concept design
part 2: behavior

Daniel Jackson · Autodesk Online Workshop · June 2025

on details

The details are not details. They make the design. *Charles Eames*

13721
Eames Contract Storage
1507 Folding bed unit; light shelf, bed platform, reading light. Accessories: mattress.
HERMAN MILLER INC.
ES 103

what kind of behavioral details?

	for online bookstore, eg
details to include	
steps the user takes	buy a book
system responses to the user	book gets delivered
data the user gives & gets	address, arrival estimate
details to exclude	
coding & algorithmic details	order id has checksum
distribution, replication, etc	orders on separate server
internal steps	request to warehouse

also UI independent
layout & styling of pages
navigation between pages
“micro-steps”

UI-dependent questions: important but not conceptual

how many steps to enter data?

should available slots be red?

is this helpful?

Terra - Eataly Boston

4.5 (3940) • $31 to $50 • Contemporary Italian

Overview | Experiences | Popular dishes | Photos

About this restaurant

Charming | Lively | Good for special occasions

Located on the third floor of Eataly Boston, Terra is a unique restaurant inspired by earth and fire. The dining room centers around a wood-burning Italian grill, where the Terra culinary team cooks raw ingredients over burning flames, allowing the...
Read more

Experiences

Brunch at Terra

Aug 22, 2024 - Jan 28, 2026

Every Saturday and Sunday from 11AM-4PM, indulge in our brunch menu featuring all your favorites...with an Italian...

Make a reservation

2 people

Jun 13, 2025 | 7:00 PM

Select a time

6:00 PM* | 6:15 PM* | 8:00 PM* | 9:00 PM*

+1,000 pts | | | +1,000 pts

Notify me

Booked 110 times today

You're in luck! We still have 4 timeslots left

Experiences are available. See details

Additional seating options

why postpone UI-dependent details?

they're a lot of work
we need to tend to
more basic things first

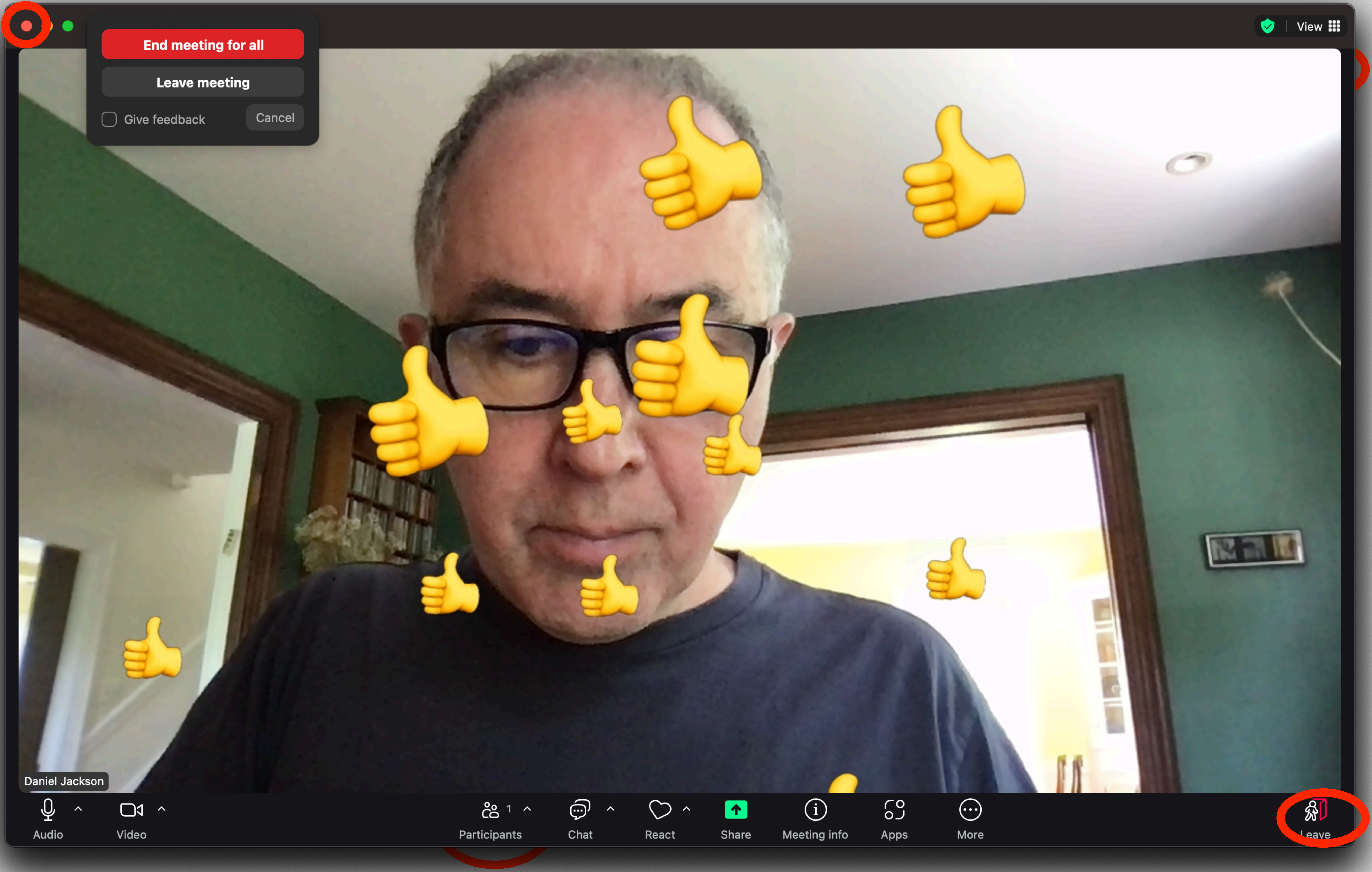
they can be a distraction
color of slots before we've
decided that we have slots?

want to judge a UI
projects concepts well?
then need pure concepts

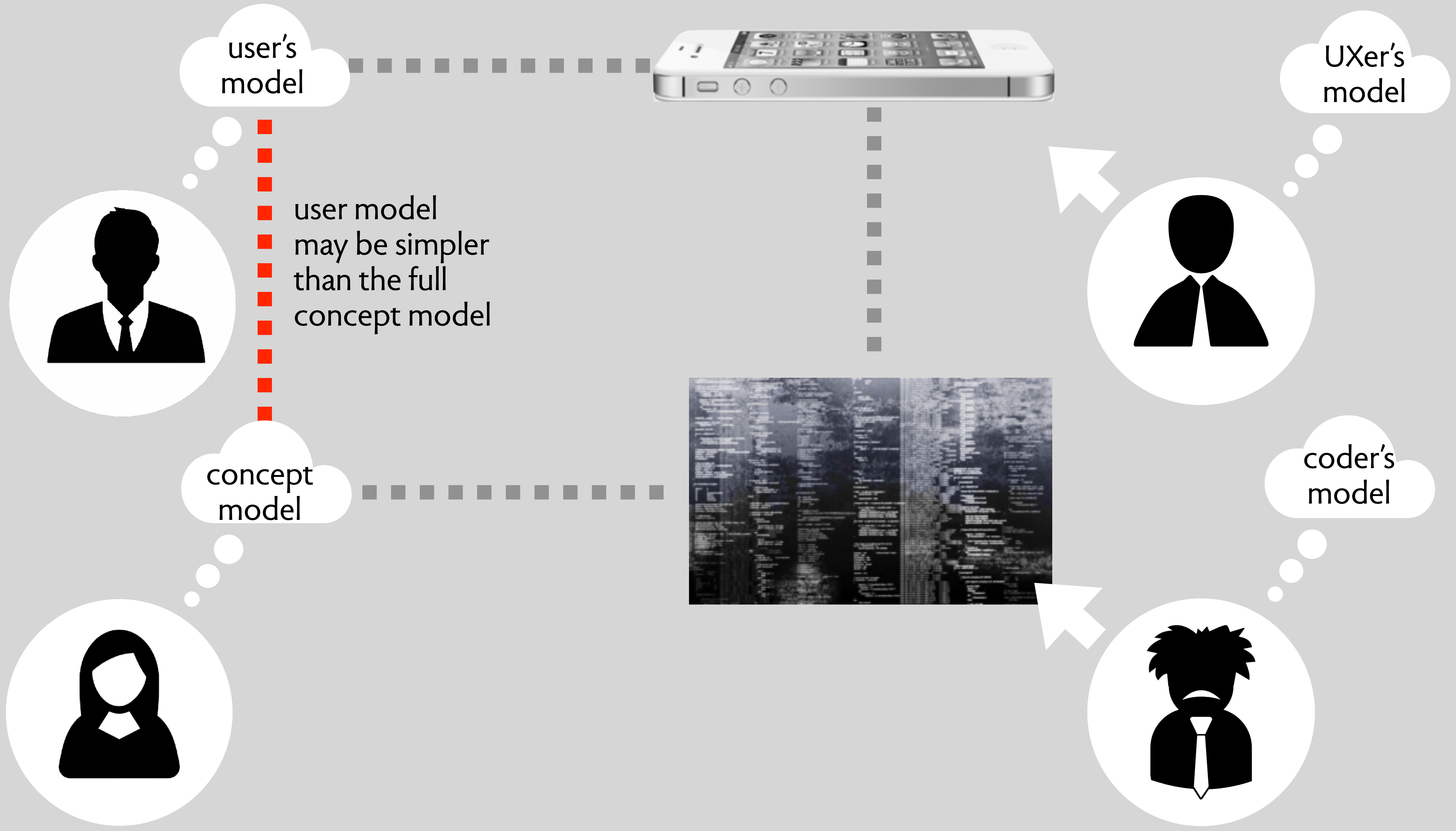
shared understanding
between UX & engineering
capturing the overlap

what this doesn't mean
can't sketch UI ideas
during concept design
often helpful to concretize

which steps are concept actions?



many models playing different roles



a full example

a reservation concept

how to design a concept

pick a name
specific to function
but for general use

describe purpose
why design or use it?
value to stakeholders

tell story
a simple scenario
of how it's used

list actions
by user or system
key steps, not UI

specify state
what's remembered
enough for actions

picking a name

pick a name
specific to function
but general enough

Restaurant

RestaurantReservation ✓

OpenTableReservation

Reservation

describing a purpose

describe purpose
why design or use it?
value to stakeholders

- reducing wait time for tables ✓
- maximizing use of available tables
- making money for reservation service
- tracking occupancy patterns

telling the story

the restaurant makes slots available at various times; a diner reserves for a particular time, and then can be assured of being seated at that time

✓

tell story
a simple scenario
of how it's used

listing actions

list actions
by user or system
key steps, not UI

select date
select time
click reserve

no! these are
all low-level
UI interactions

login
search for restaurant
review restaurant

no! these belong
to other concepts

let's return to our
story for hints:

the restaurant makes slots available at various times; a diner reserves for a particular slot, and then can be assured of being seated at that time

createSlot
reserve
seat

✓

what other actions
might be needed?

cancel

noShow

deleteSlot

defining action arguments

createSlot	createSlot (t: Time)
reserve	reserve (u: User, t: Time): Reservation
seat	seat (r: Reservation)
cancel	cancel (r: Reservation)
noShow	noShow (r: Reservation)
deleteSlot	deleteSlot (s: Slot)

devising the state

specify state
what's remembered
enough for actions

a set of slots each with
 the start time (includes date)
a set of reservations each with
 the user who made it
 the slot being reserved

defining the actions

state

a set of slots each with
 the start time (includes date)
a set of reservations each with
 the user who made it
 the slot being reserved
 whether seated

actions

createSlot (t: Time)
ensures
 creates a fresh slot
 associates it with time t

reserve (u: User, t: Time): Reservation
requires
 some slot at time t not yet reserved
ensures
 creates & returns a fresh reservation
 associates it with user u and the slot

"precondition"
what's true of state before

"postcondition"
relates state after to before

seat (r: Reservation)
requires
 r is a reservation for about now
ensures
 mark r as seated

state
a set of slots each with
 the start time (includes date)
a set of reservations each with
 the user who made it
 the slot being reserved
 whether seated

actions

createSlot (t: Time)
ensures creates a fresh slot
 associates it with time t

reserve (u: User, t: Time): Reservation
requires some slot at time t not yet reserved
ensures creates & returns a fresh reservation
 associates it with user u and the slot

seat (r: Reservation)
requires r is a reservation for about now
ensures mark r as seated

initially

slot	time

res	user	slot	seated

createSlot (July 4, 2025 at 7pm)

slot	time
s0	July 4, 2025 at 7:00pm

res	user	slot	seated

reserve (u1, July 4... 7pm): r0

slot	time
s0	July 4, 2025 at 7:00pm

res	user	slot	seated
r0	u1	s0	FALSE

seat (r0)

slot	time
s0	July 4, 2025 at 7:00pm

res	user	slot	seated
r0	u1	s0	TRUE

putting it all together

pick a name
specific to function
but for general use

describe purpose
why design or use it?
value to stakeholders

tell story
a simple scenario
of how it's used
including setup

list actions
by user or system
key steps, not UI

specify state
what's remembered
enough for actions

concept RestaurantReservation

purpose reducing wait time for tables

principle the restaurant makes slots available at various times; a diner reserves for a particular time, and then can be assured of being seated at that time

state
a set of slots each with
 the start time (includes date)
a set of reservations each with
 the user who made it
 the slot being reserved
 whether seated

actions
createSlot (t: Time)
 ensures creates a fresh slot & associates with time t
reserve (u: User, t: Time): Reservation
 requires some slot at time t not yet reserved
 ensures creates & returns a fresh reservation
  associates it with user u and the slot
seat (r: Reservation)
 requires r is a reservation for about now
 ensures mark r as seated

heuristics
for states & actions

do you have enough actions?

is purpose/value delivered?
note that being in the state may be enough

have you covered the whole life cycle?
is there an initial setup? a winding down?

are there ways to undo previous actions?
or to compensate if erroneous?

