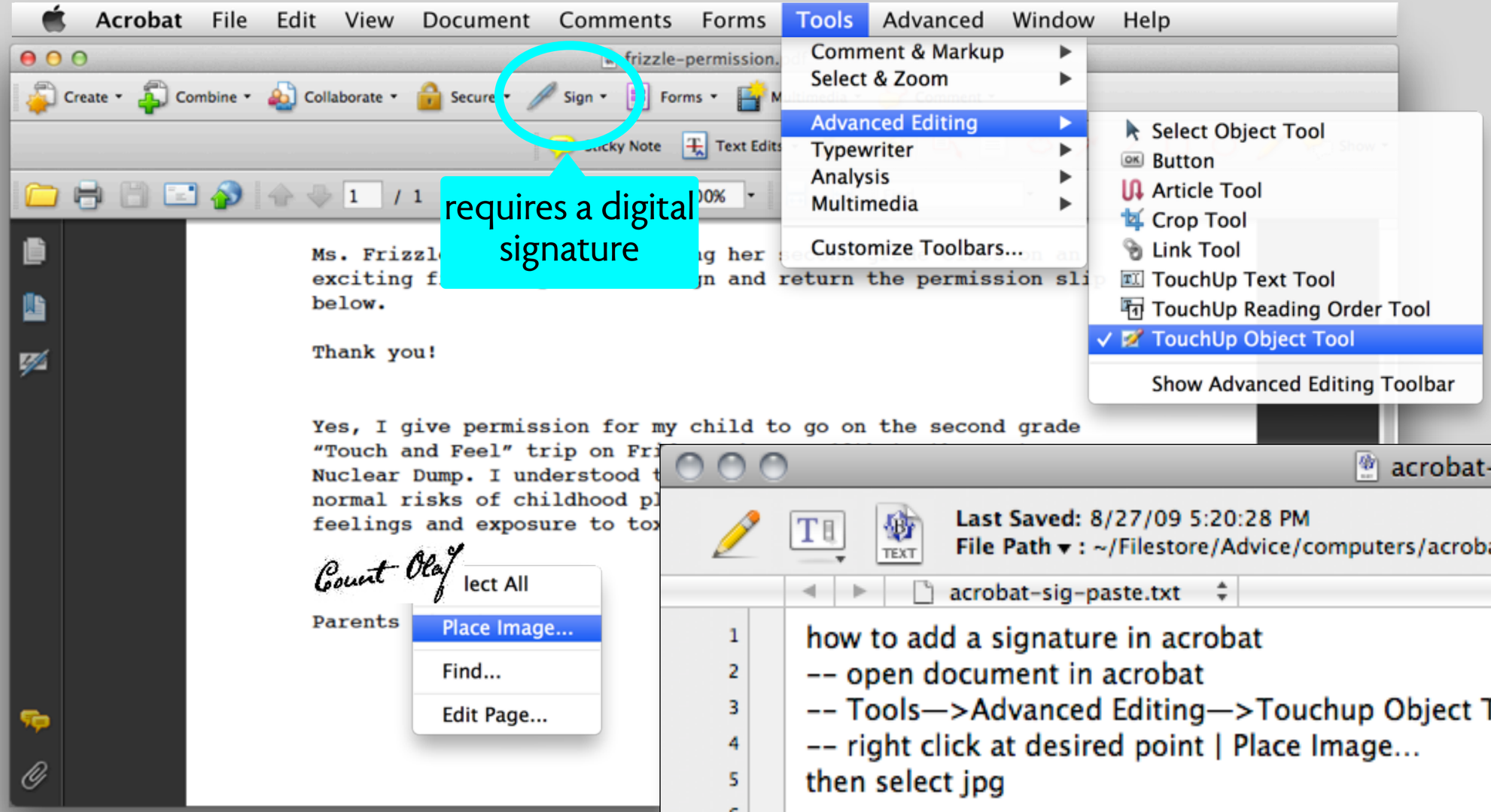
Thank you!

Yes, I give permission for my child to go on the second grade "Touch and Feel" trip on Friday February 13th to the NastyCo Nuclear Dump. I understood that my child may encounter the normal risks of childhood play, including grazed knees, hurt feelings and exposure to toxic waste.

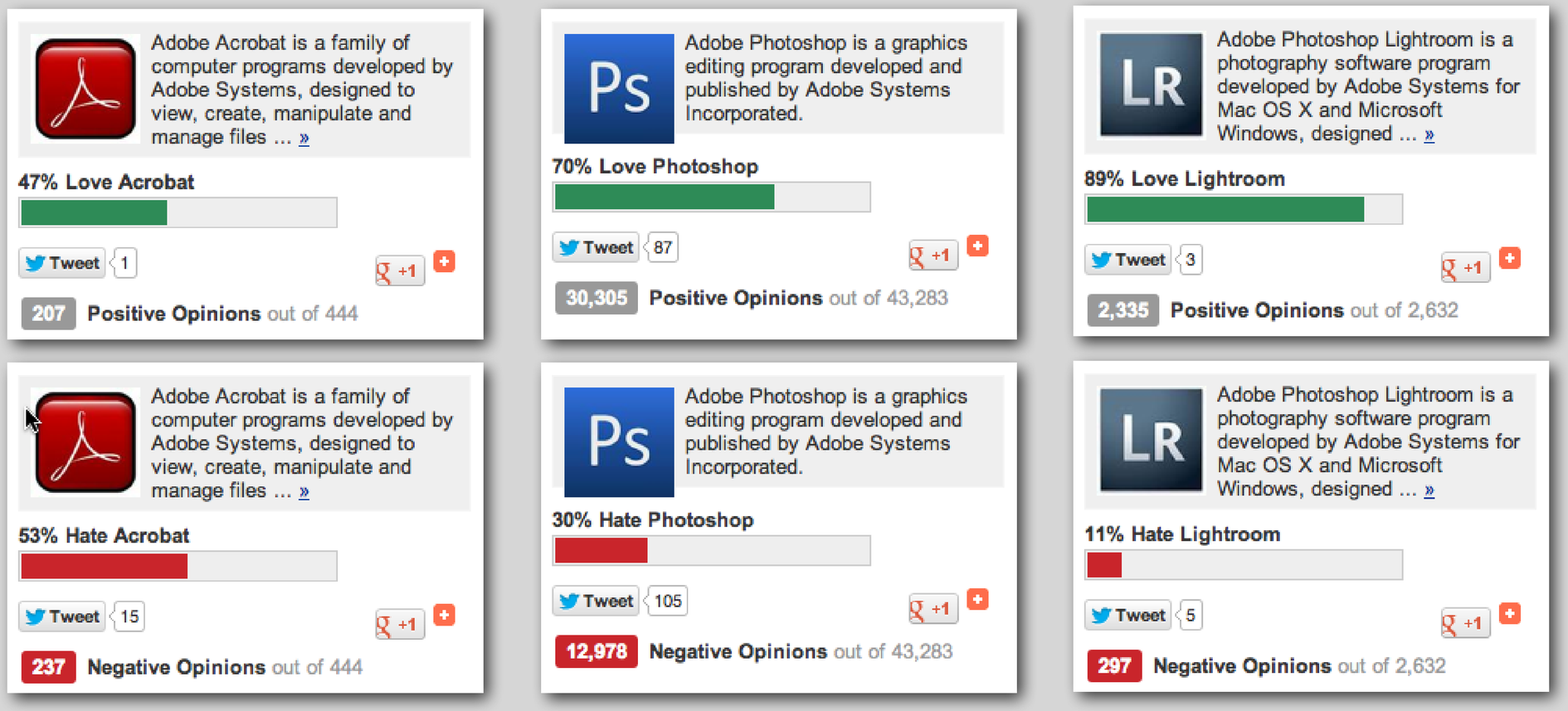
Count Olaf Dec 12, 2009

Parents signature Date

acrobat to the rescue?



not just me ...



from http://amplicate.com

what kinds of problems are these?

not human errors

not bugs in the code

not UI design flaws

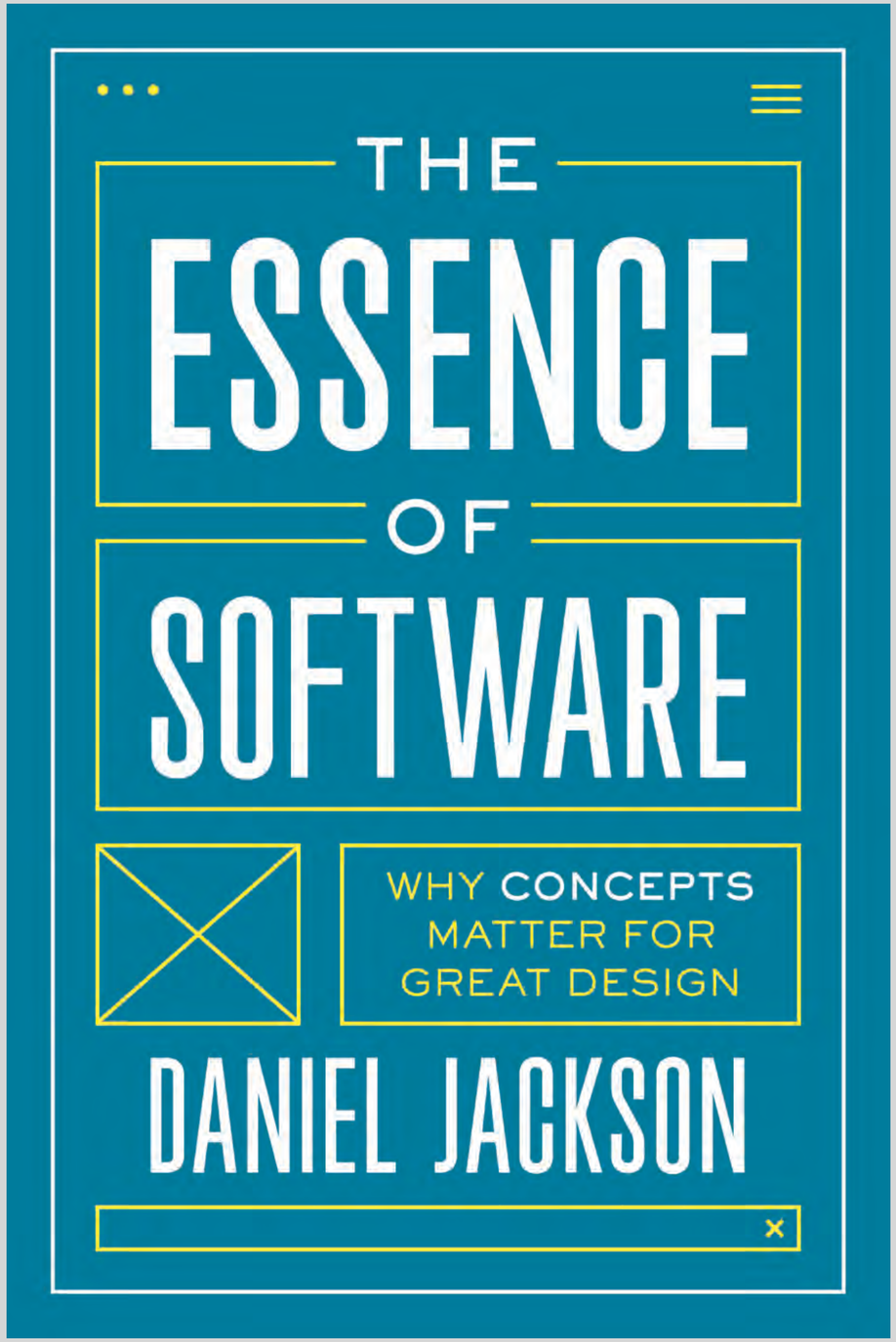
not lack of technology

if only...

we could figure out...
what makes some apps slick and easy-to-use and some clunky?
why some products take off and others gather dust?
how to design apps to make them flexible, powerful and simple?

then we might...
improve the quality of software & people's lives
know how to design successful products
reduce complexity for users & developers alike

another way to put it
what do the best designers know?
can we formulate it systematically?



analyzed about 100 software apps
what makes them good? what causes problems?
best not worst: Adobe, Apple, Google, Microsoft, etc