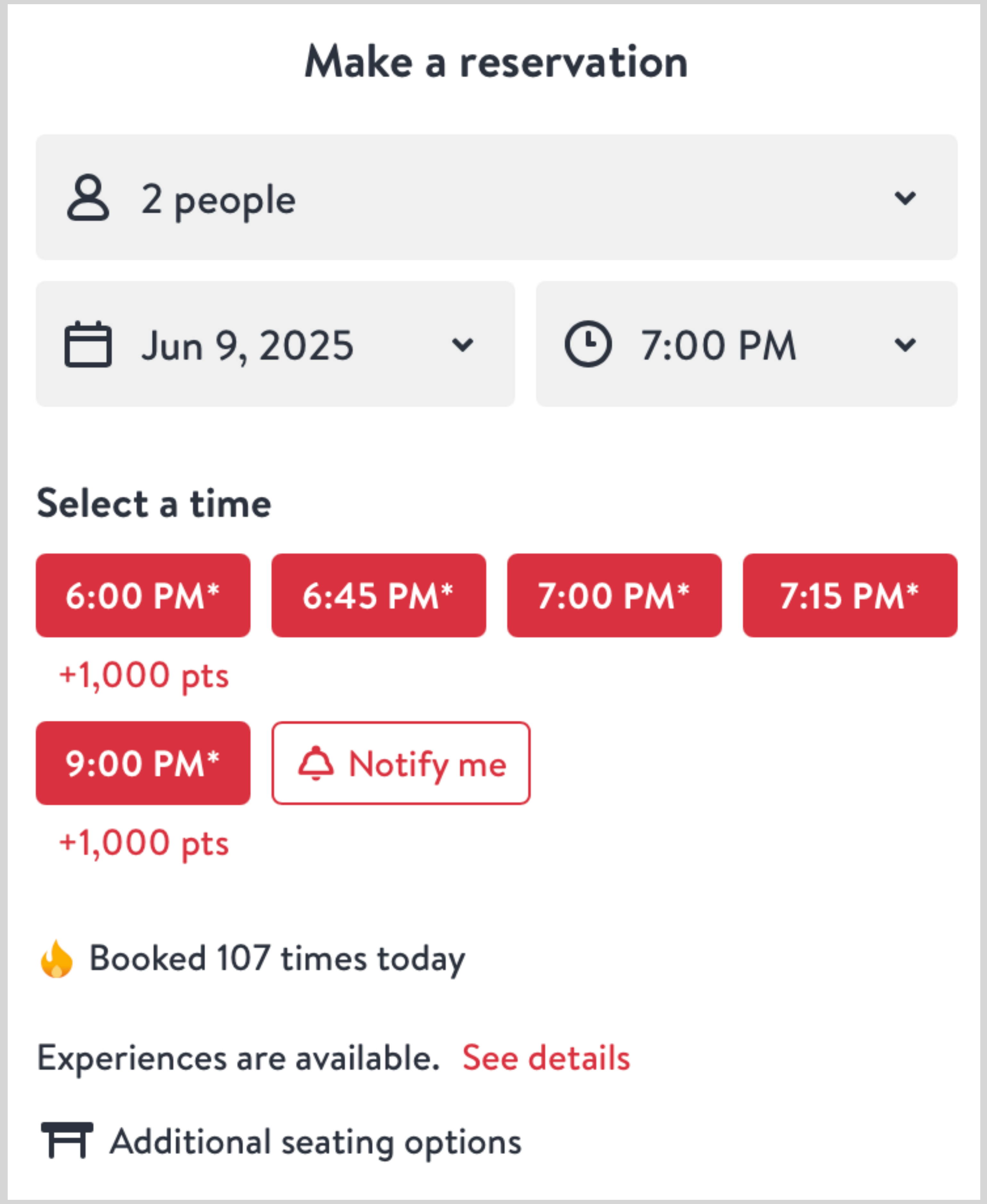
do all nouns have create, update, delete?
for associated state?

seat action?

create slots?

unseat?
cancel reservation?

change reservation?



Make a reservation

2 people

Jun 9, 2025

7:00 PM

Select a time

6:00 PM* +1,000 pts

6:45 PM*

7:00 PM*

7:15 PM*

9:00 PM* +1,000 pts

Notify me

Booked 107 times today

Experiences are available. See details

Additional seating options

concept Reservation
actions reserve...

do you have a rich enough state?

can you support all your actions?
determine if allowed, and generate results

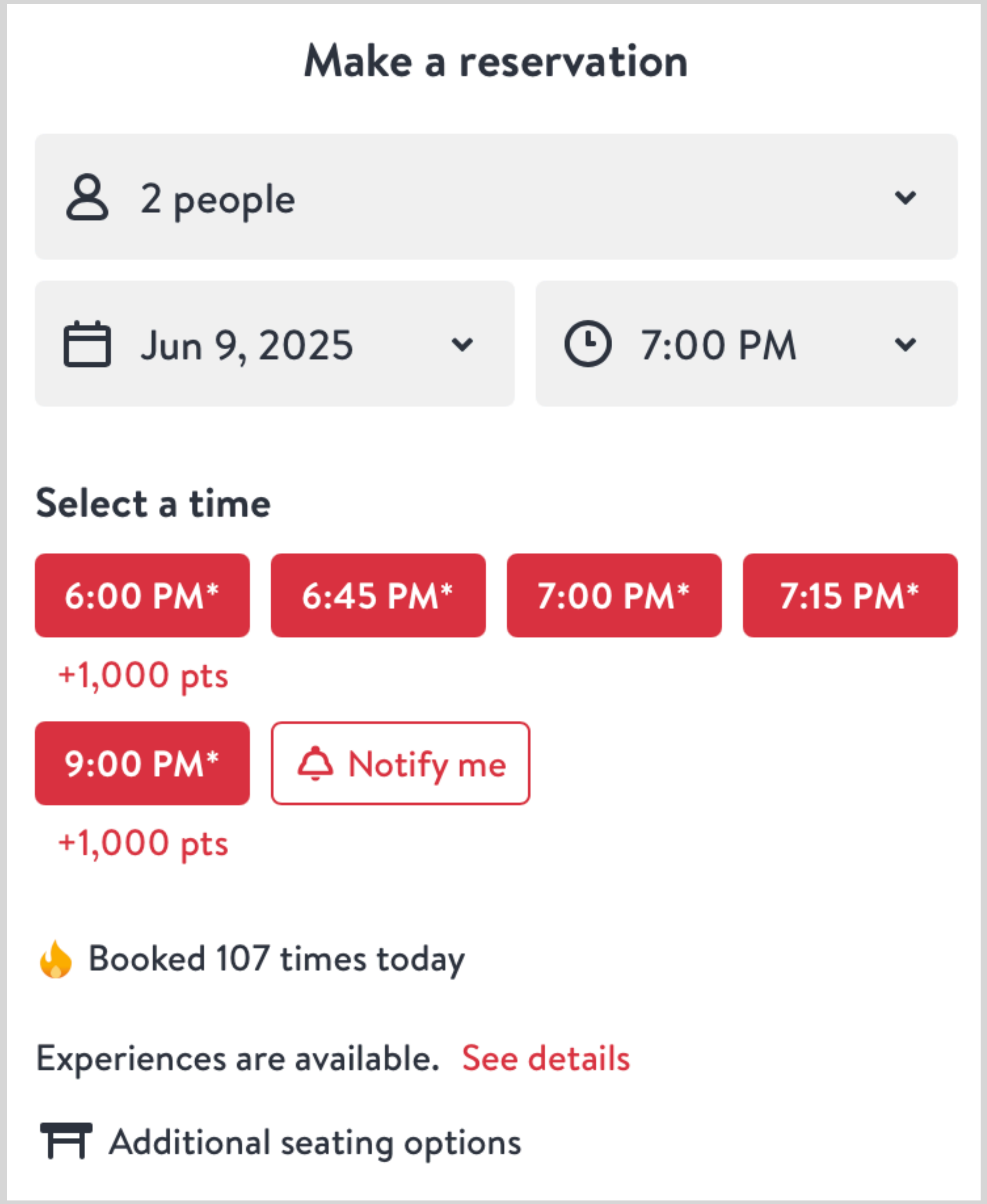
should you track history?
remember completions, deletions, undos?

what info about action occurrence?
maybe also who did it? when?

table sizes?

retain after seat?

by vs. for?
time of reservation?



Make a reservation

2 people

Jun 9, 2025

7:00 PM

Select a time

Booked 107 times today

Experiences are available. See details

Additional seating options

concept Reservation
actions createSlot, reserve, cancel, seat, unseat, no-show, ...

check your understanding

How are concept actions and user interface interactions related? (pick one)

(a) Every interaction in the UI corresponds to a concept action

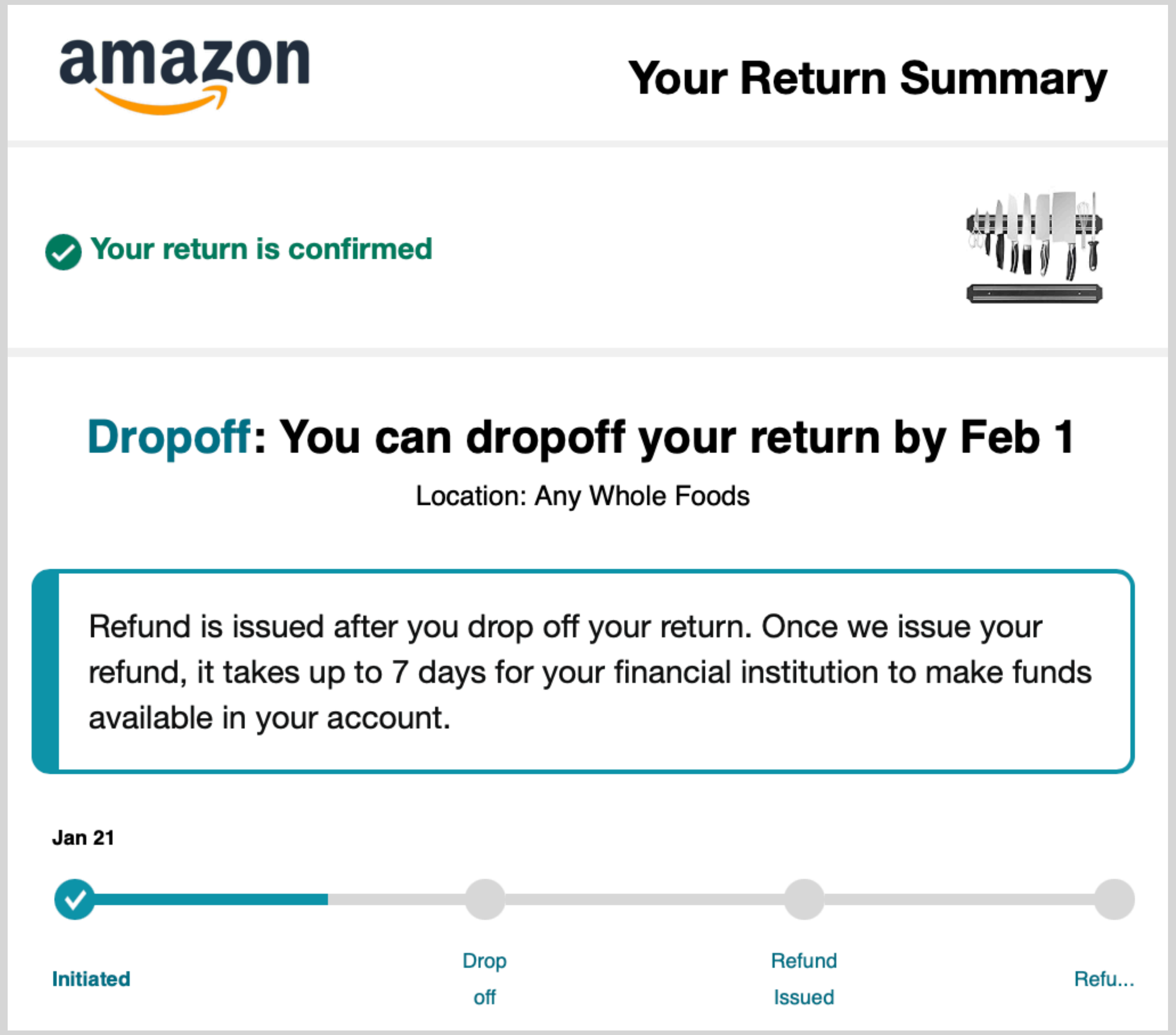
(b) Every concept action must be represented as a button or input in the UI

(c) A concept action can comprise a whole sequence of UI interactions

are concepts modal?

a modal concept: merchandise return

$$\begin{tikzcd}
\text{item received} \arrow[d, "\text{requestReturn}"] \\
\text{request sent} \arrow[d, "\text{receiveRMA}"] \\
\text{RMA received} \arrow[d, "\text{returnItem}"] \\
\text{item returned} \arrow[d, "\text{credit}"] \\
\text{credit granted}
\end{tikzcd}$$



amazon **Your Return Summary**

Your return is confirmed

Dropoff: You can dropoff your return by Feb 1

Location: Any Whole Foods

Refund is issued after you drop off your return. Once we issue your refund, it takes up to 7 days for your financial institution to make funds available in your account.

Amazon shows you the steps

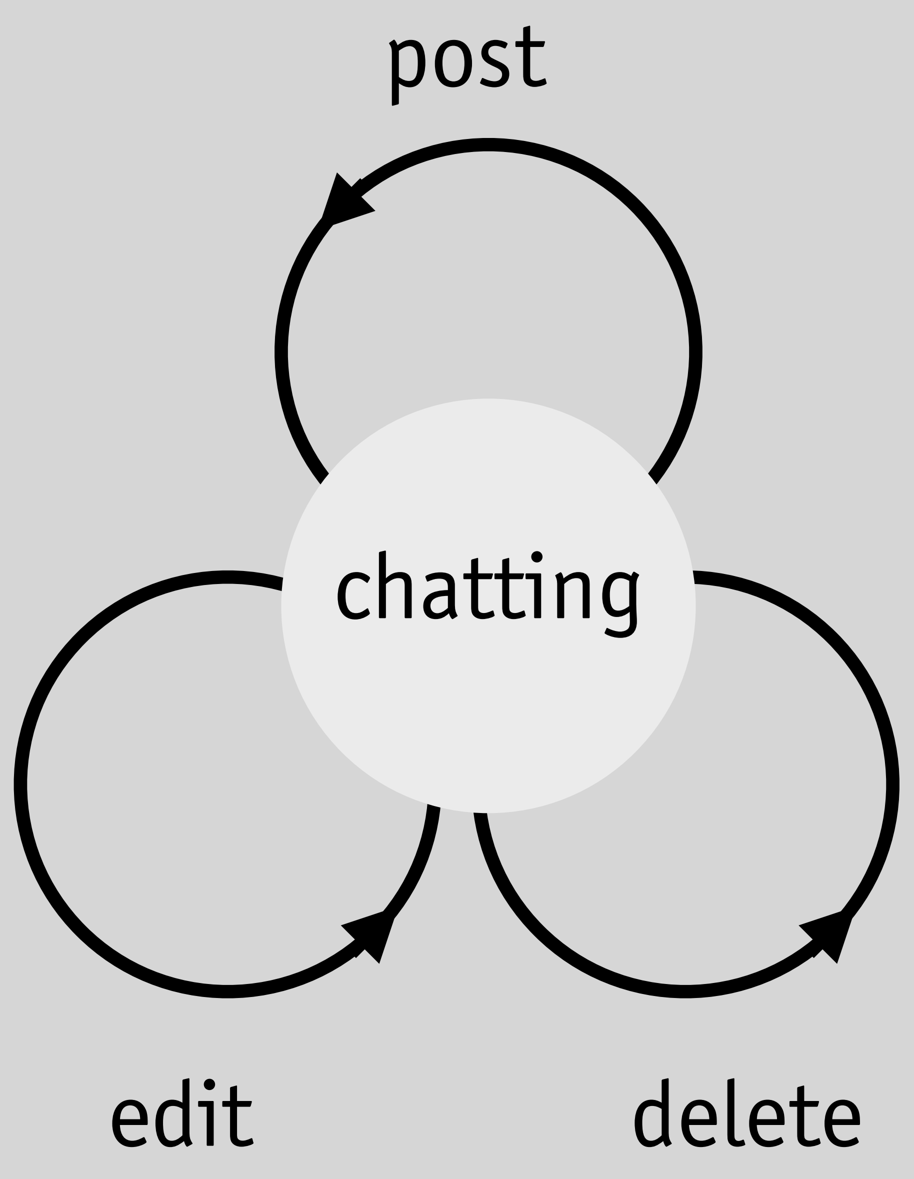
very **constrained** order of actions

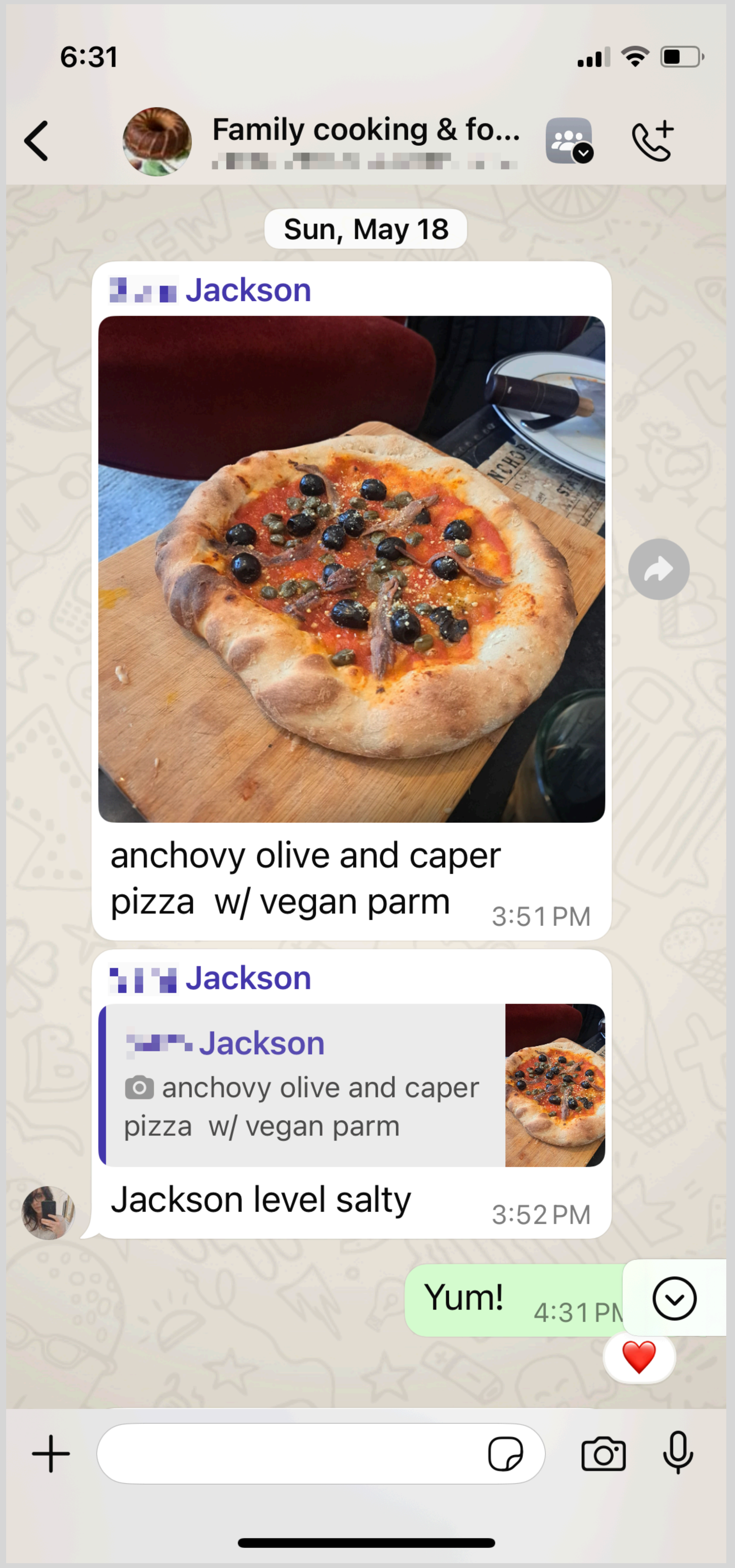
few **deviations** (eg, for canceling)

user knows what **mode** they're in

target of action often **implicit**

a “noun and verbs” concept: social media chat





Jackson
anchovy olive and caper pizza w/ vegan parm 3:51 PM

Jackson
Jackson
anchovy olive and caper pizza w/ vegan parm
Jackson level salty 3:52 PM

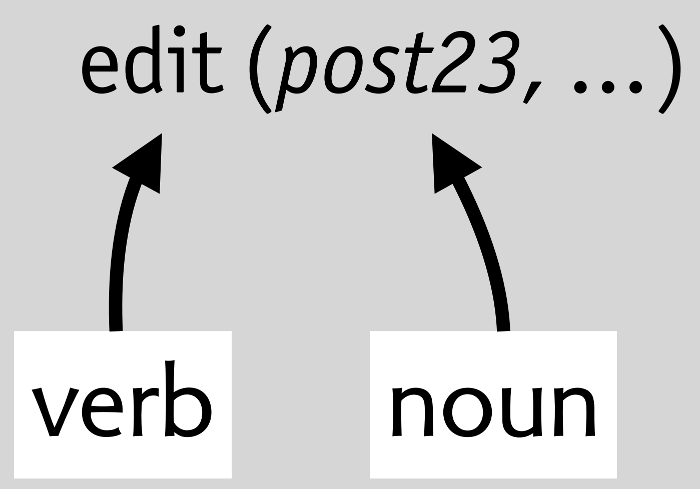
Yum! 4:31 PM

very **free** order of actions

options at every step

user thinks of **things**, not **modes**

target of action **explicit**

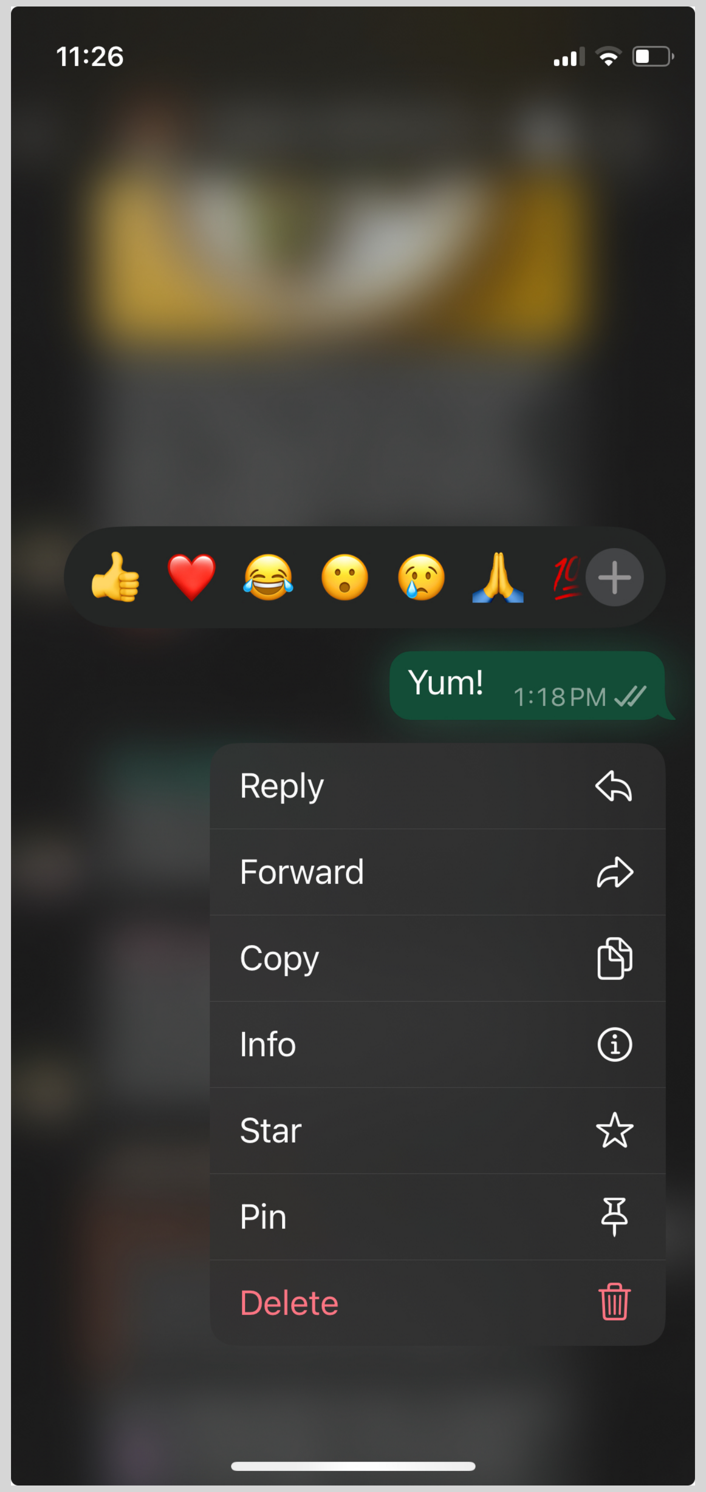


but are they really so different?