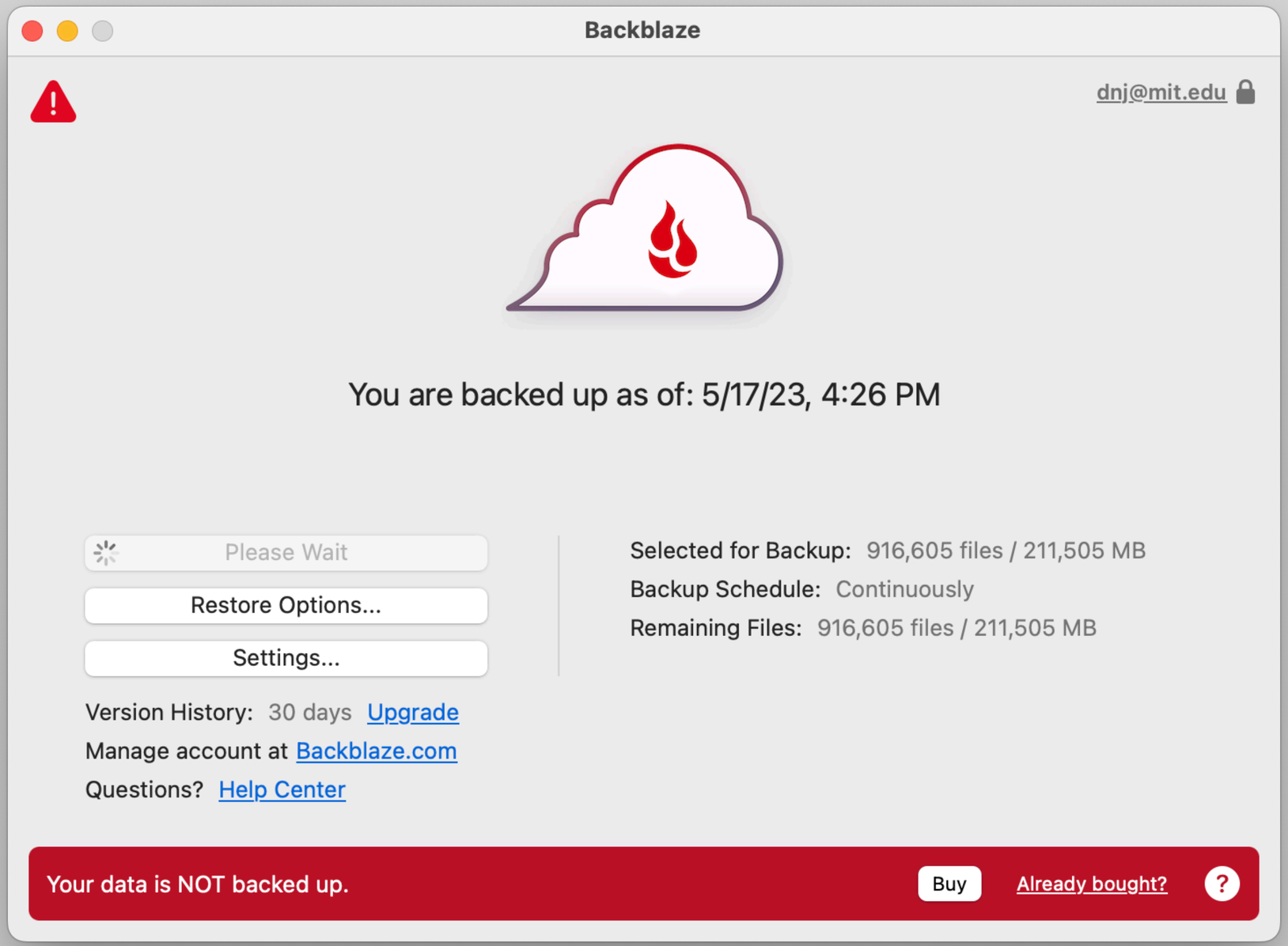
evolved approach to concept design
a way to describe & structure functionality
simple & applicable design principles
showed how violations lead to bad experiences

half the book is end notes
organized as standalone mini-essays
where the substance is (including my best pasta recipe)

a UX puzzle

Backblaze

backing up on Backblaze



was modification at 10pm saved?

Backblaze Backup

Search

dnj@mit.edu

You are backed up as of: 6/6/22, 10:10 PM

Currently backing up newer files

is backup running or not?

Pause Backup

Restore Options...

Selected for Backup: 509,021 files / 2,379,995 MB

Backup Schedule: Continuously

Remaining Files: 0 files / 0 KB

huh?

Transferring: photo.0259-22.RAF

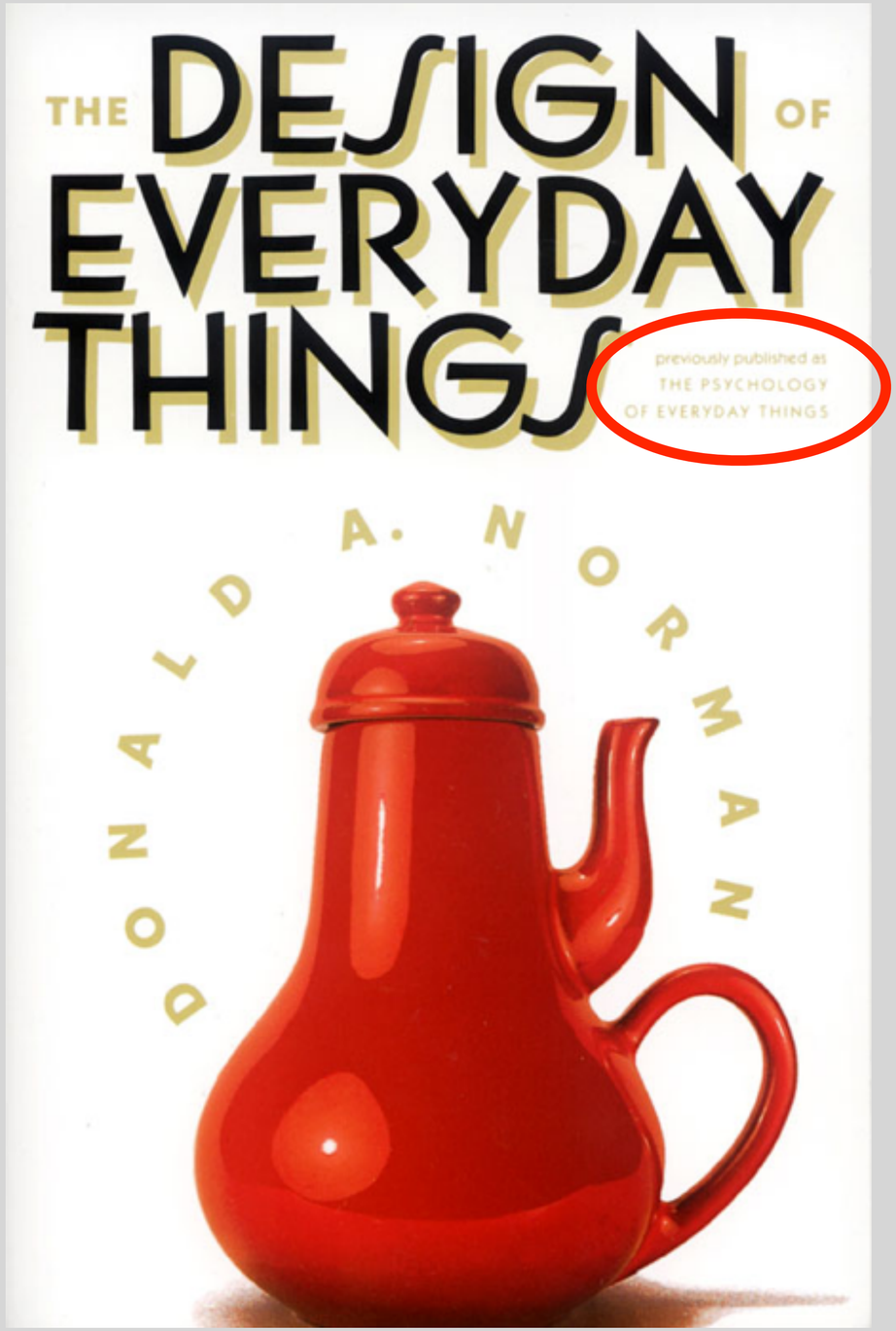
Settings...

What is being backed up?

How long will my first backup take?

View files and manage account at: Backblaze.com

conceptual models
solving Backblaze

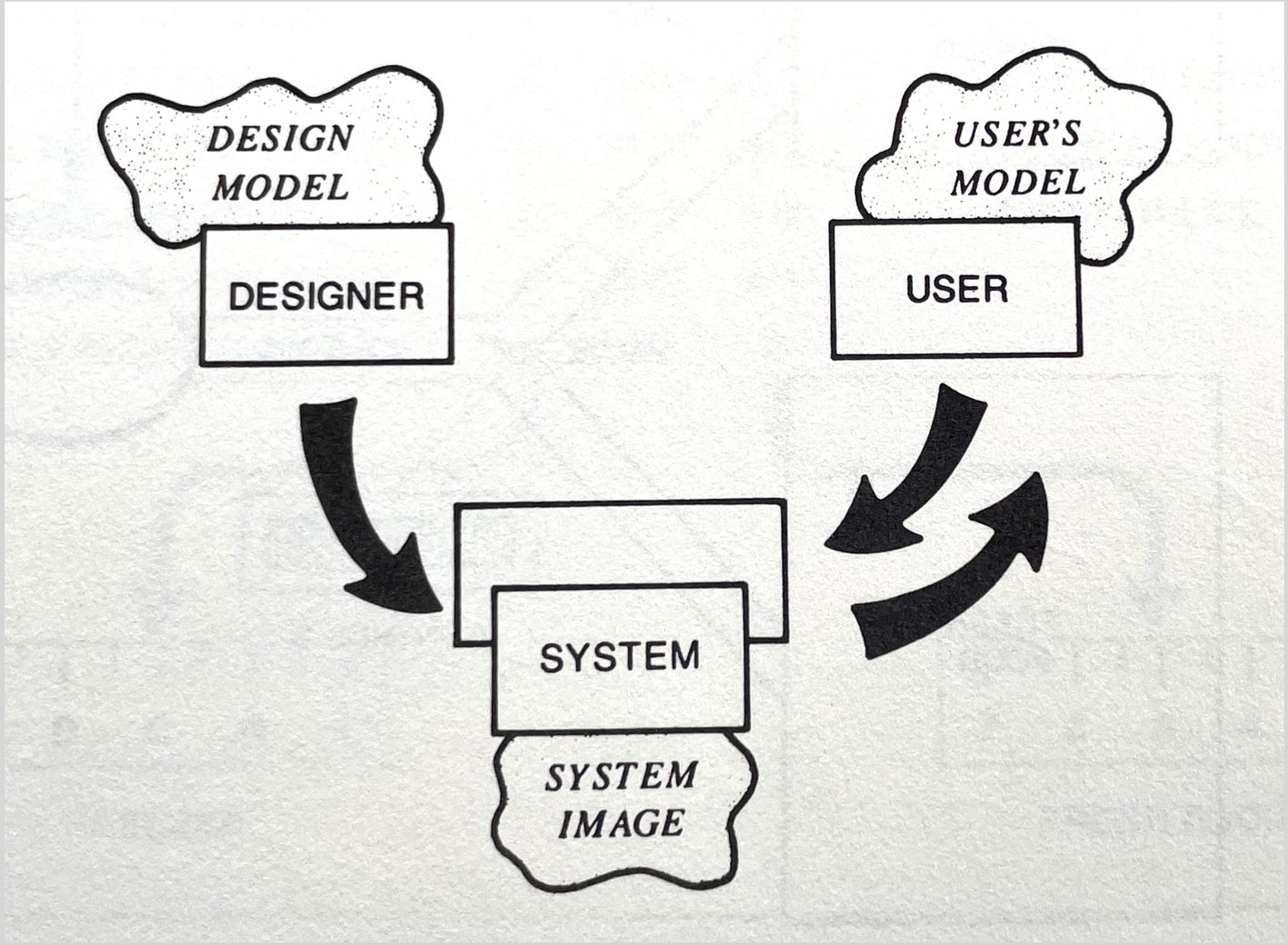


When the designers fail to provide a conceptual model, we will be forced to make up our own, and the ones we make up are apt to be wrong.
Conceptual models are critical to good design.

preface to 2013 edition

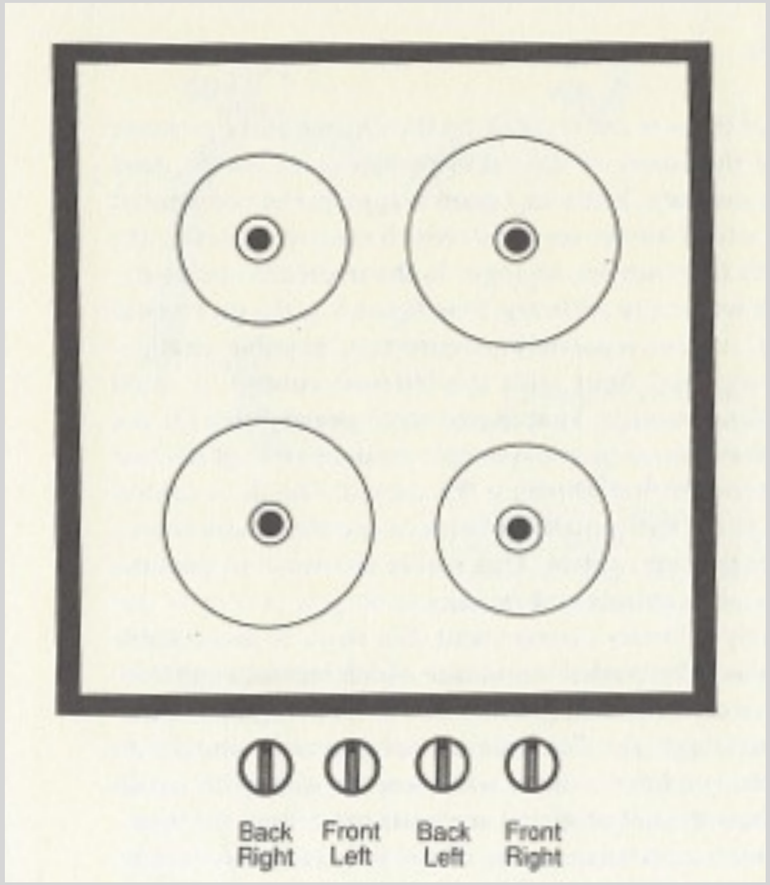
Donald Norman

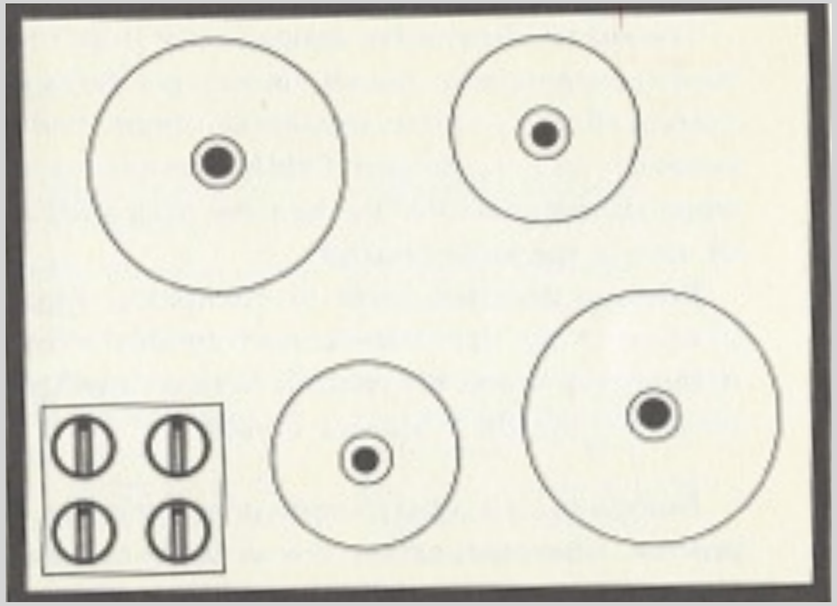
the “system image”



from *The Design of Everyday Things (1988)*

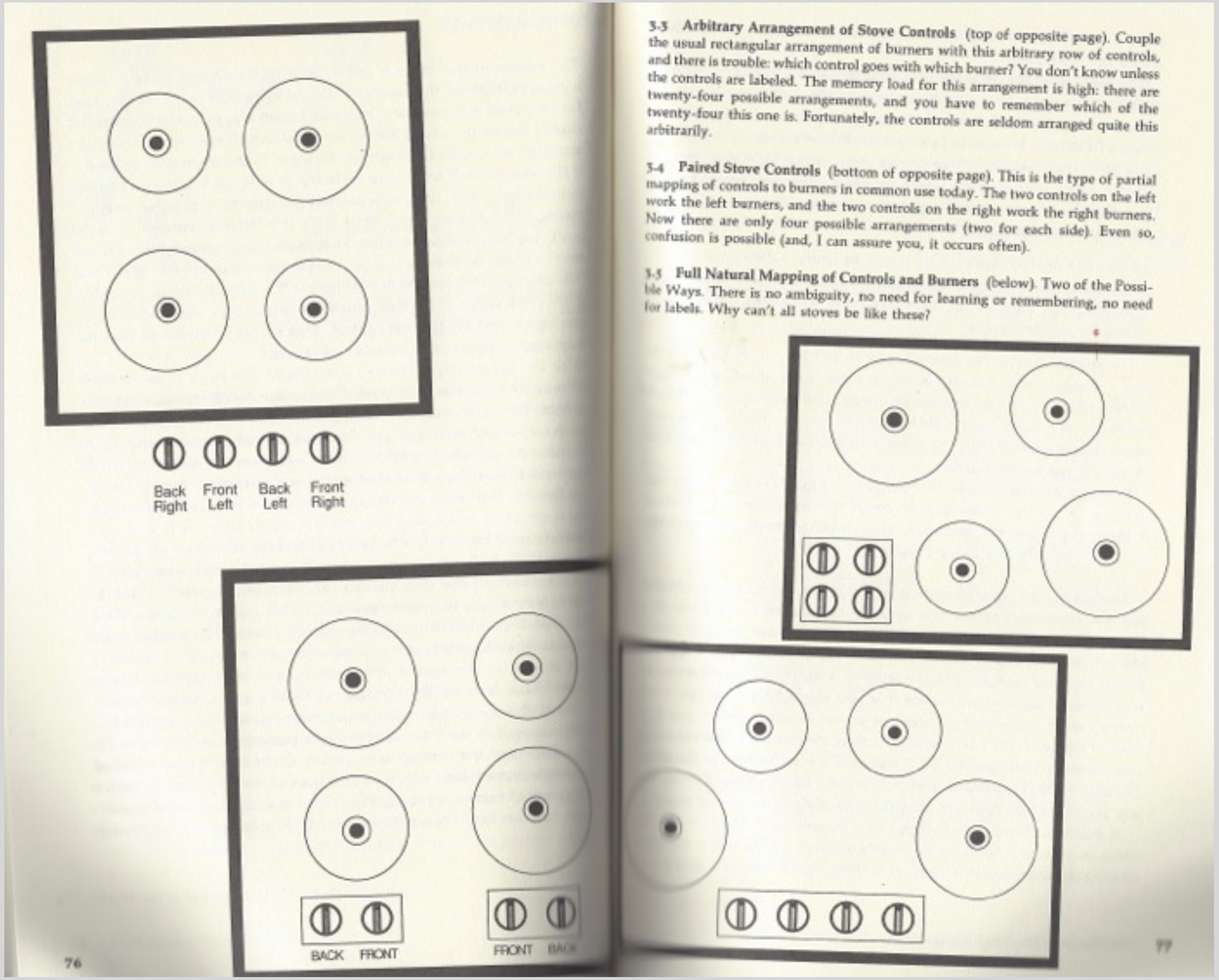
mapping: one strategy to improve the system image





from *The Design of Everyday Things (1988)*

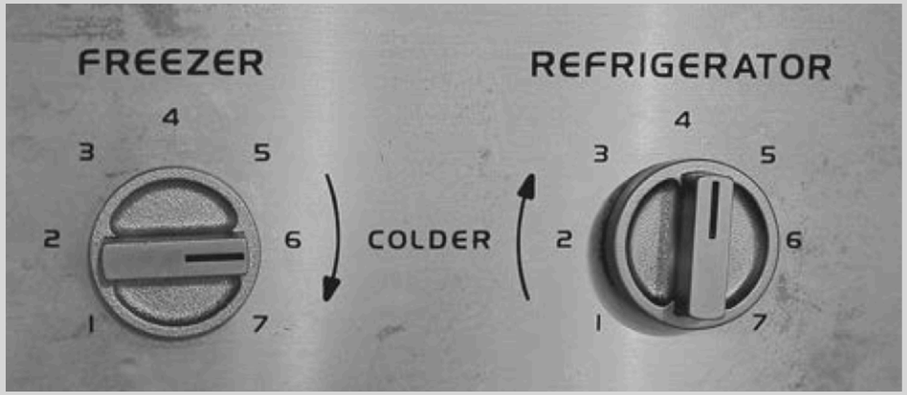
did the book designer read the book?



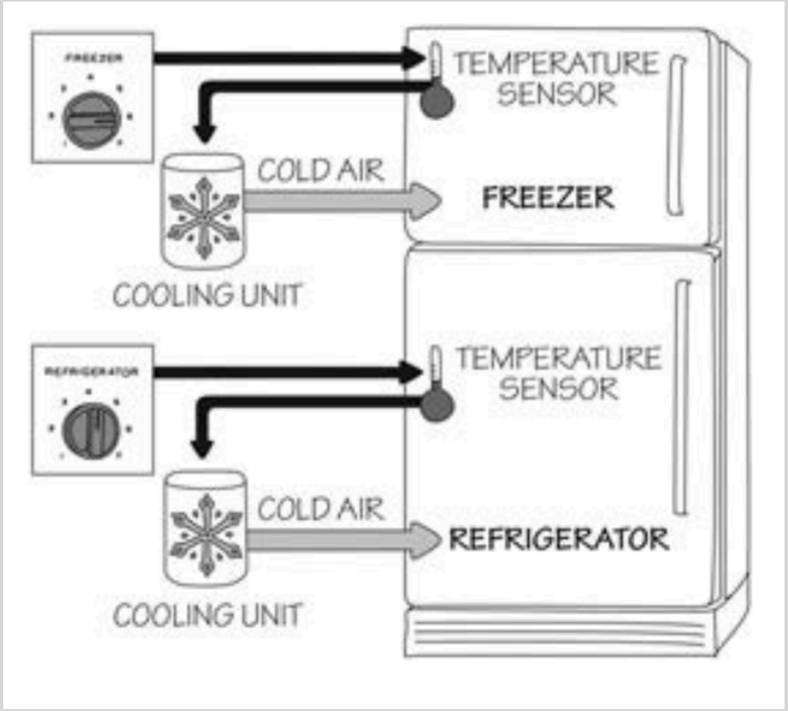
3-3 Arbitrary Arrangement of Stove Controls (top of opposite page). Couple the usual rectangular arrangement of burners with this arbitrary row of controls, and there is trouble: which control goes with which burner? You don't know unless the controls are labeled. The memory load for this arrangement is high: there are twenty-four possible arrangements, and you have to remember which of the twenty-four this one is. Fortunately, the controls are seldom arranged quite this arbitrarily.

3-4 Paired Stove Controls (bottom of opposite page). This is the type of partial mapping of controls to burners in common use today. The two controls on the left work the left burners, and the two controls on the right work the right burners. Now there are only four possible arrangements (two for each side). Even so, confusion is possible (and, I can assure you, it occurs often).

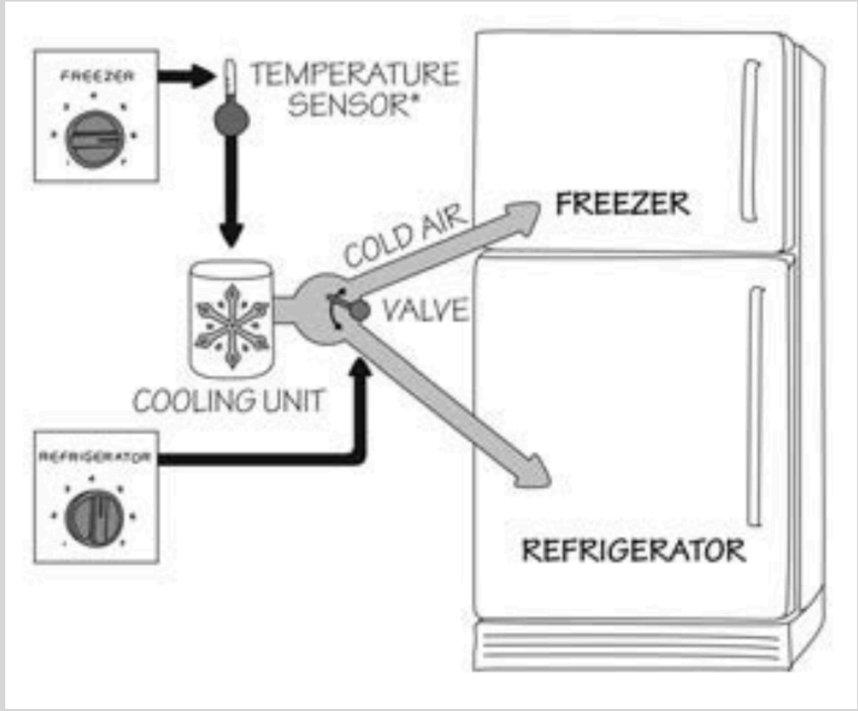
3-5 Full Natural Mapping of Controls and Burners (below). Two of the Possible Ways. There is no ambiguity, no need for learning or remembering, no need for labels. Why can't all stoves be like these?