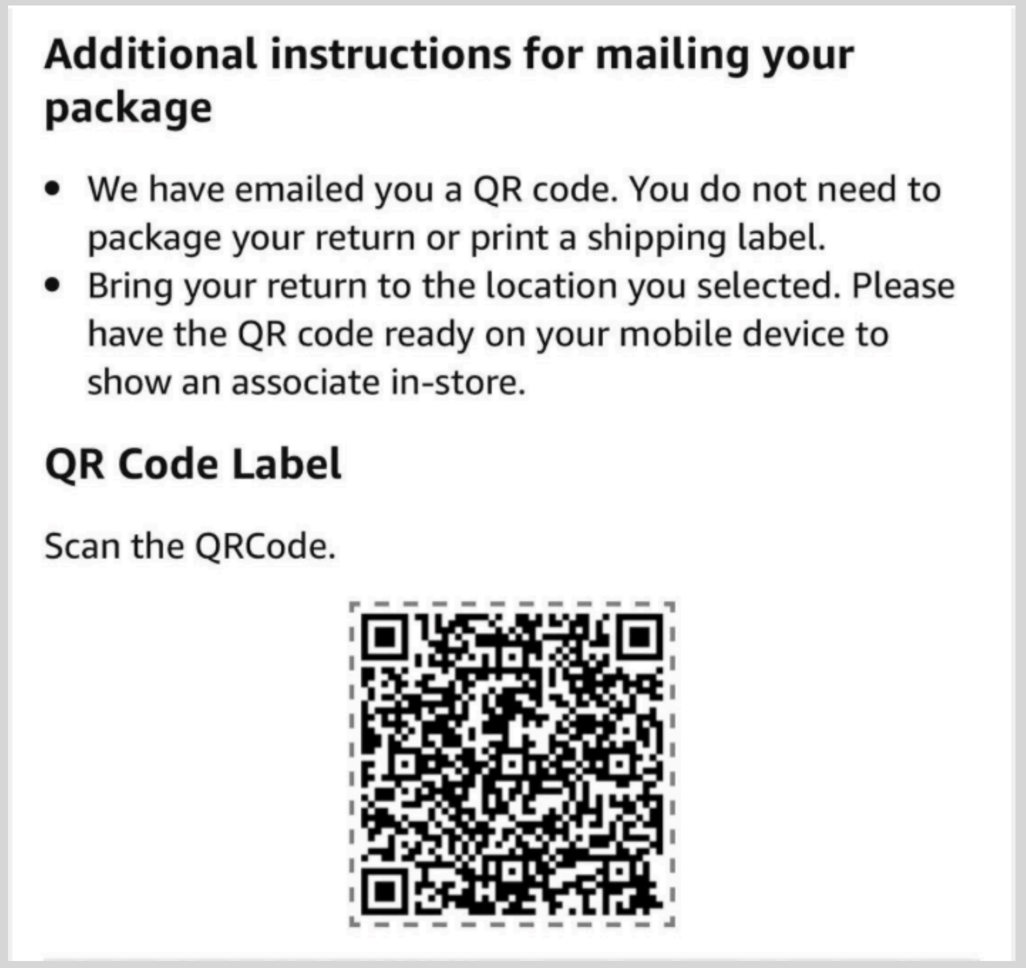
$$\begin{tikzcd}
\text{item received} \arrow[d, "\text{requestReturn}"] \\
\text{request sent} \arrow[d, "\text{receiveRMA}"] \\
\text{RMA received} \arrow[d, "\text{returnItem}"] \\
\text{item returned} \arrow[d, "\text{credit}"] \\
\text{credit granted}
\end{tikzcd}$$

$$\begin{tikzcd}
{} \arrow[d, bend right, "\text{post}"'] \\
\text{post exists} \arrow[u, bend right, "\text{delete}"'] \arrow[d, "\text{edit}"] \\
\text{post edited}
\end{tikzcd}$$

social media post has a lifecycle with modes too


Yum! 1:18 PM

edit (*post23, …*)


Additional instructions for mailing your package

- We have emailed you a QR code. You do not need to package your return or print a shipping label.
- Bring your return to the location you selected. Please have the QR code ready on your mobile device to show an associate in-store.

QR Code Label

Scan the QRCode.

returnItem (*item23*)

which item is returned is in the QR code

in modal interactions target may be present in the **context**

takeaways

arguments of actions
are "nouns" and context

strength of preconditions
determines how modal

```
cancelReservation (r: Reservation)
requires
  r is a reservation
ensures
  removes reservation r
```

concept design encourages
less modal interactions
because concepts run in parallel
& are unconstrained until sync'd

traces

action histories

a password session concept

concept PasswordSession

purpose authenticate users for extended period

principle after a user registers with a name and password, they can login with that same name and password (and if they enter the wrong password, they can't login)

actions
register (n, p: String)
login (n, p: String): Session
logout (s: Session)

traces: histories of actions

actions

register (n, p: String)
login (n, p: String): Session
logout (s: Session)

traces

<>
<**register** (“Alvaro”, “secret”)>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “secret”): s0>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “secret”): s0, **logout** (s0)>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “secret”): s0, **logout** (s0), **login** (“Alvaro”, “secret”): s1>
...

non traces

<**login** (“Alvaro”, “secret”): s0>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “foo”): s0>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “foo”): s0, **logout** (s1)>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “secret”): s0, **login** (“Alvaro”, “secret”): s0>

can we define the traces without using states?

some legal traces

<>
<**register** (“Alvaro”, “secret”)>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “secret”): s0>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “secret”): s0, **logout** (s0)>
<**register** (“Alvaro”, “secret”)>, **login** (“Alvaro”, “secret”): s0, **logout** (s0), **login** (“Alvaro”, “secret”): s1>
...

sample trace rules

when is a **register** action allowed?	allow **register** (n, p) if no prior **register** (n, ...)
when is a **login** action allowed?	allow **login** (n, p): s0 if prior **register** (n, p) ... and no prior **login** (...): s0 without intervening **logout** (s0) ...

this gets very complicated very quickly!

action-state specs: a simpler way to define traces

instead of trace rules:

when is a **login** action allowed?

allow **login** (n, p): s0 if prior **register** (n, p)
  ... and no prior **login** (...): s0 without intervening **logout** (s0) ...

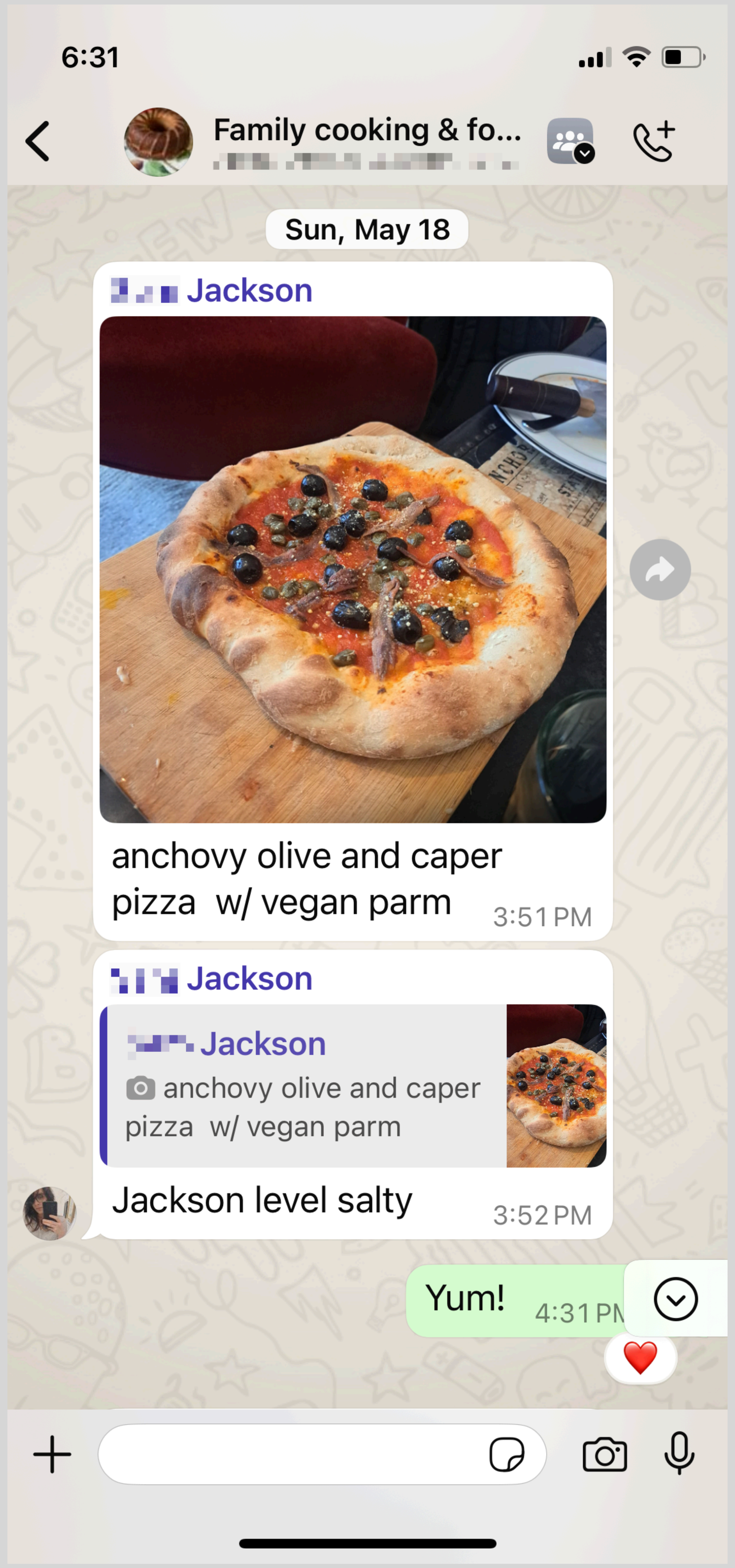
define actions over states:

concept PasswordSession

```
state
a set of registered users each with
  a username and a password
a set of active sessions each with
  an associated user
```

```
actions
 login (n, p: String): Session
   requires some registered user u with name n and password p
   ensures returns some session s not currently active
     and sets user of session s to be u
```

states aren't just an artifact



6:31

Family cooking & fo...
[redacted]

Sun, May 18

[redacted] Jackson

anchovy olive and caper pizza w/ vegan parm 3:51 PM

[redacted] Jackson

[redacted] Jackson
anchovy olive and caper pizza w/ vegan parm

Jackson level salty 3:52 PM

Yum! 4:31 PM

concept GroupChat

state
a set of chats each with
 a set of messages
for each message
 the user who sent it
 the date/time sent
 the content of the message

in concept design, we assume the state is visible
so can query the concept for all messages in chat c sorted by date/time

in approaches that require invisible states (eg, OOP)
you can define "observer actions"

getMessagesForChat (c: Chat): seq Message
requires
 c is a chat
ensures
 returns messages in c in date/time order

many of these!
tedious to specify
often artifact of UI

check your understanding

States & actions in concept design … (pick one)

(a) Both describe aspects of what the user experiences

(b) Are not well-suited to noun-&-verb-style interactions

(c) Can be defined independently of each other

state invariants

aka integrity constraints

designing invariants for concepts

concept PasswordSession

state
a set of registered users each with
  a username and a password
a set of active sessions each with
  an associated user

invariants?

at most one user with a given username

what goes wrong if violated?

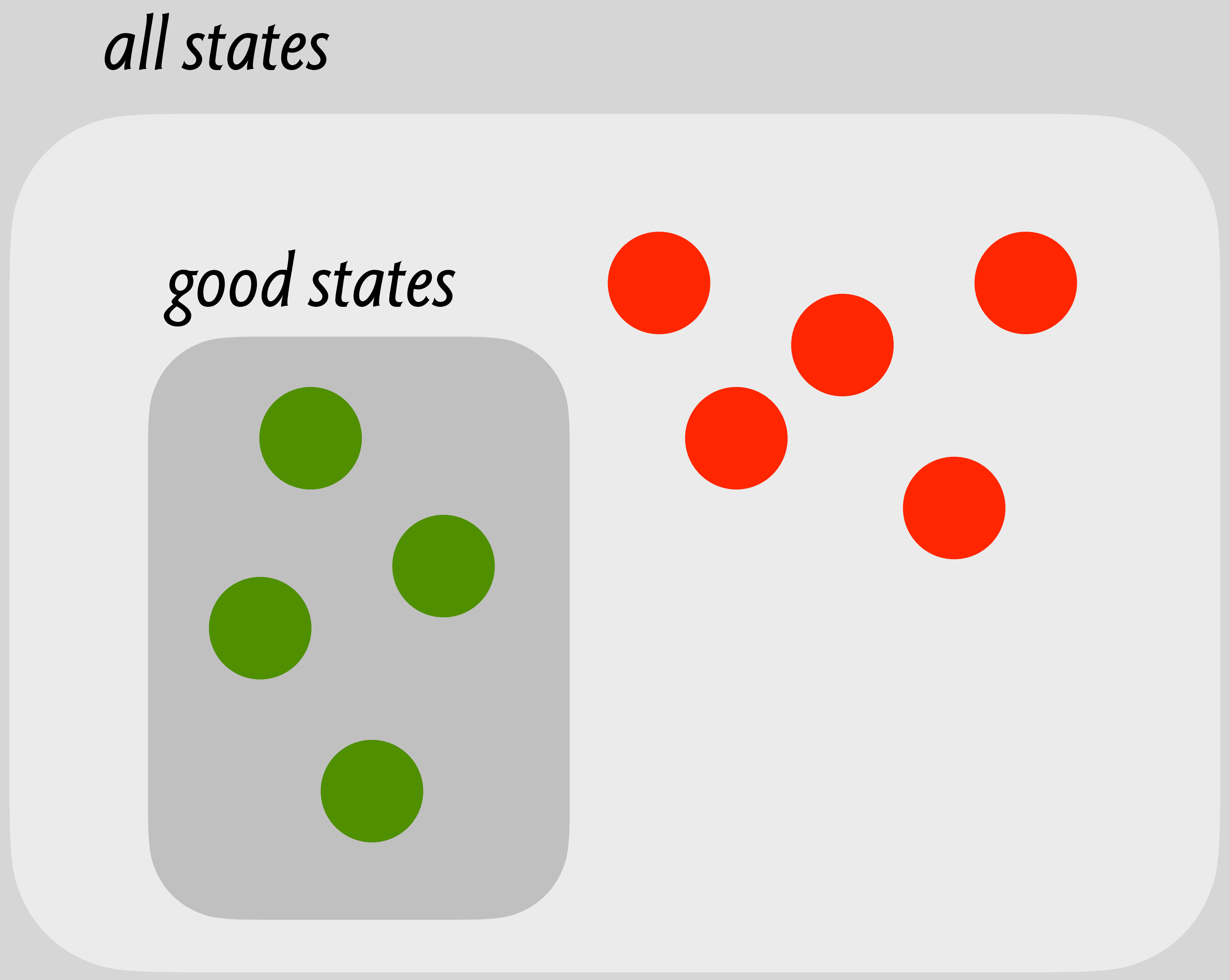
concept RestaurantReservation

state
a set of slots each with
  the start time (includes date)
a set of reservations each with
  the user who made it
  the slot being reserved