typical controls on American fridge



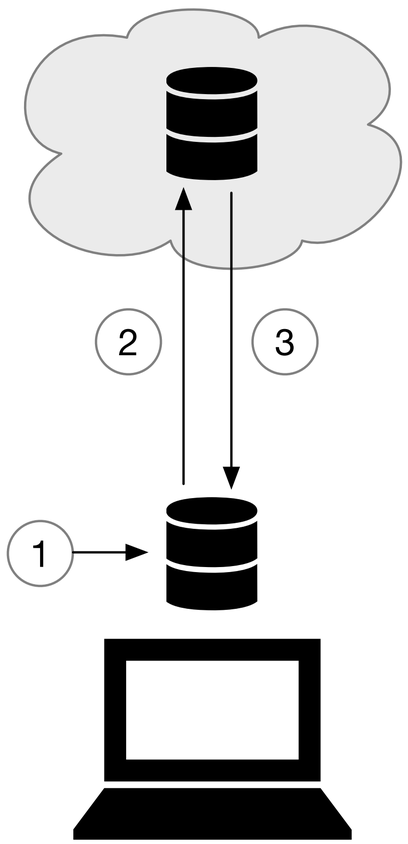
conceptual model (imagined)



conceptual model (actual)

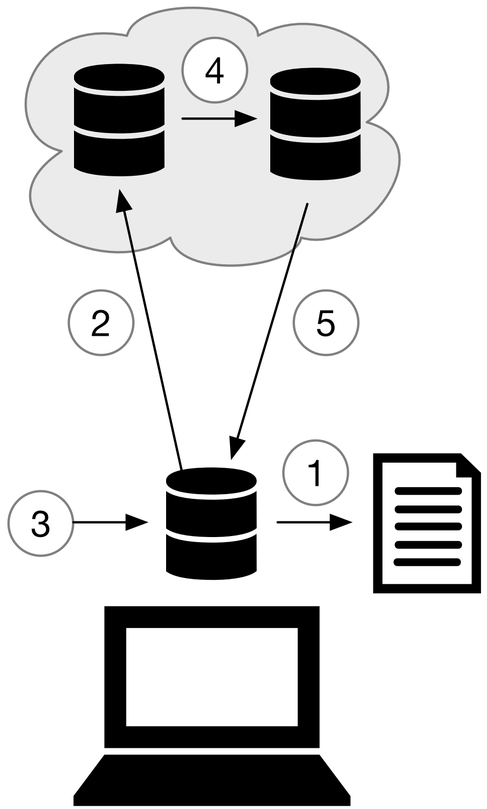
from Don Norman, Design of Everyday Things (2013)

Backblaze's conceptual model, real and imagined



"**continuous backup**"
what I imagined

Each time you modify a file (1), the modification is detected immediately and a snapshot of the new version of the file is taken and copied to the backup server (2), from where it is available for restoring (3).



"**continuous backup**"
what actually happens

Periodically, the backup utility scans the disk and makes a list of file modified since the last backup (1). It begins to copy files on this list to a special server (2). This process can take a long time, during which you might update additional files (3). When the backup is complete, at some later point the files are copied to a different server (4) from which they can be restored (5).

check your understanding

Which is most consistent with Don Norman's notion of conceptual models?

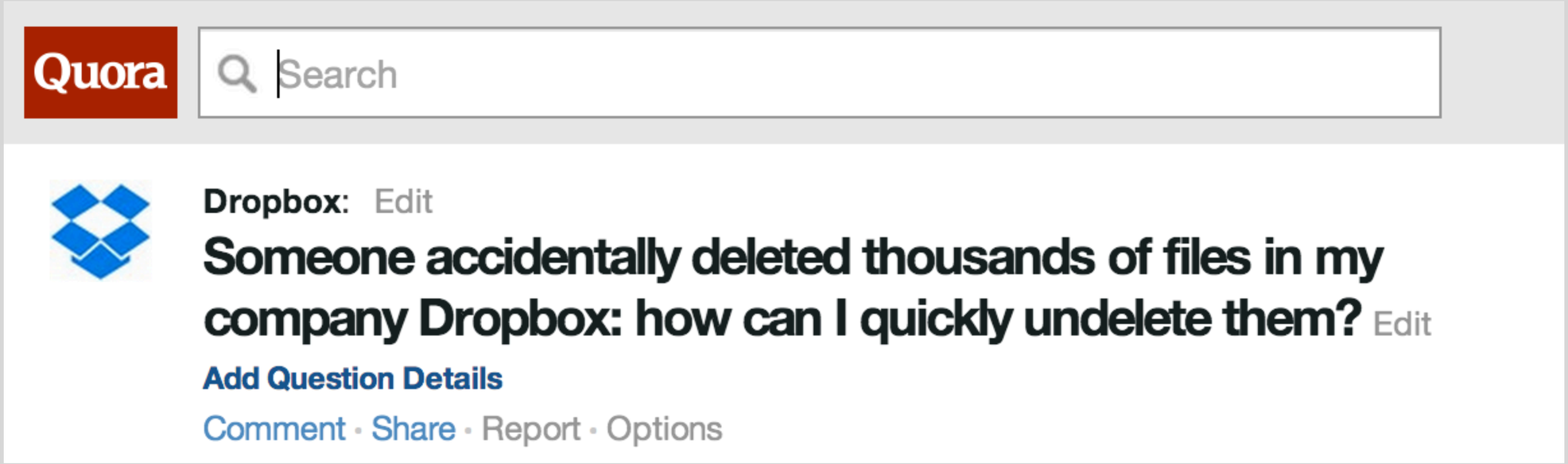
(a) The conceptual model *is* the design, and must be conveyed in the user interface

(b) A conceptual model is just an explanation of how a system works, so several are possible

(c) The conceptual model is just in the user's head, so it's OK for each user to invent their own

a UX puzzle

Dropbox



Sharing files with Dropbox (2021)

A family member of mine wanted to clear space on her drive

Listed very large files, and deleted ones she didn't recognize

Panicked message from colleague: where's our data?!

Ava is a party planner

Search

AA

Dropbox

Overview Show

Name Members

Bella Party 2 members

does the name change for Ava too?