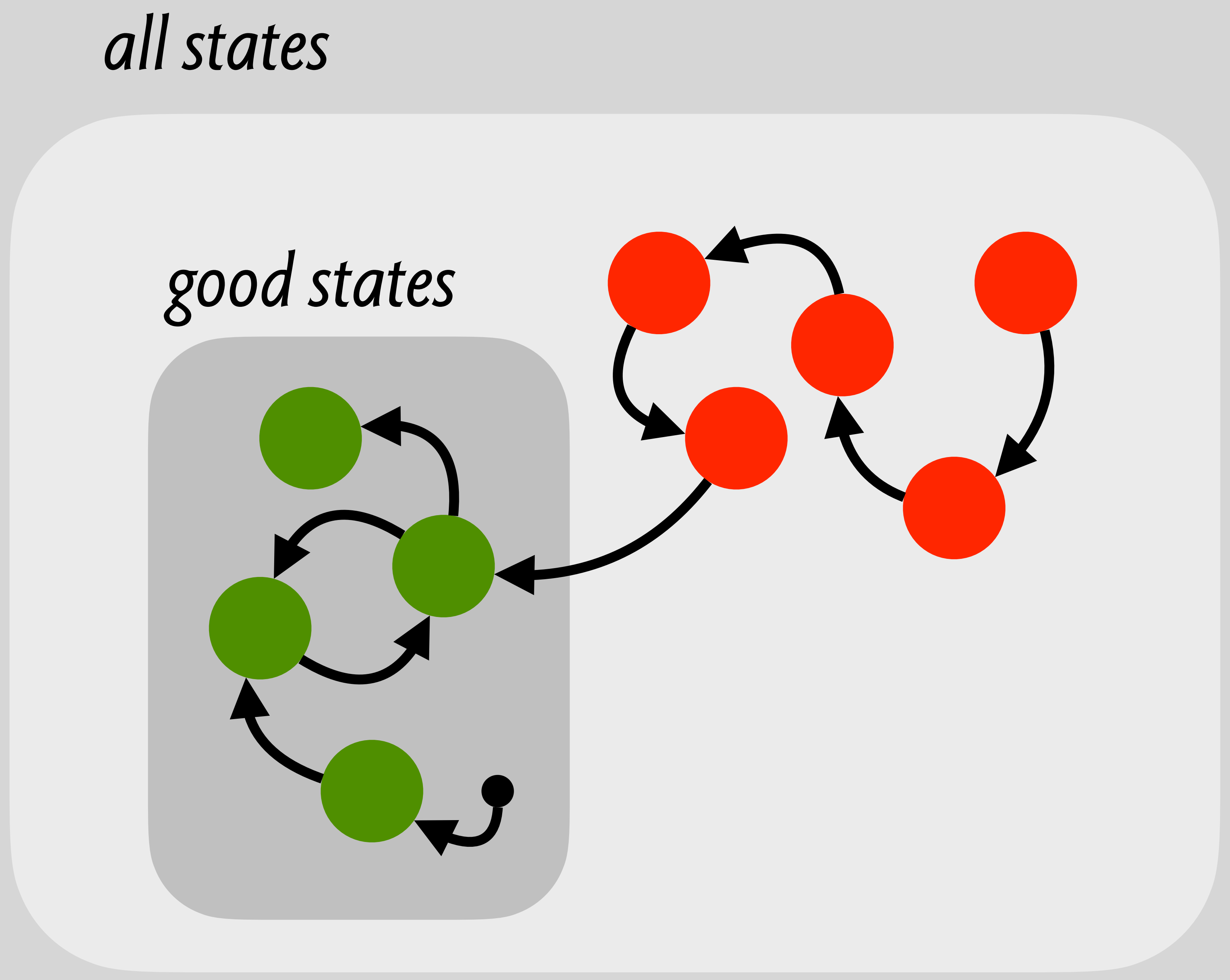
at most one reservation for a given slot

at most one reservation for a given user ?

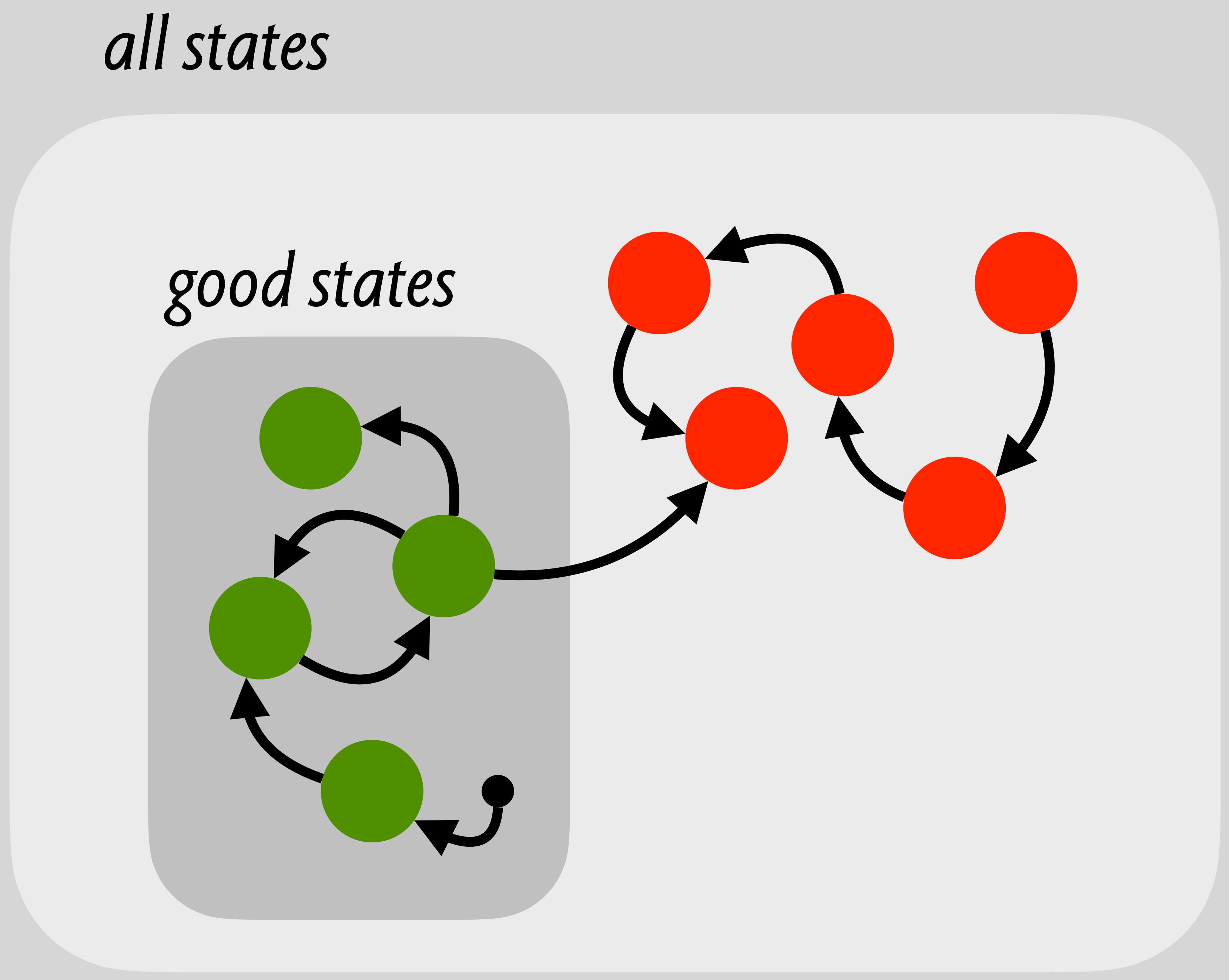
classifying states



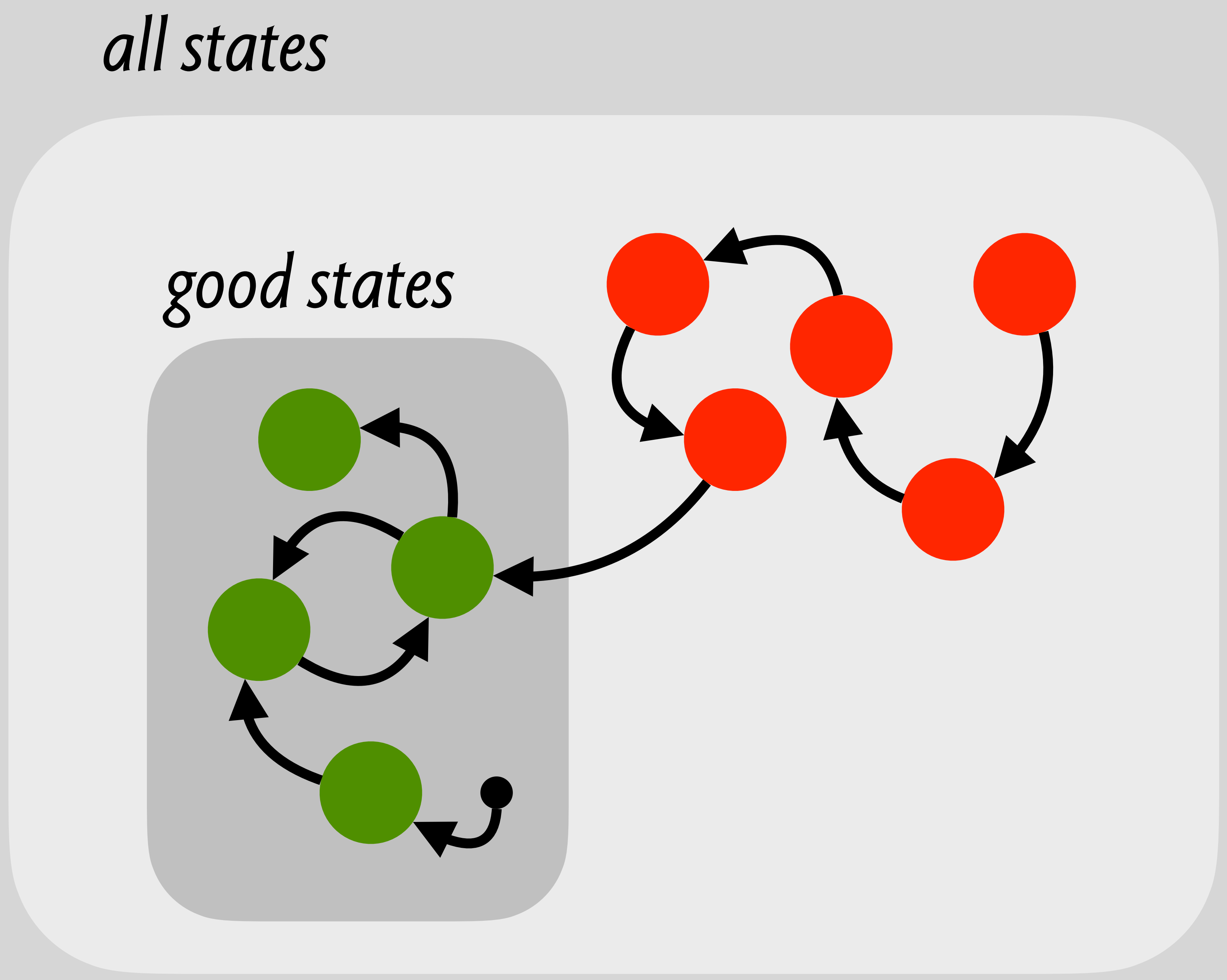
a safe design



an unsafe design



inductive reasoning strategy



what we want to avoid
reasoning about traces
complicated and tedious!

a better approach
reasoning about steps taken by actions
(1) check that the initial state is good
(2) and no action goes from a good to a bad state

applying inductive reasoning to reservation concept

concept RestaurantReservation

state
a set of slots each with
 the start time (includes date)
a set of reservations each with
 the user who made it
 the slot being reserved
 whether seated

actions
 createSlot (t: Time)
  ensures creates a fresh slot & associates with time t
 reserve (u: User, t: Time): Reservation
  requires some slot at time t not yet reserved
  ensures creates & returns a fresh reservation
   associates it with user u and the slot
 seat (r: Reservation)
  requires r is a reservation for about now
  ensures mark r as seated

invariant

at most one reservation for a given slot

check invariant holds in initial state

✓ initially, no reservations

check each action preserves invariant

✓ only the reserve action modifies set of reservations

reserve action's ensures slot is not reserved

states & data models

getting more precise

simplifying the state

concept RestaurantReservation

state
a set of slots each with
 the start time (includes date)
a set of reservations each with
 the user who made it
 the slot being reserved

before, we represented like this

slot	time
s0	July 4, 2025 at 7:00pm

res	user	slot
r0	u1	s0

here's a simpler, more atomized representation

Slot
s0

Reservation
r0

User
u1

these are SETS

time

s0	Ju..