answer: no, Ava sees no change

Bella is having a party

Search

BB

Dropbox

Overview Show

Name Members

My Party 2 members

Share

Download

Send with Transfer

Request files

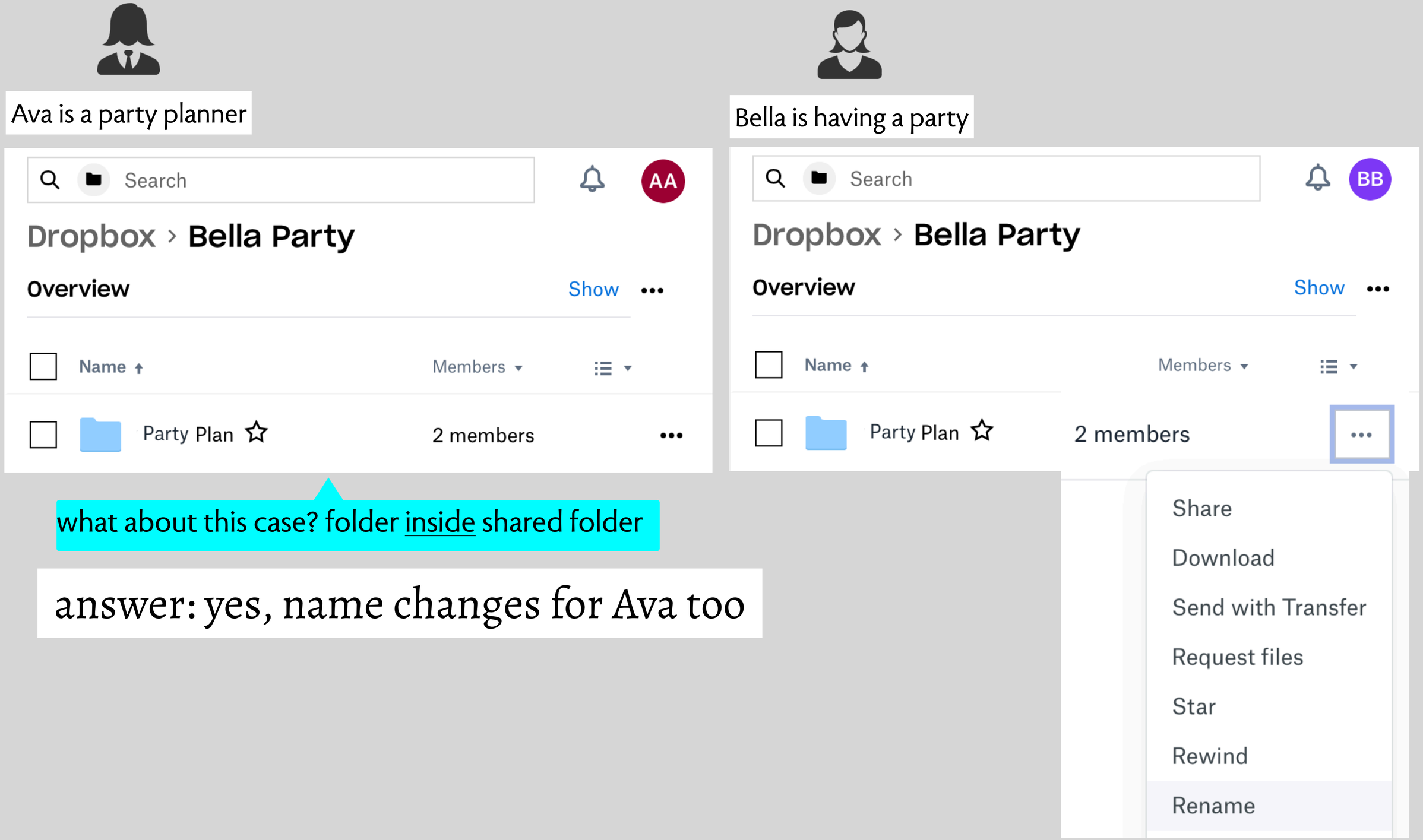
Star

Rewind

Rename

Ava is a party planner

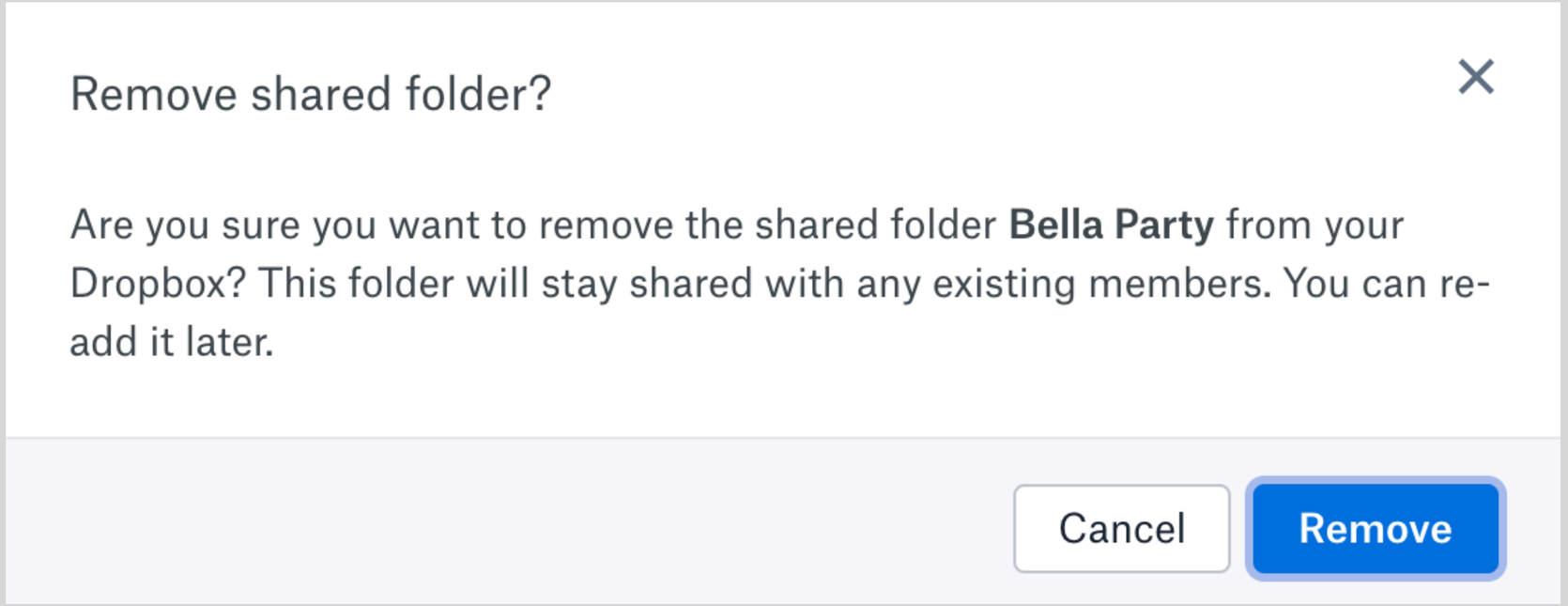
Bella is having a party



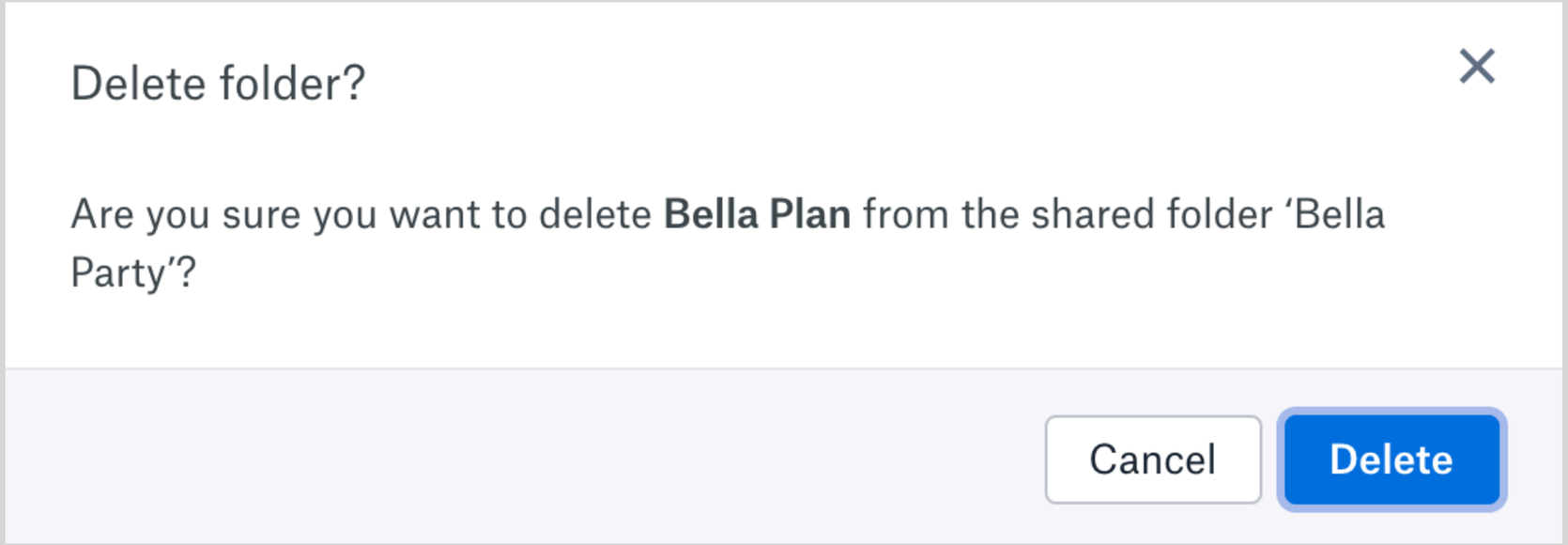
what about this case? folder inside shared folder

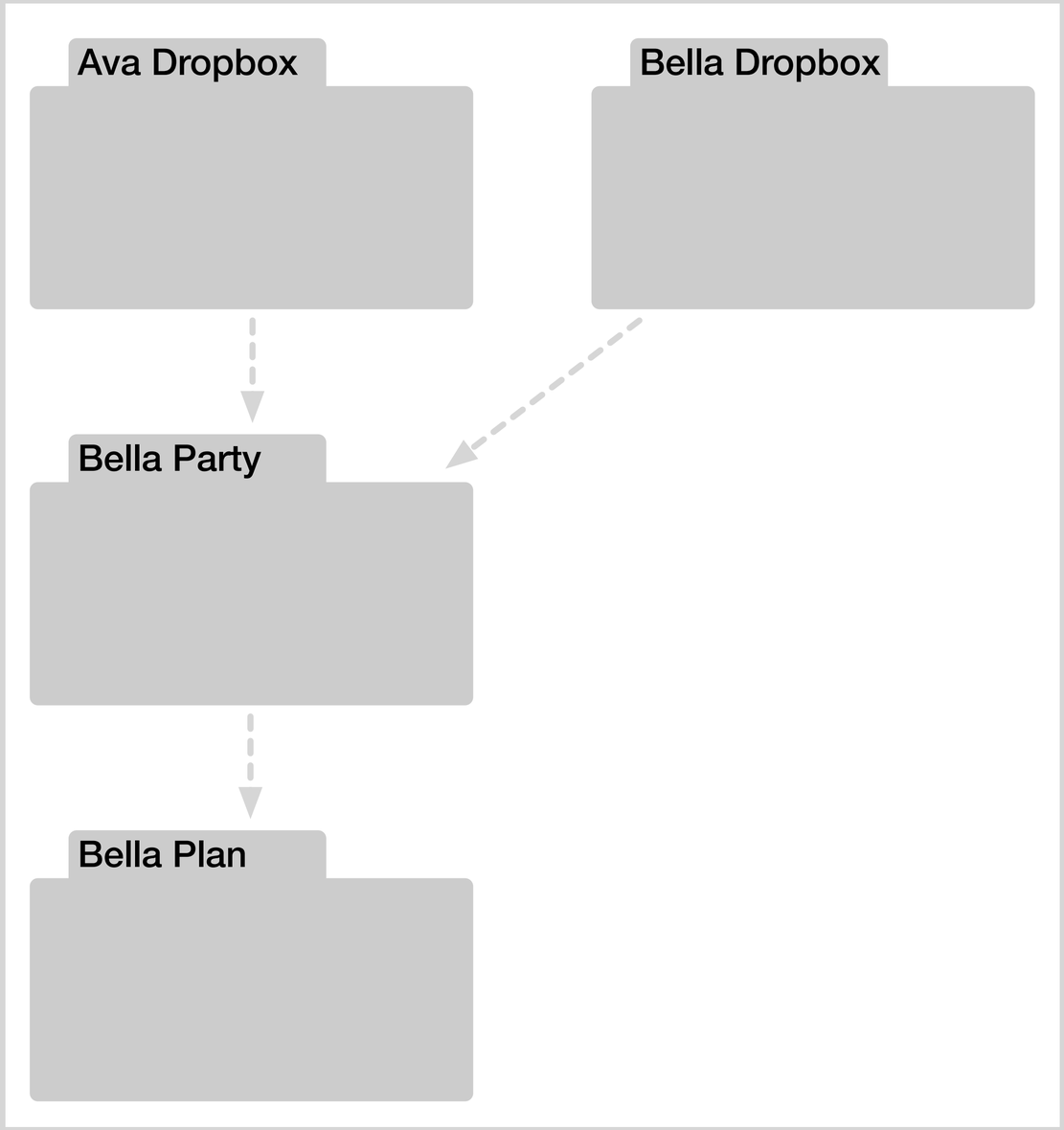
answer: yes, name changes for Ava too

Bella deletes shared folder Bella Party

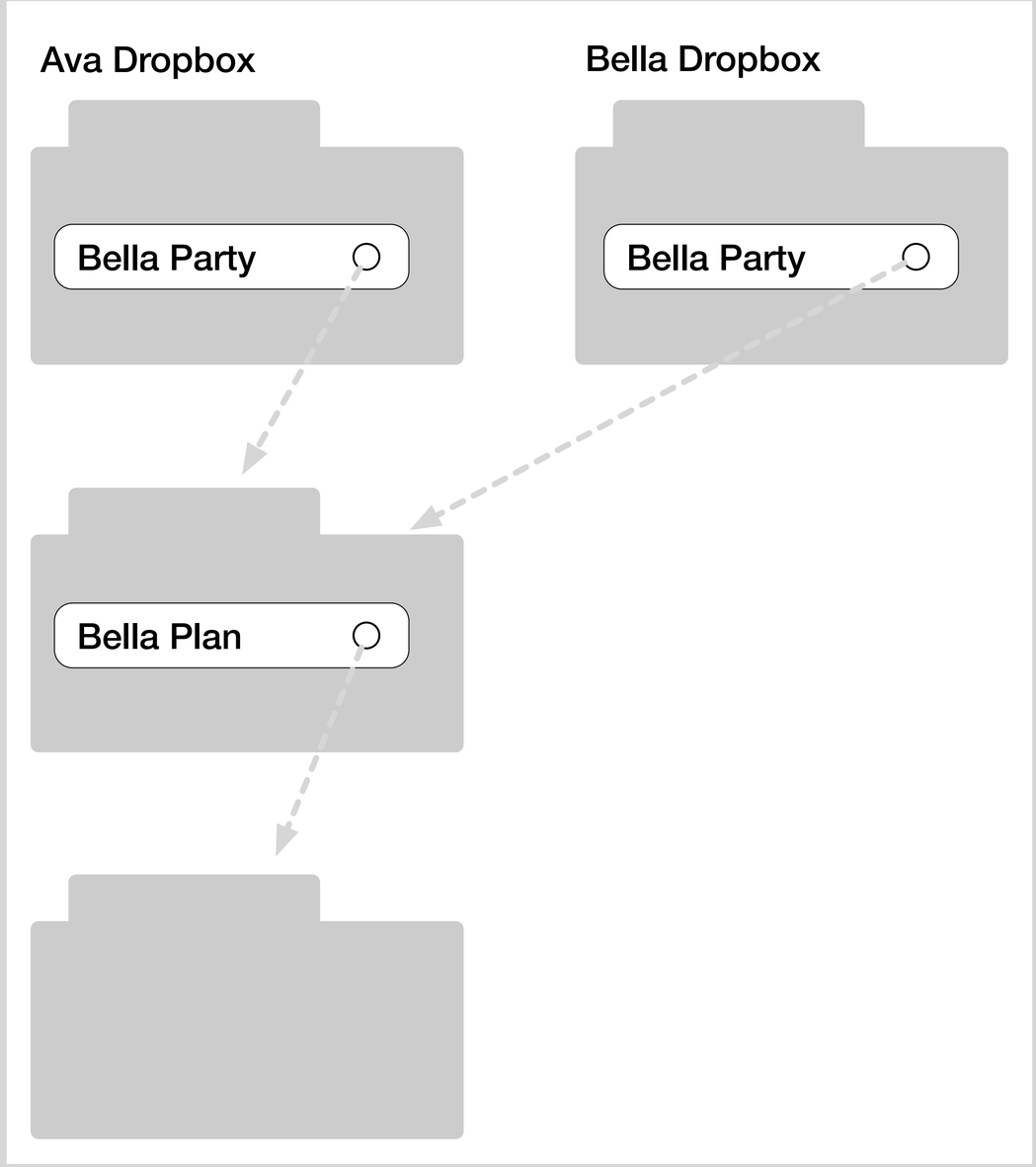


Bella deletes Bella Plan from shared folder Bella Party





how many users believe the folder concept works



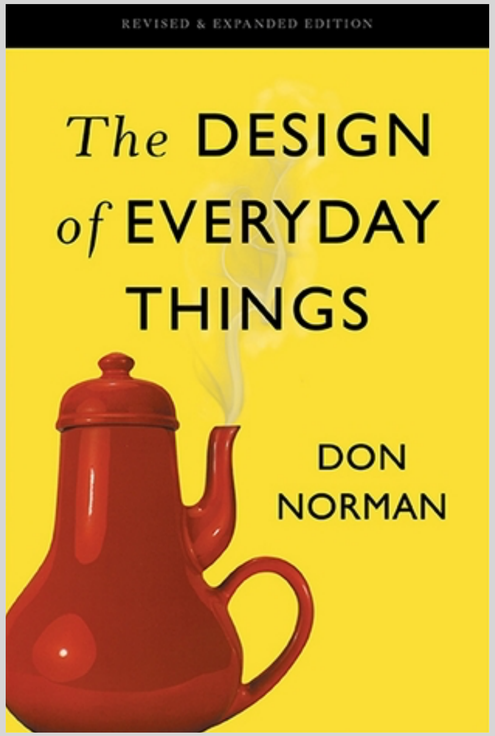
how folders actually work (in Dropbox, Unix, Multics)

where's the design?
interface or model?

user's model

classic usability research

designer's model



user's model

go this way instead

designer's model

but what if the designed model is just wrong?

user-centered design (1980s)

concepts are a **byproduct** of design

designer's job: **shape UI** to project concepts

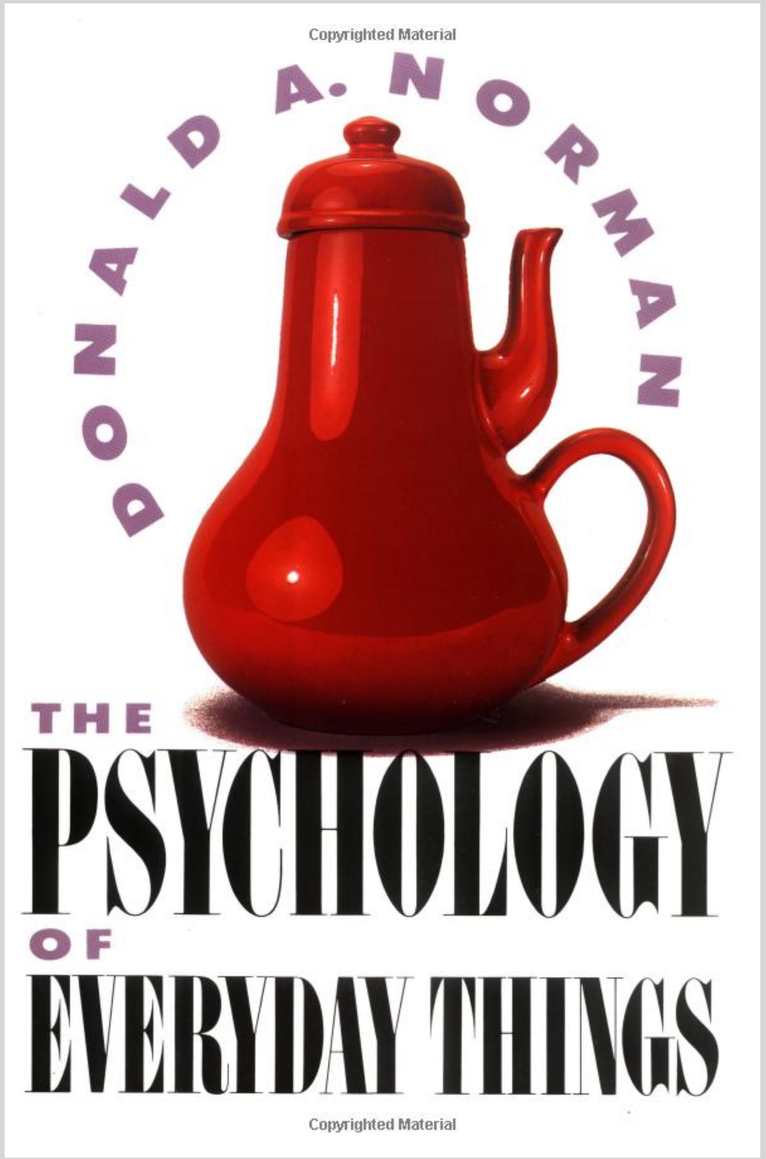
concepts are **psychological**

concept-based design

concepts are the **essence** of design

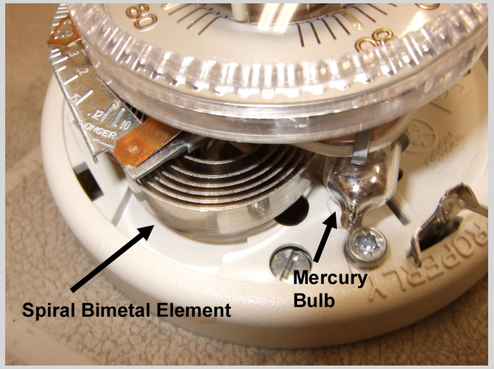
designer's job: **shape concepts**

concepts are **computational**



implications for UX designers

classic UX approach
underlying mechanism is fixed by engineers
UX designer's job is to create an **explanation**
UX is secondary to engineering



electrical-forensics.com

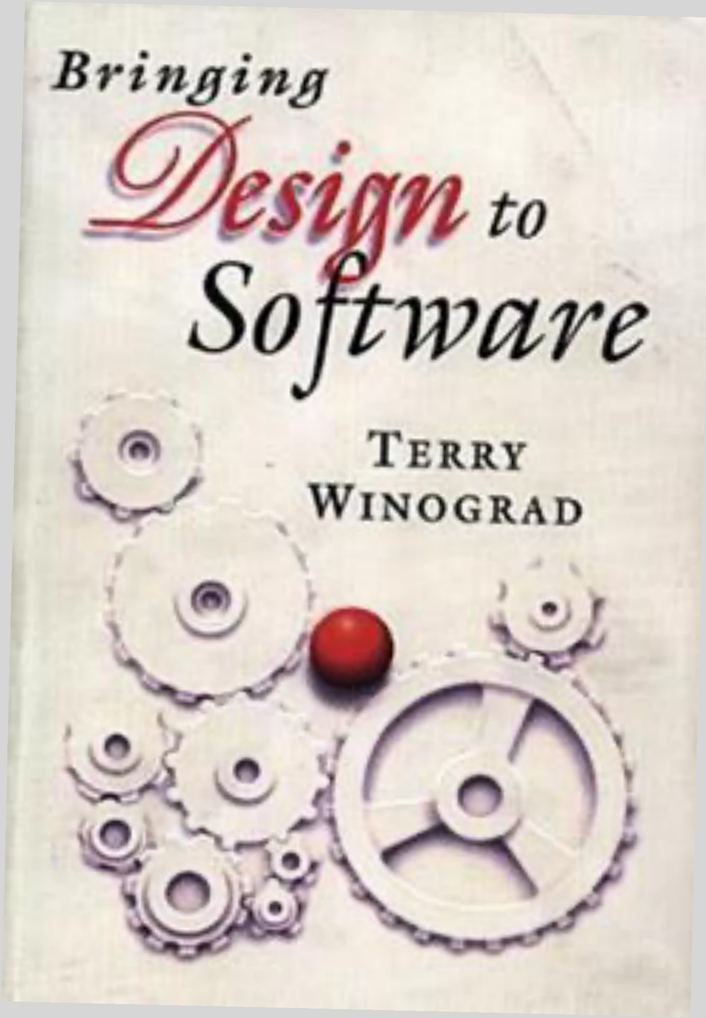
new UX approach
the conceptual model is the mechanism!
UX designer's job is to create **the model**
UX and engineering go hand-in-hand

When you go to design a house **you talk to an architect first**, not an engineer. Why is this?

Because the criteria for what makes a good building fall outside the domain of engineering.

Similarly, in computer programs, the **selection of the various components** and elements of the application must be driven by the conditions of use. **How is this to be done?** By software designers.

A Software Design Manifesto
Mitchell Kapor, 1996
paraphrasing slightly



are conceptual models real?

A conceptual model is an **explanation**, usually highly simplified, of how something works. It doesn't have to be complete or even accurate as long as it is useful. The files, folders, and icons you see displayed on a computer screen help people create the conceptual model of documents and folders inside the computer, or of apps or applications residing on the screen, waiting to be summoned. **In fact, there are no folders inside the computer—those are effective conceptualizations designed to make them easier to use.**

from *The Design of Everyday Things (1988)*

check your understanding

Which of these accurately describes concept design's view of conceptual models?

(a) The conceptual model represents the designer's view and not the user's view

(b) The conceptual model is inseparable from the design of the underlying mechanism

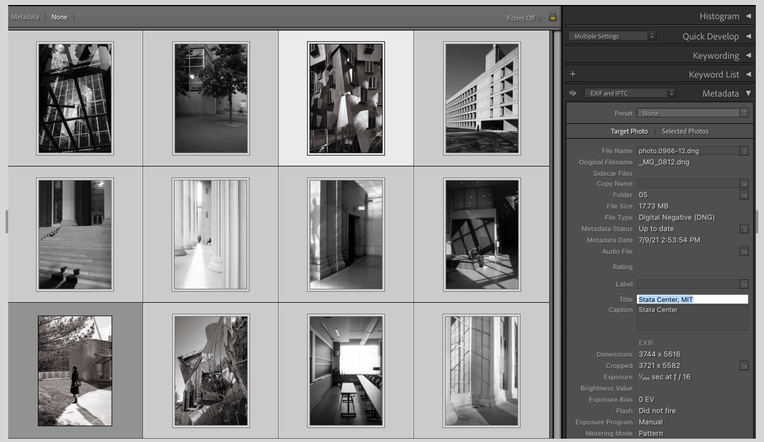
(c) The conceptual model is based more on engineering concerns than the user's experience

why concepts?
finding granularity

a limitation of UI-driven UX

if the user interface is the focus of our attention,
how can we ask if it projects the right concepts?
and what if the underlying concepts are wrong?

designer's concept



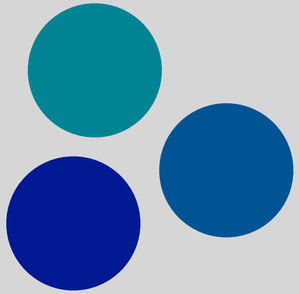
user's concept

essential features of a conceptual model

user facing
what the user experiences
no invisible implementation

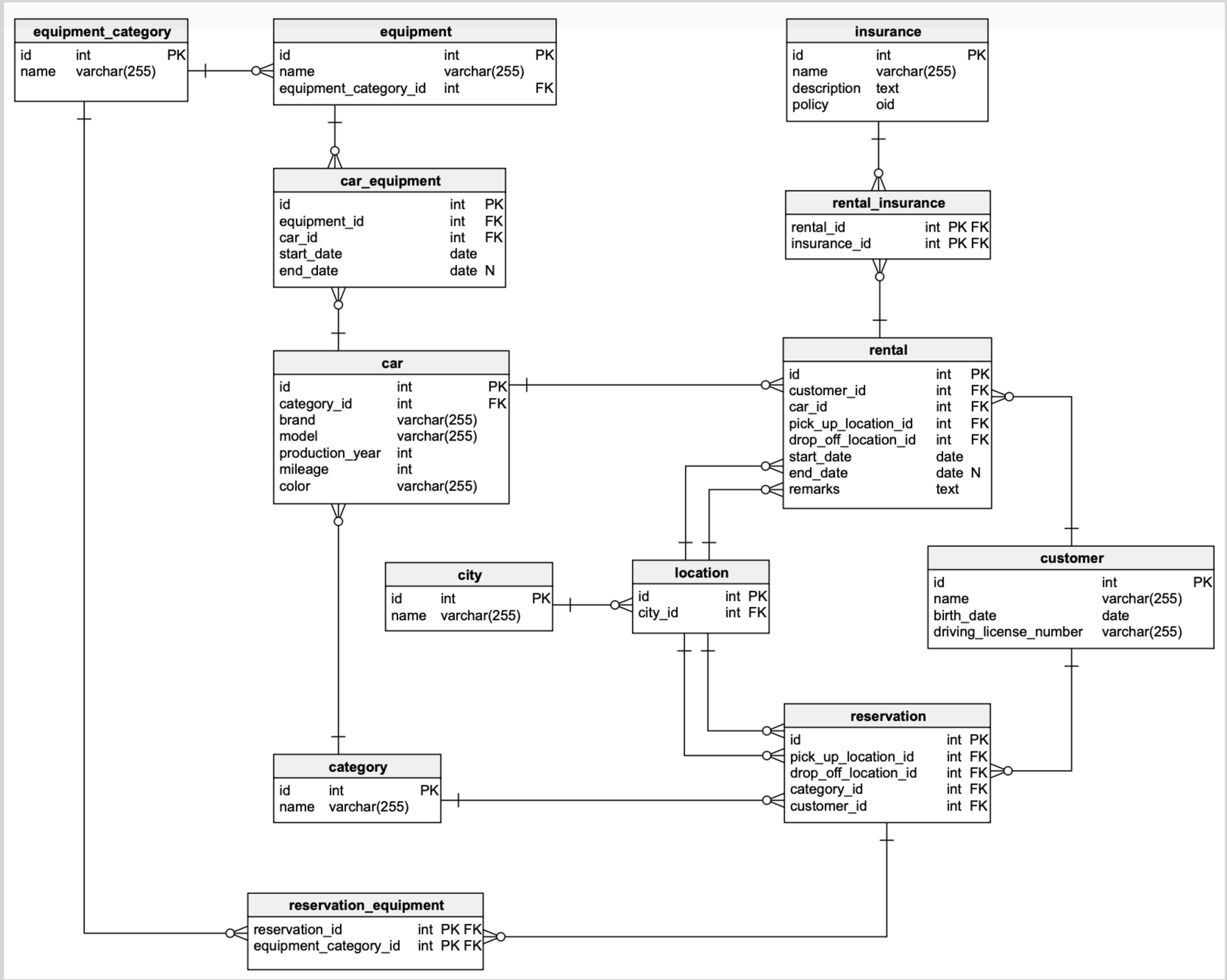
clear & precise
know what it means & implies
can specify details if we want

why **granularity matters**
incremental work
division of labor
reuse of prior knowledge
familiarity for users



granular
separable components
independently grasped

a conceptual model for car rental (entity-relationship diagram)