user

r0	u1

slot

r0	s0

these are BINARY RELATIONS

a diagrammatic form

Slot	Reservation	User
s0	r0	u1

these are SETS

time		user		slot	
s0	Ju..	r0	u1	r0	s0

these are BINARY RELATIONS

$$\begin{tikzcd}
 & \text{Reservation} \arrow[dl, "\text{slot}"'] \arrow[dr, "\text{user}"] & \\
\text{Slot} \arrow[d, "\text{time}"'] & & \text{User} \\
\text{DateTime} & &
\end{tikzcd}$$

why kind of set is DateTime?
a set of built-in values

what are the values of Slot, eg?
they're identifiers

about this notation

states can be represented as just sets & binary relations
never need tables with more than two columns

this allows a nice diagrammatic representation
this is the “entity relationship diagram”

there are no objects here
a slot is just an identifier associated with a time etc
not a composite object (but could be implemented as one)

why this model helps
succinct and precise, brings clarity during design
easily translated into code (and database schemas etc)

check your understanding

When a concept has stronger state invariants… (select all that apply)

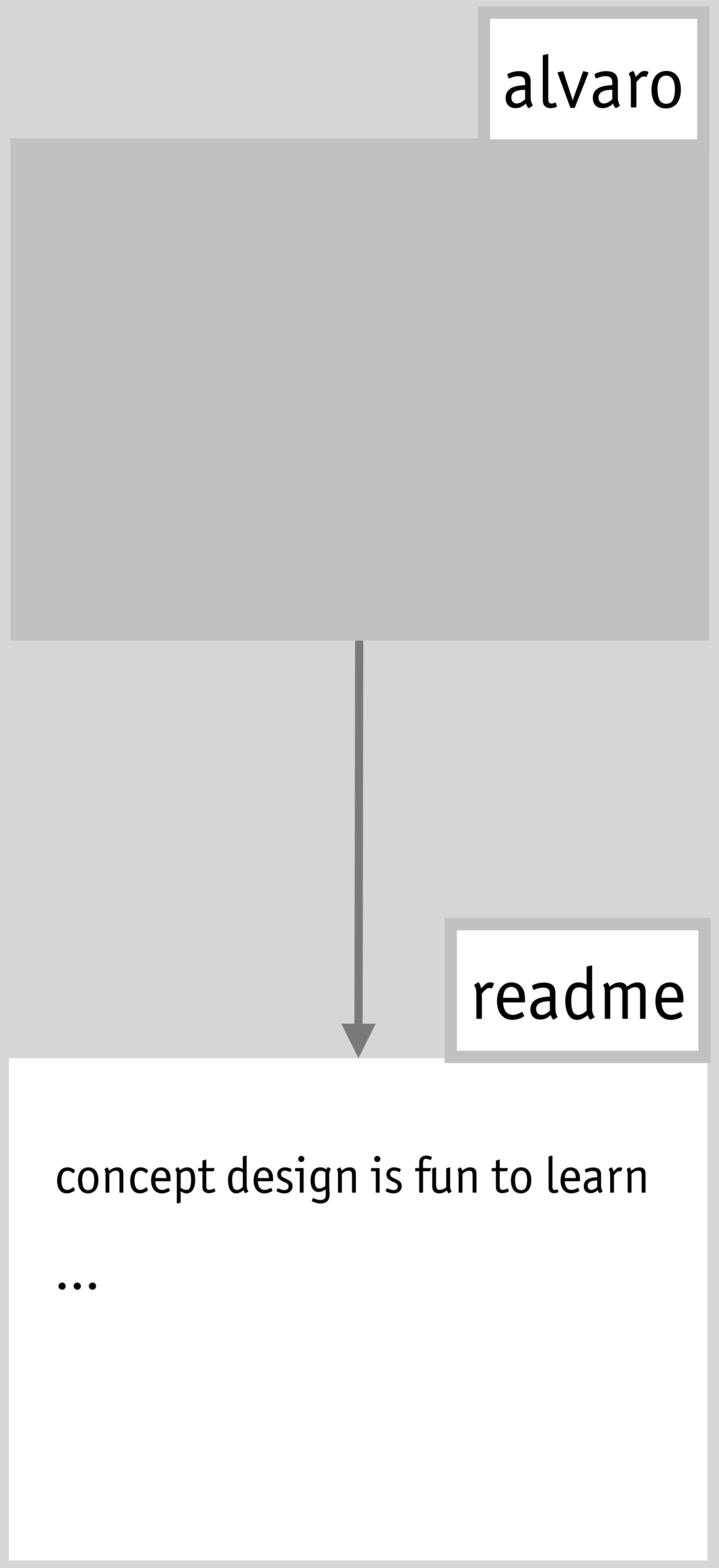
(a) User behavior will generally be more constrained

(b) The concept will be easier to implement

(c) More input validation will generally be needed

two folder concepts

a simple folder concept



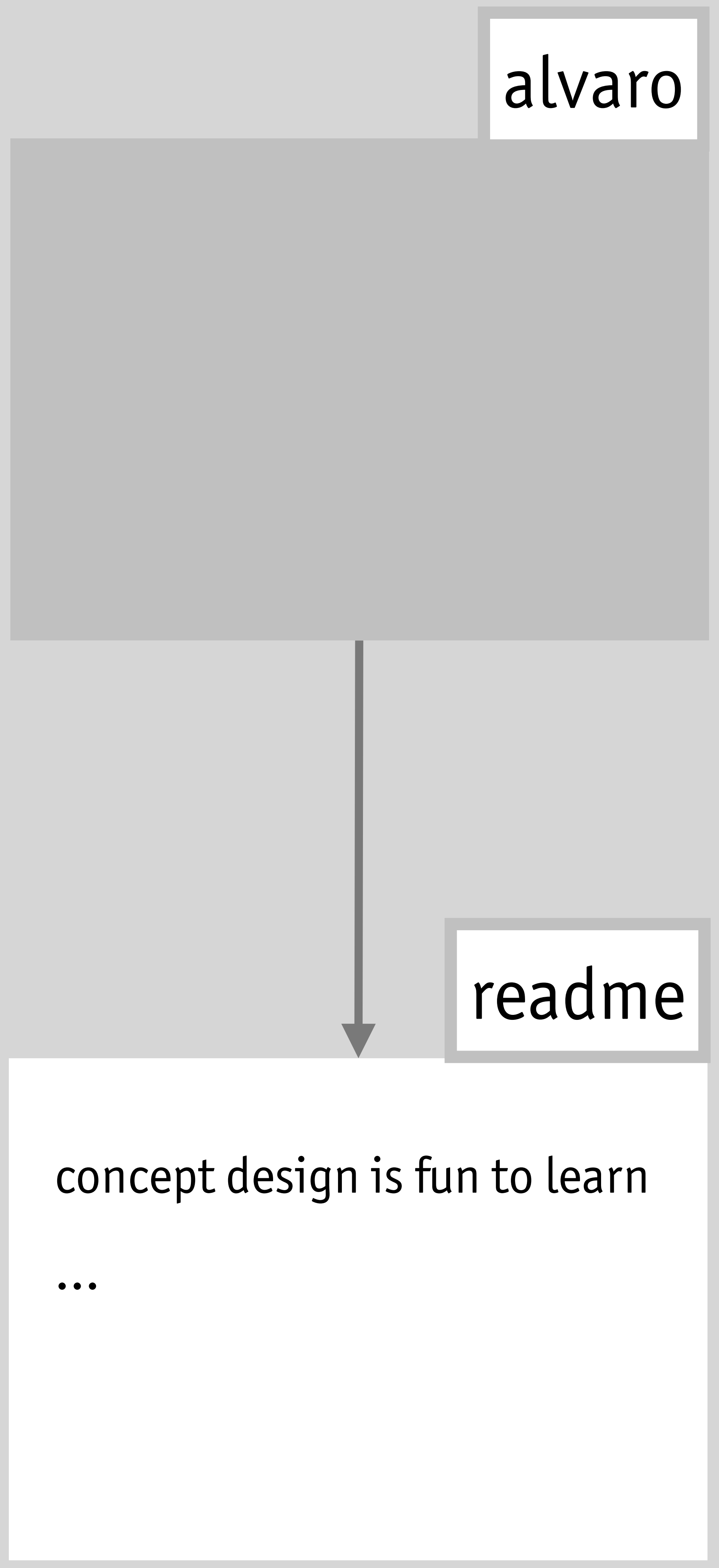
concept SimpleFolder

state
a set of folders each with
  a name
  some contents (files or folders)
a set of files each with
  a name
  some body (text)

$$\begin{tikzcd}
 & \mathit{Item} \arrow[r, "\text{name}"] & \text{String} \\
\text{Folder} \arrow[ur, "\text{contents}"] \arrow[ur, dashed, bend left] & \text{File} \arrow[u, dashed, "\text{is-a}"'] \arrow[d, "\text{body}"] & \\
 & \text{Content} &
\end{tikzcd}$$

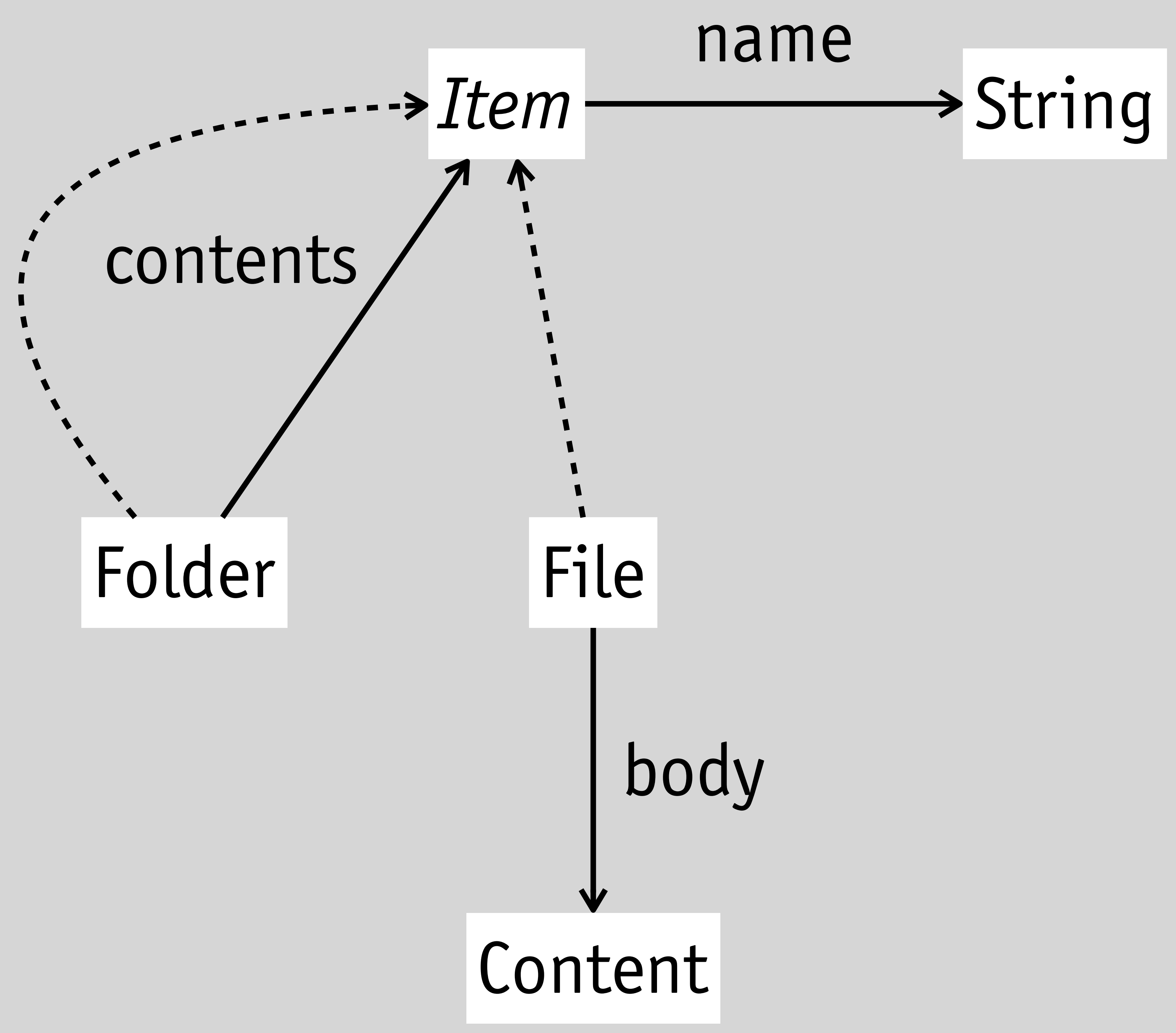
diagram introduces a new trick
an arrow for is-a (aka subset)
allowing sets for generalization

what invariants?