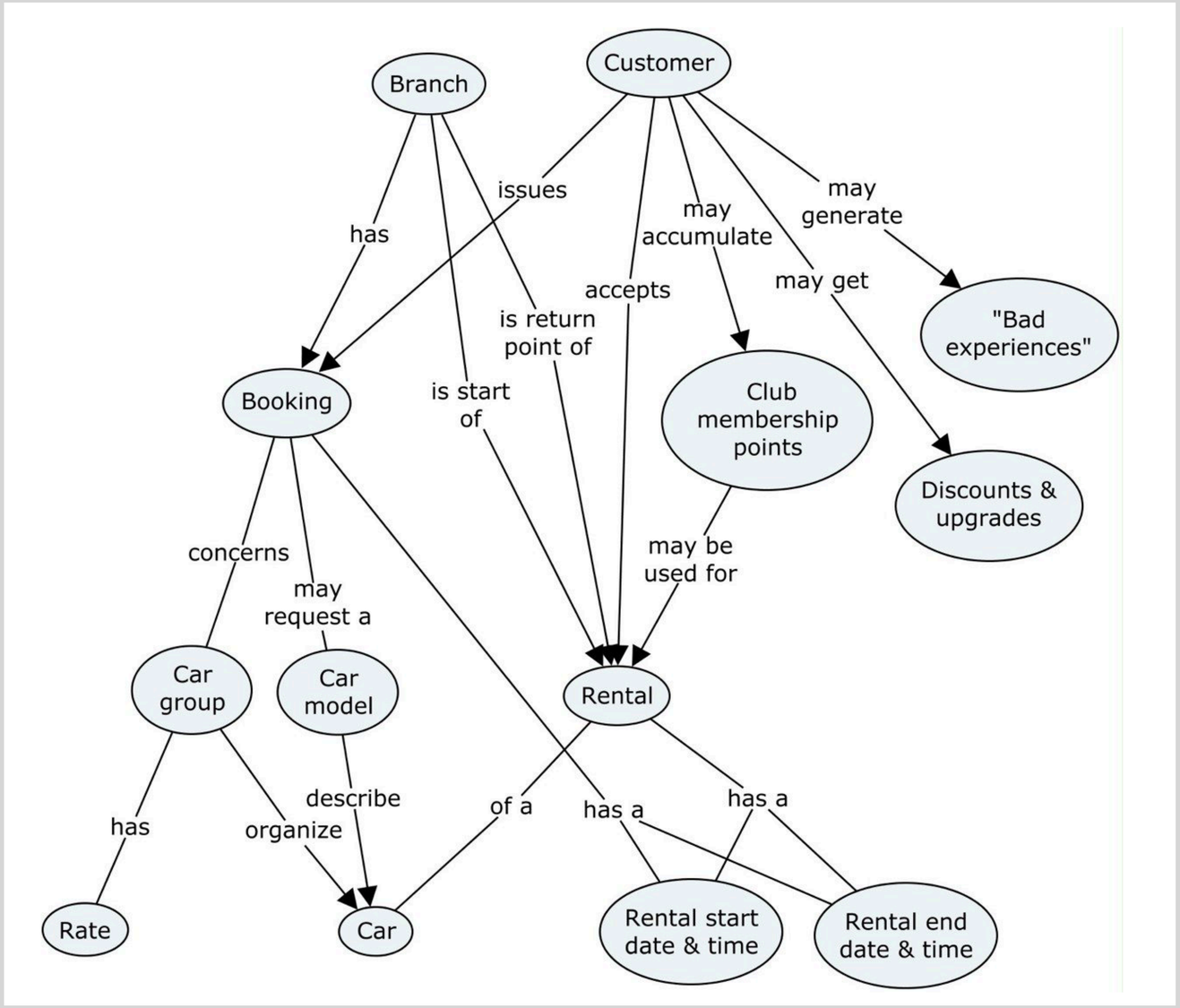


user facing?
more than code
but implementation details

clear & precise?
yes for structure
less clear for UX

granular
too granular!
entities aren't concepts

another conceptual model for car rental (concept diagram)



user facing?
yes

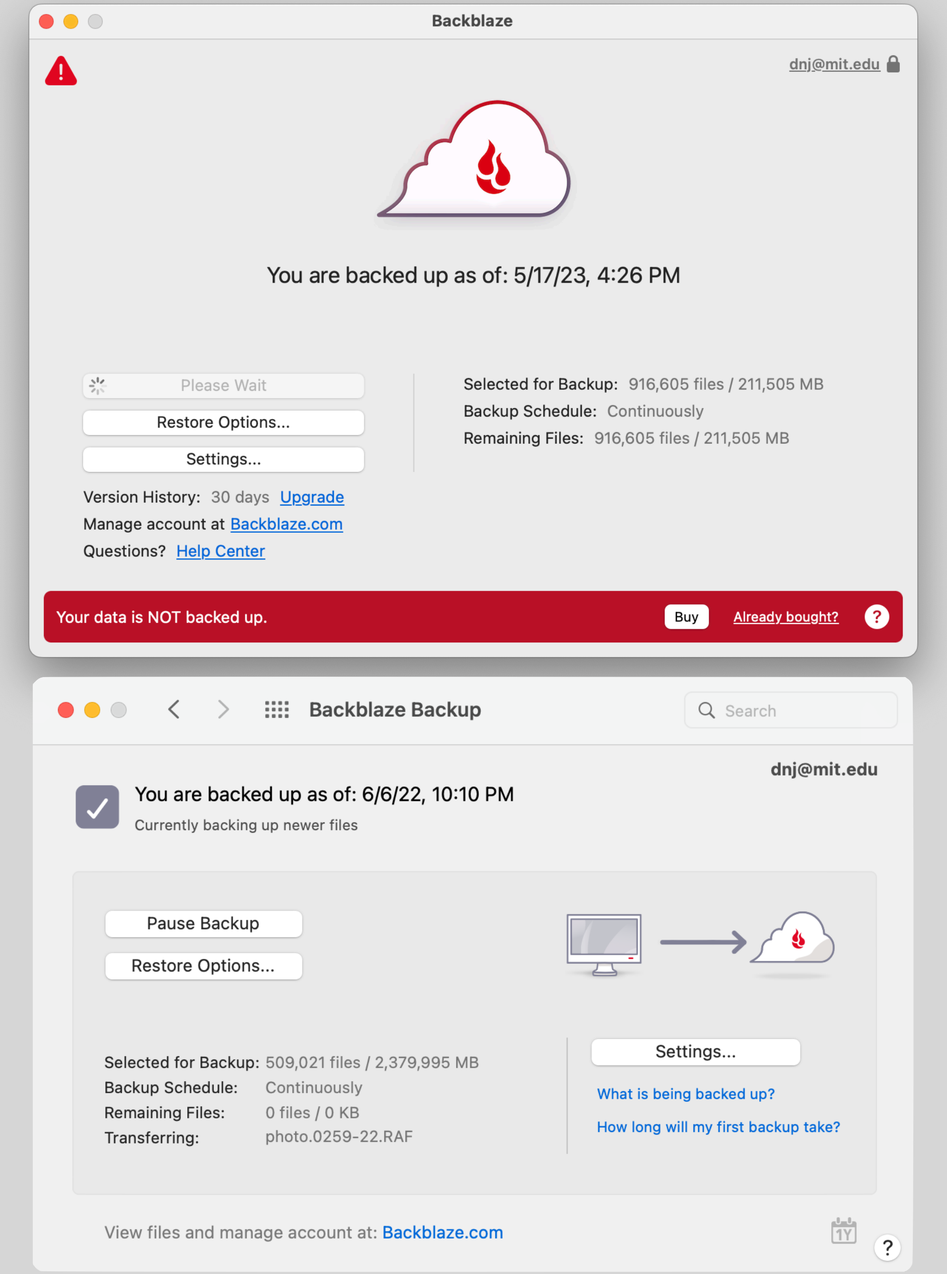
clear & precise?
no, very vague in places
Booking concerns Car group?

granular
better, but not quite right
what does Booking involve?

explaining Backblaze with concepts

Periodically, the backup utility scans the disk and makes a list of file modified since the last backup (1). It begins to copy files on this list to a special server (2). This process can take a long time, during which you might update additional files (3). When the backup is complete, at some later point the files are copied to a different server (4) from which they can be restored (5).

a scenario that conflates concepts



Backblaze is a **paid service** (so when service period has expired, may still see results of previous periods)

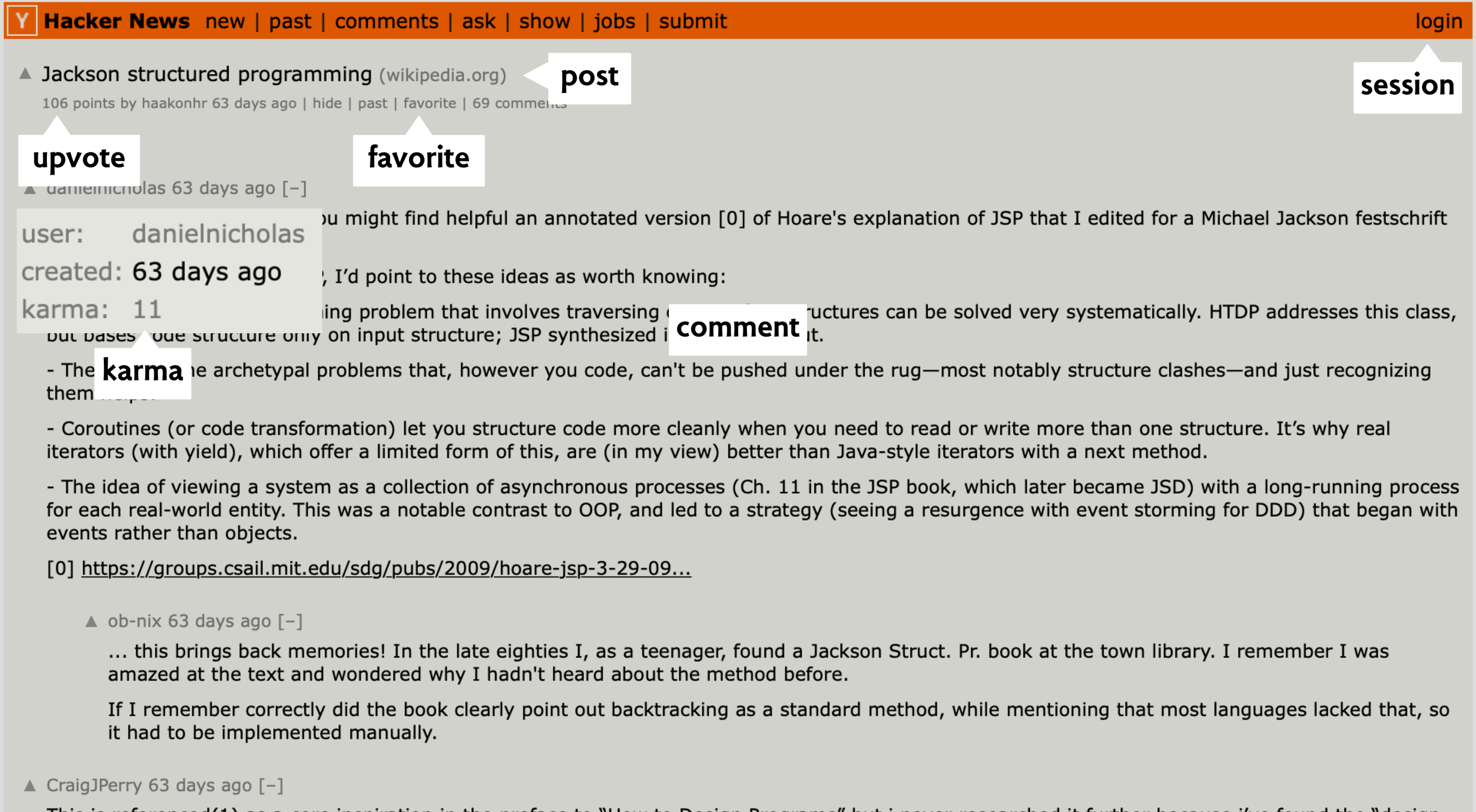
Backblaze uses a **write-only filestore** to maintain backups of your files

To backup the files, Backblaze runs a periodic batch task that creates a **worklist** of modified files.

separating out concepts

so what's a concept?