concept SimpleFolder

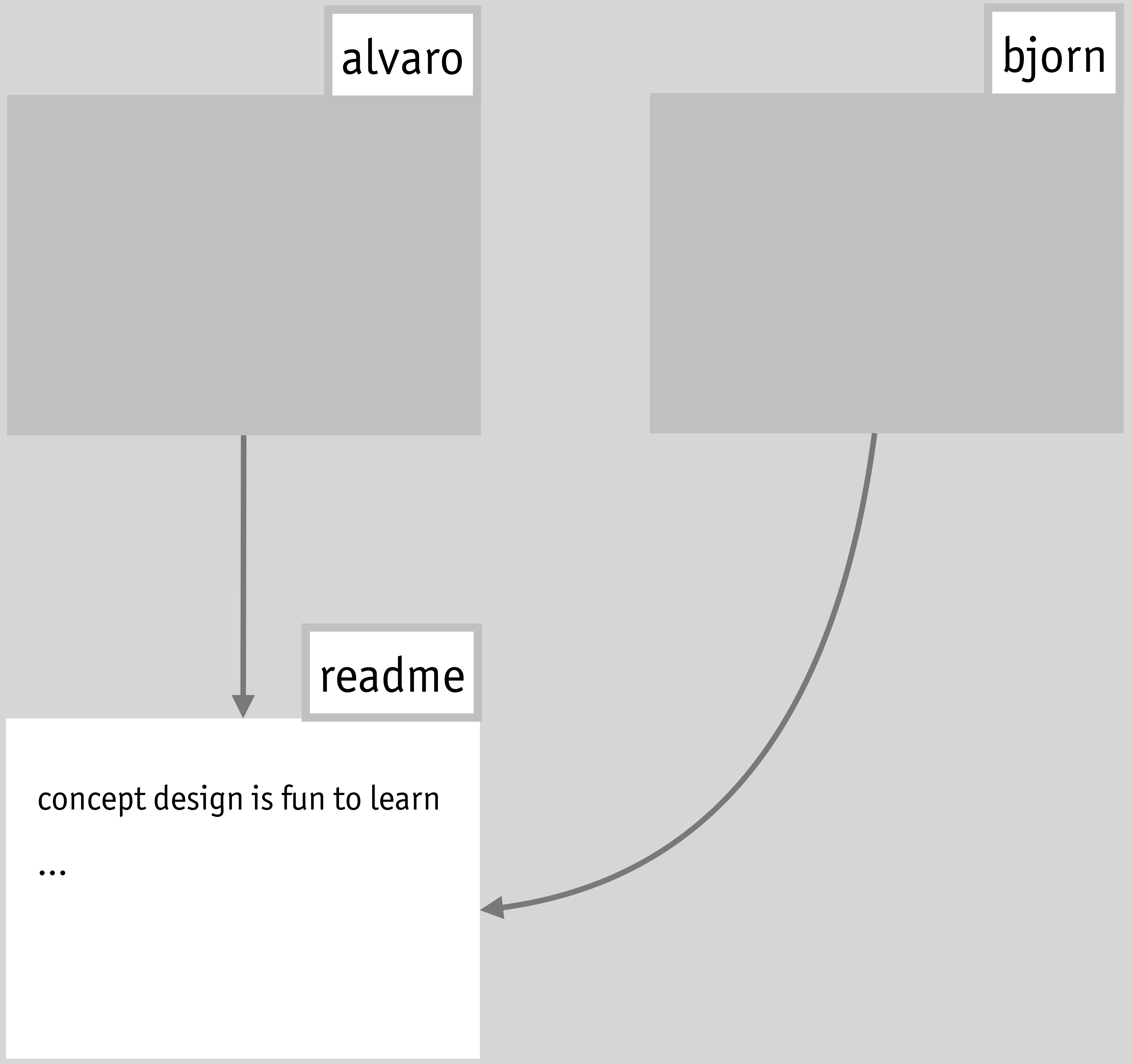
state
a set of folders each with
  a name
  some contents (files or folders)
a set of files each with
  a name
  some body (text)



some invariants

✓ every file belongs to a folder
✓ no folder contains itself (directly or indirectly)
✓ some root folder contains all others (directly or indirectly)
? each file or folder belongs to at most one folder
? no two contents of a folder have the same name

suppose alvaro shares a file with bjorn

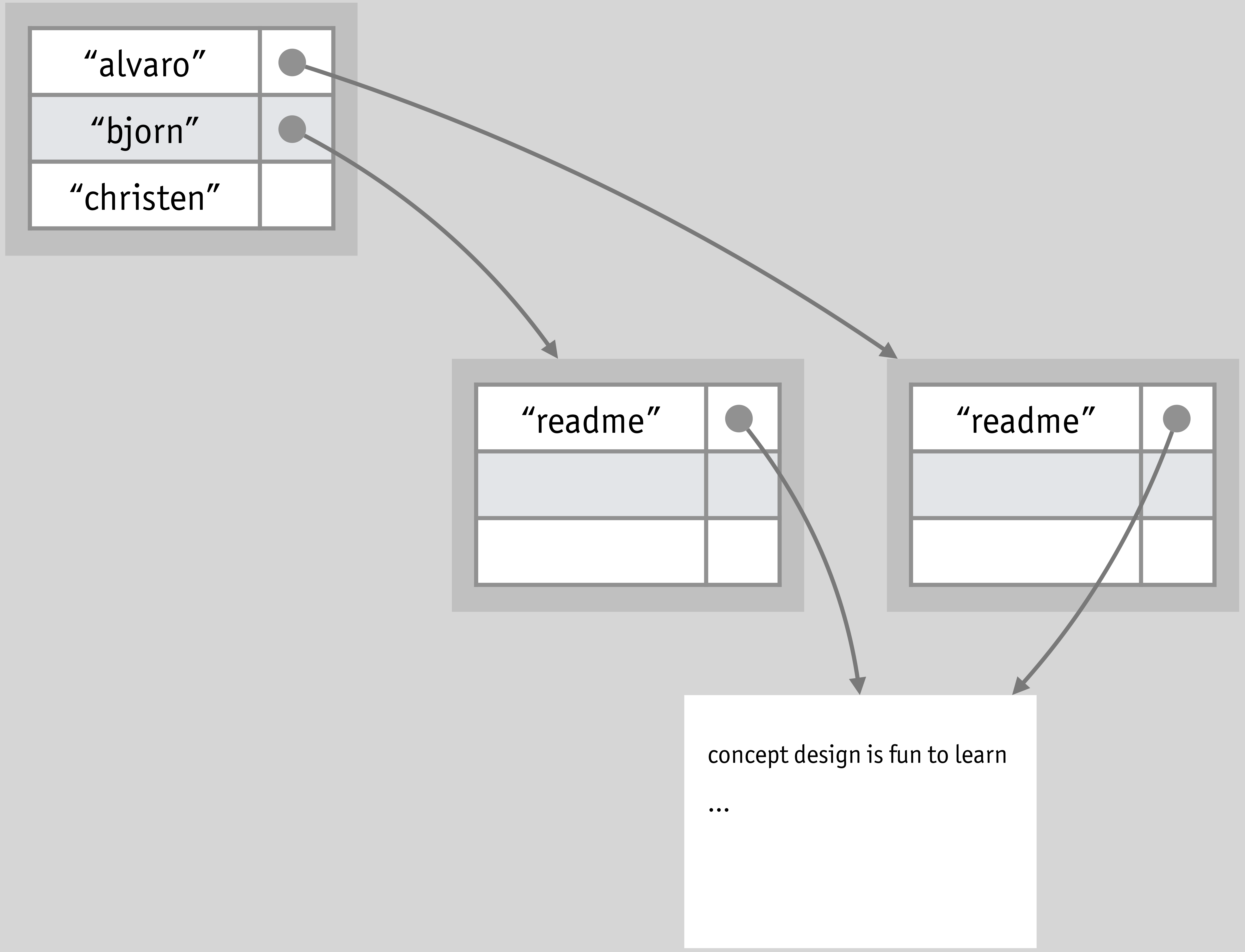


now the file called "readme" belongs to two folders!

if Bjorn can rename the file how to maintain unique names in Alvaro's folder?

a version of Google Drive was exactly like this: filenames not unique in folder

an alternative design: the Unix directory concept



concept design is fun to learn

...

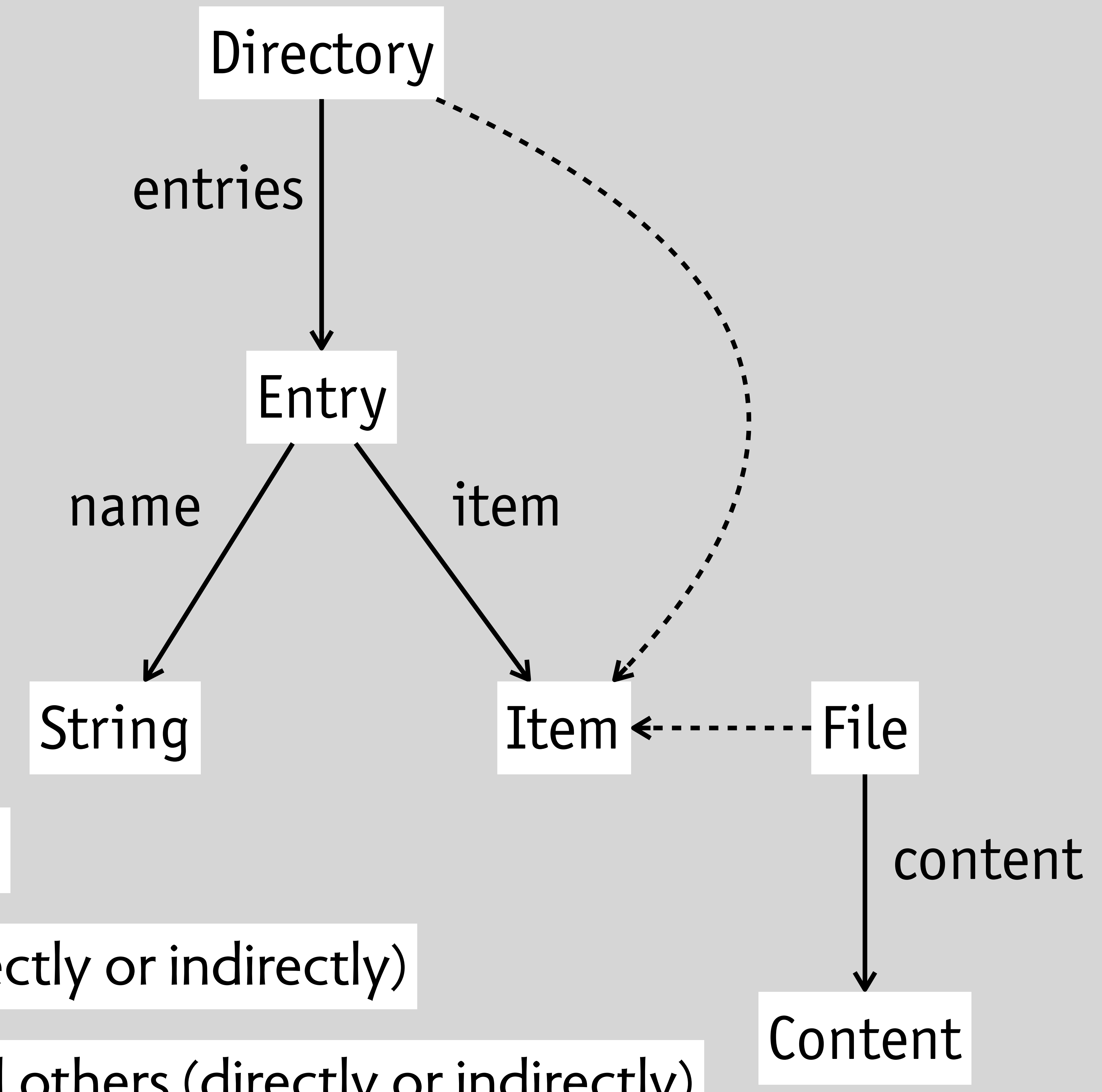
concept UnixDirectory

state
a set of directories each with
  a set of entries
a set of entries each with
  a name
  an item (directory or file)
a set of files each with
  a body (text)

the state of the Unix directory concept

concept UnixDirectory

state
a set of directories each with
  a set of entries
a set of entries each with
  a name
  an item (directory or file)
a set of files each with
  a body (text)



some invariants

✓ every file belongs to a directory

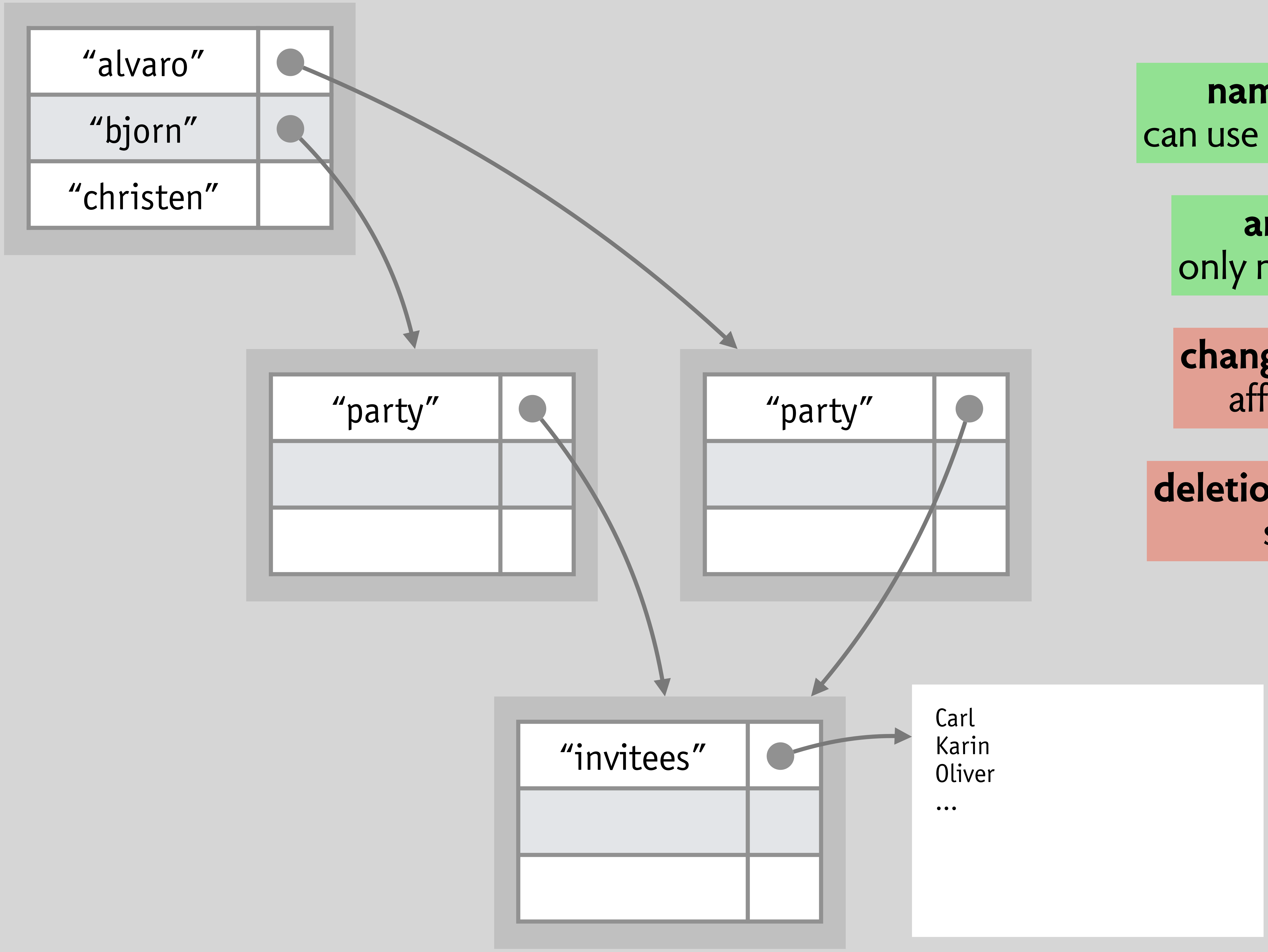
✓ no directory contains itself (directly or indirectly)

✓ some root directory contains all others (directly or indirectly)

✗ each file or directory belongs to at most one directory

✓ no two contents of a directory have the same name

how is this for the user?