defining structure


Hacker News new | past | comments | ask | show | jobs | submit
login
Jackson structured programming (wikipedia.org)
post
106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments
session
upvote
favorite
danielnicholas 63 days ago [–]
user: danielnicholas
created: 63 days ago
karma: 11
u might find helpful an annotated version [0] of Hoare's explanation of JSP that I edited for a Michael Jackson festschrift
, I'd point to these ideas as worth knowing:
ing problem that involves traversing uctures can be solved very systematically. HTDP addresses this class,
comment
but bases ode structure only on input structure; JSP synthesized i t.
karma
- The e archetypal problems that, however you code, can't be pushed under the rug—most notably structure clashes—and just recognizing them
- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators (with yield), which offer a limited form of this, are (in my view) better than Java-style iterators with a next method.
- The idea of viewing a system as a collection of asynchronous processes (Ch. 11 in the JSP book, which later became JSD) with a long-running process for each real-world entity. This was a notable contrast to OOP, and led to a strategy (seeing a resurgence with event storming for DDD) that began with events rather than objects.
[0] https://groups.csail.mit.edu/sdg/pubs/2009/hoare-jsp-3-29-09...
ob-nix 63 days ago [–]
... this brings back memories! In the late eighties I, as a teenager, found a Jackson Struct. Pr. book at the town library. I remember I was amazed at the text and wondered why I hadn't heard about the method before.
If I remember correctly did the book clearly point out backtracking as a standard method, while mentioning that most languages lacked that, so it had to be implemented manually.
CraigJPerry 63 days ago [–]
This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but i never researched it further because i've found the "design

concept elements: name, purpose, principle

concept Upvote

purpose rank items by popularity

principle after series of votes of items, the items are ranked by their number of votes

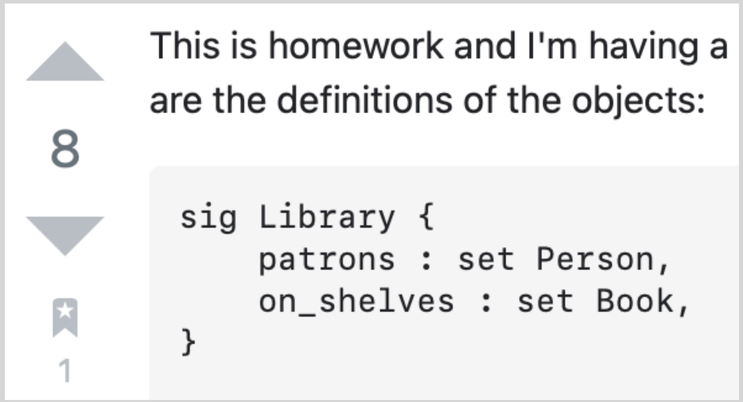
Michael Polanyi (1891-1976)

similar UIs, very different concepts

concept Upvote

purpose rank items by popularity

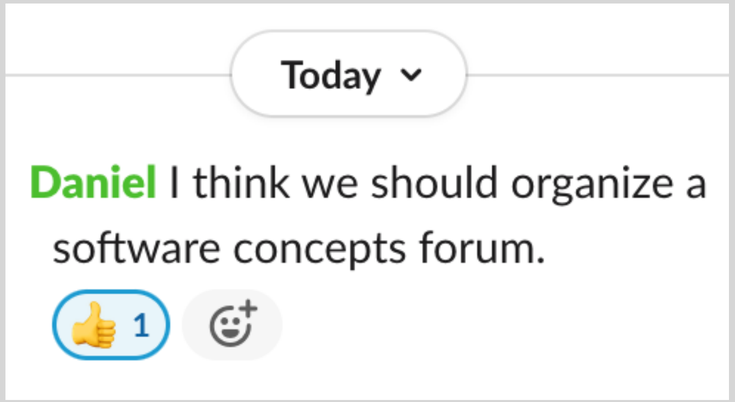
principle after series of votes of items, the items are ranked by their number of votes



concept Reaction

purpose send reactions to author

principle when user selects reaction, it's shown to the author (often in aggregated form)



concept Recommendation

purpose use prior likes to recommend

principle user's likes lead to ranking of kinds of items, determining which items are recommended

defining concept behavior in detail

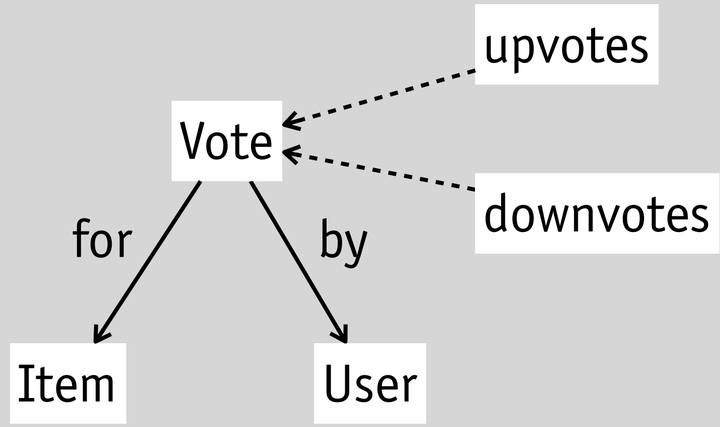
concept Upvote

purpose rank items by popularity

principle after series of votes of items, the items are ranked by their number of votes

state
a set of upvotes
a set of downvotes
for each vote
  the user it is by
  the item it is for

actions
upvote (u: User, i: Item)
downvote (u: User, i: Item)
unvote (u: User, i: Item)



downvote (i: Item, u: User)
requires no downvote by u for i
ensures
  remove any existing votes by u for i
  add a downvote by u for i

concepts as carriers of design knowledge

concept: Upvote

related concepts
Rating, Recommendation, Reaction, ...

design variants
downvote as unvote
use age in ranking
weigh downvotes more
various identity tactics
freezing old posts

known issues
high votes can promote old content
feedback favors early upvotes
upvoting encourages echo chamber
preventing double votes

typical uses
social media posts
comments on articles
Q&A responses

often used with
Karma, Auth, ...

check your understanding

Which of these best describes concepts?

(a) Concepts are user interface components that serve a particular purpose

(b) Concepts are like entities (such as votes) that have particular properties

(c) Concepts are more like services that manage a piece of functionality

concepts as atoms

apps as molecules

concepts as application ingredients

StackExchange = Q&A + Comment + Upvote + Reputation + Moderator + Bounty + ...

HackerNews = Post + Comment + Upvote + Karma + Favorite + Hiding + Flag + ...

Facebook = Post + Comment + Reel + Upvote + Reaction + Friend + Follow + Feed + Tag + ...

Instagram = Post + Comment + Reel + Upvote + Follow + Feed +...

Concepts common to almost all apps
Notification, UserAuthentication, Session

Concepts common to all social media apps
Post, Comment, Upvote

Variant concepts with related function
Reputation (StackExchange) vs Karma (HackerNews)
Friend (Facebook) vs Follow (Instagram)

Differentiator concepts
IPbasedAuth (Wikipedia), Duet (TikTok),
Bounty (StackExchange), Carousel (Instagram)

what else makes an app?

some app-specific details of HackerNews

a post can contain a URL, a title and some text

if a post's title contains “AskHN”, it can’t have a URL

suggest the Post concept is special

you can delete a post only if you're the author

you can’t comment on a post after two weeks

you can’t edit or delete a post after two hours or if it has comments

you’re rewarded 10 karma points when someone upvotes your post

you can’t downvote a post unless you have 20 karma points

all just interaction between concepts

this is what Margaret Boden called “combinatorial creativity”: familiar elements combined in new ways

familiar concepts are like a vocabulary: these ”what else” properties arguably define HackerNews!

synchronization examples

when a post is upvoted, the user is the active user in the session

when
WebRequest (“upvote", post, session)
where
active user is u (in Session state)
then
Post.upvote (post, u)

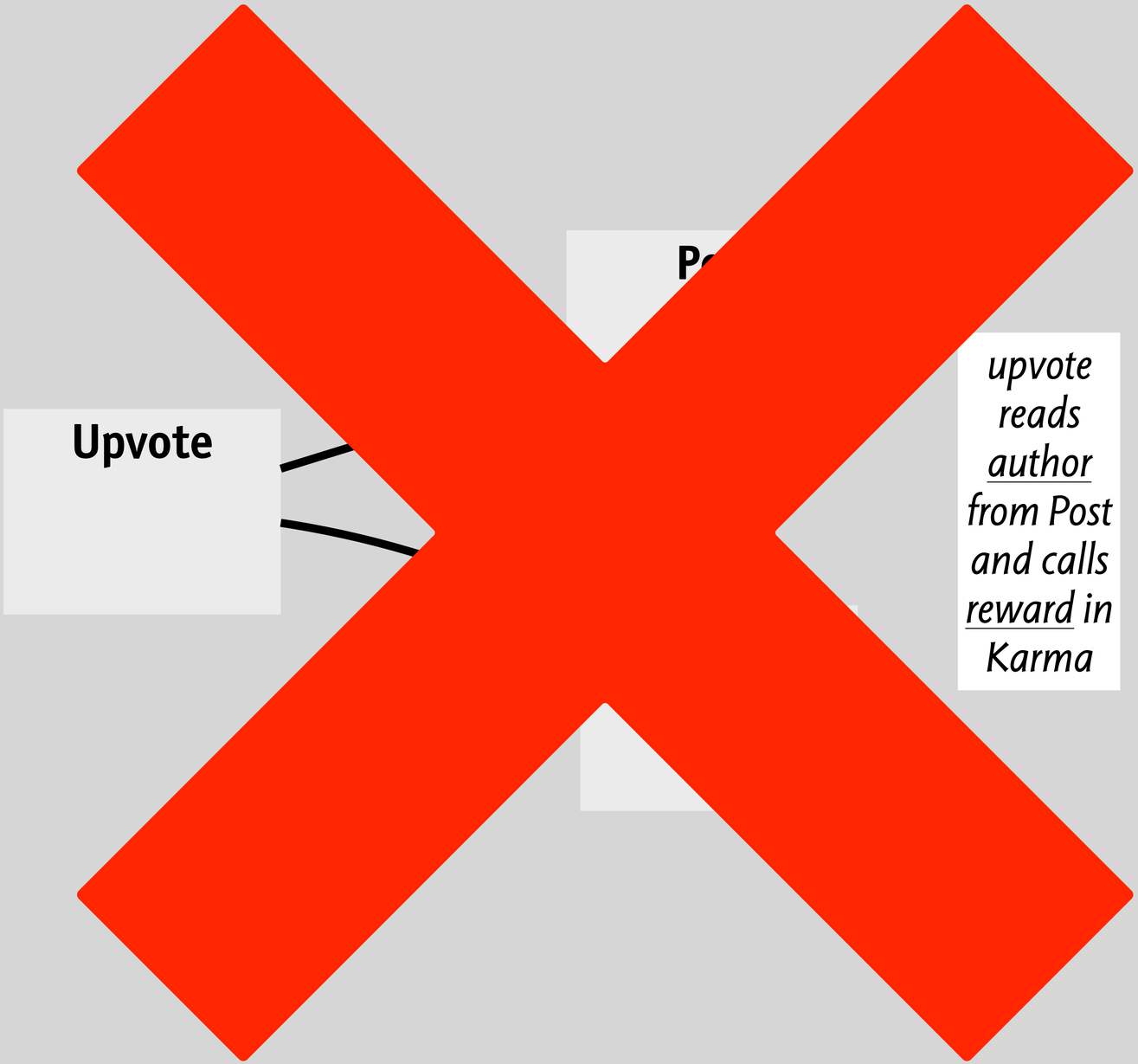
you can’t downvote a post unless you have 20 karma points

when
WebRequest (downvote, post, session)
where
active user is u (in Session state)
u has at least 20 points (in Karma state)
then
Post.downvote (post, u)

you’re rewarded 10 karma points when someone upvotes your post

when
Post.upvote (post, user)
where
author of post is u (in Post state)
then
Karma.reward (u, 10)

how concepts do not interact



concept never:
call each other's actions
read or write each other's state

no "shared objects"
Post concept creates posts
Post concept associates authors with posts
UserAuth concept creates users
Upvote concept has references to posts
Upvote concept has references to users
Upvote concept associates posts with users

the secret sauce that makes concepts modular
completely independent of each other

takeaways

what you learned today

what you learned today

conceptual model
a shared understanding of function (user, designer, engineer, ...)
the focus of design work, not just an explanation that comes later

but how to design one?
concepts bring: clarity, precision & granularity
name / purpose / operational principle
designing concepts vs. designing synchronizations

what I hope you can now do

identify concepts
in an application, in your work

start to talk about design using concepts
which concept is familiar? product-defining? troubled?

what's next?

what's next?

homework #1: post to our Slack group
what one idea did you find most useful, surprising, confusing?

homework #2: post to our Slack group
a very brief concept description (from your Autodesk work, Slack, etc)
name + purpose + something surprising or unusual about it

plans for next sessions
how to define a concept's behavior precisely & succinctly
how to break a system into concepts