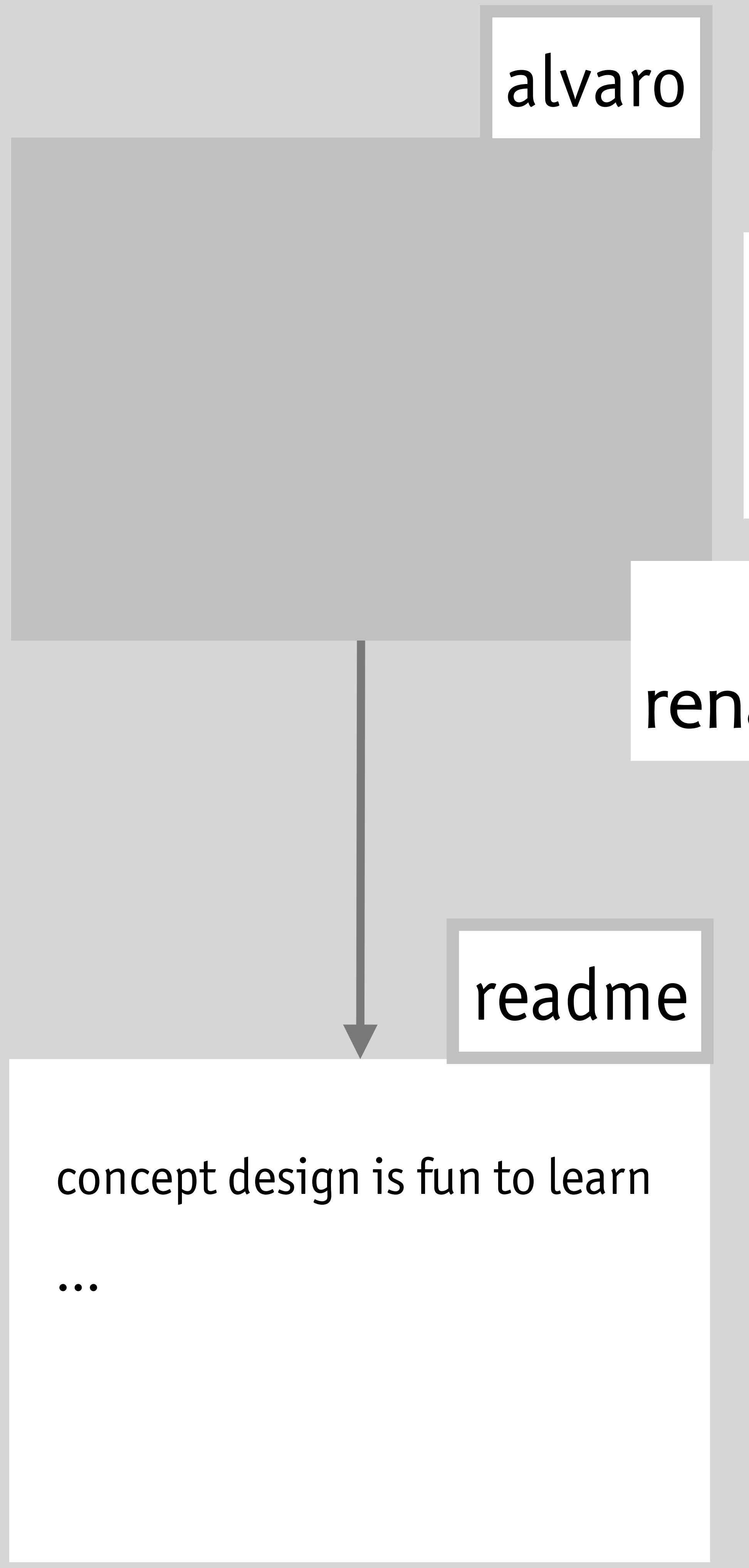
names unique within a directory
can use paths to identify files & directories

any user can change a name
only need to check uniqueness locally

changing name of shared directory
affects owner's name *sometimes*

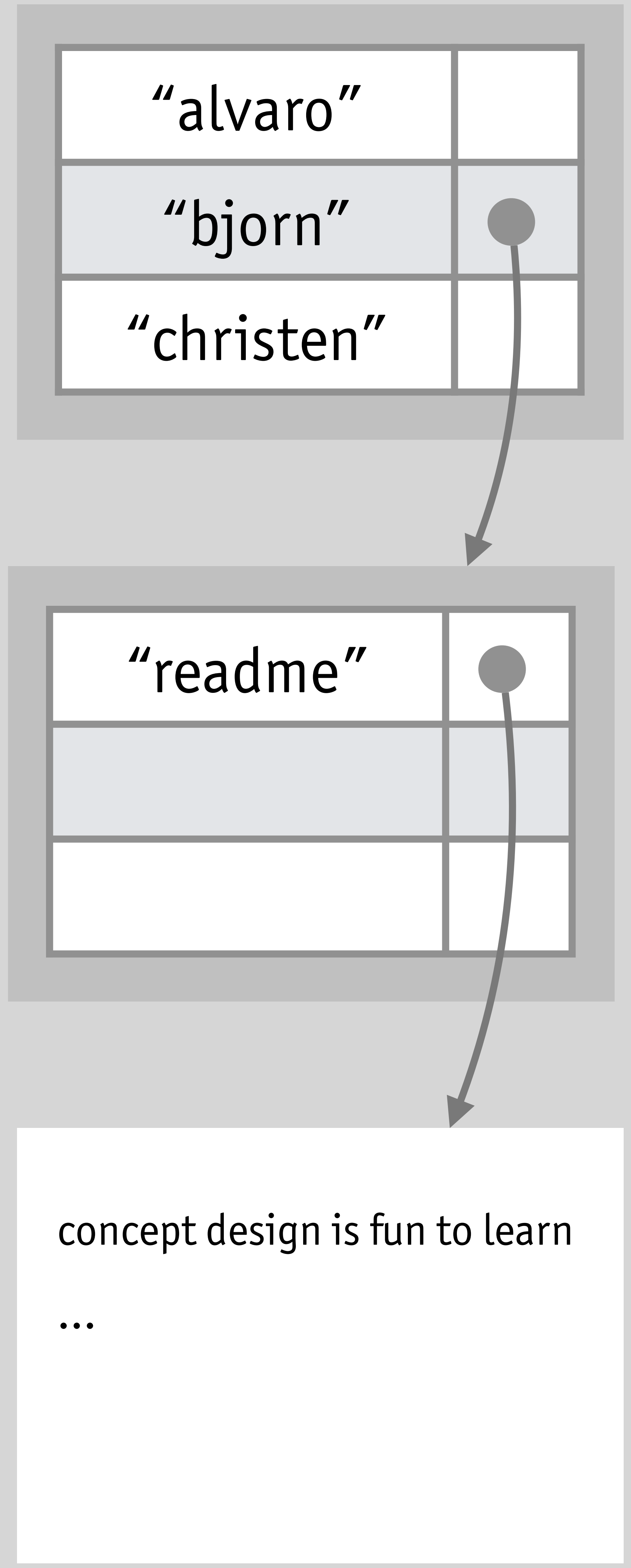
deletion removes an entry not an item
so might still be reachable!

a fine distinction with major impacts



name is property of item
could be factored out
into another concept!

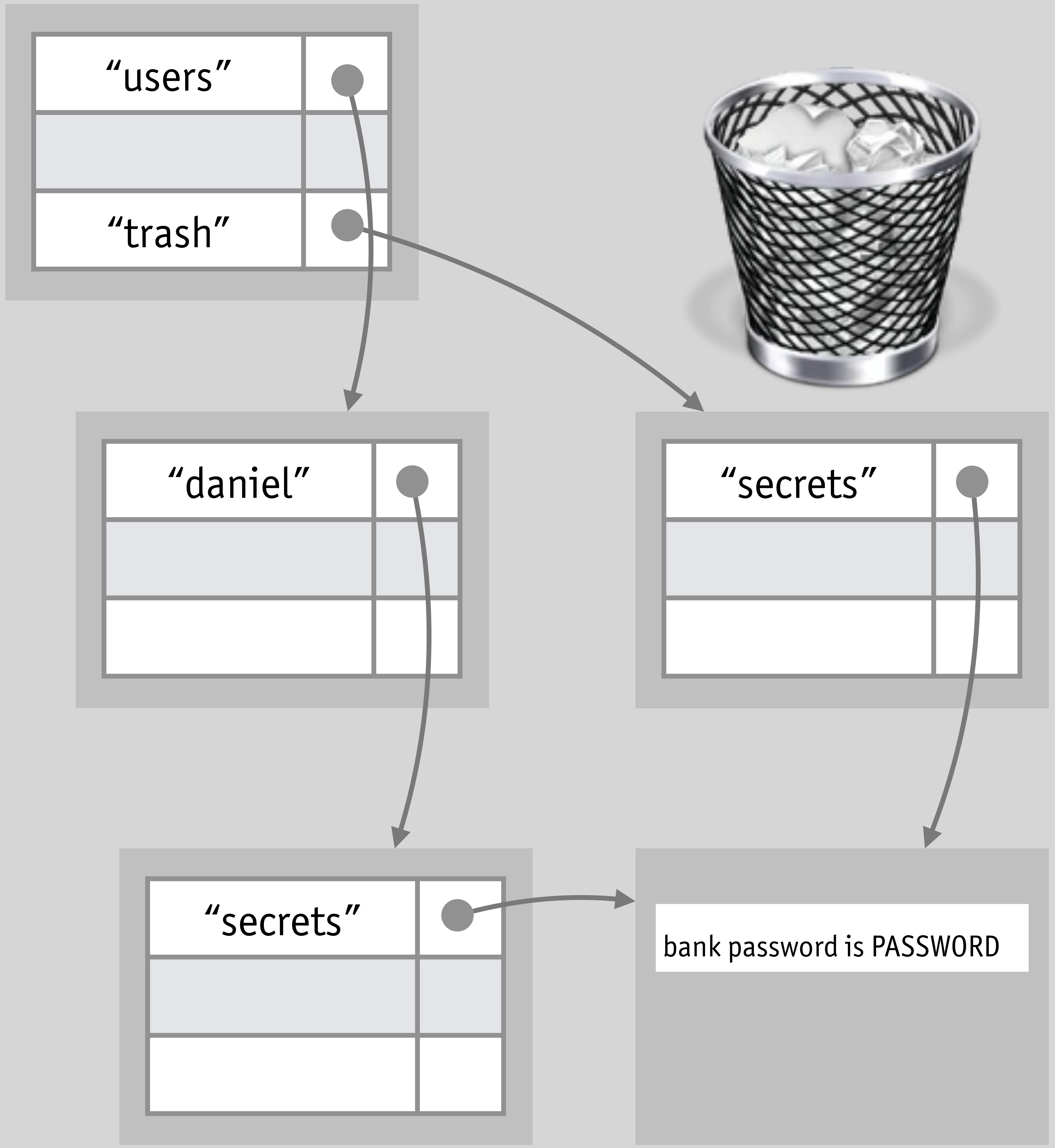
rename acts on item
rename (f: File or Folder, n: String)



name qualifies link
belongs to entry
not to the item itself!

rename acts on directory
rename (f: File or Dir, in: Dir, n: String)

a unix puzzle: what happens when trash is emptied in this case?



takeaways

takeaways

what you learned today

state machines
UI-independent model of behavior
modal vs nouns and verbs
traces as action histories
state invariants & inductive reasoning
formalizing state with data models

how detailing behavior helps
raises tricky design questions
exposes complexities that may confuse users
can suggest opportunities for simplification

what I hope you can now do

think about behavior more clearly
states, actions & traces

design concepts in detail
with states and actions

produce behavior outlines
with data model diagrams & action lists

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 state+action model of a concept, from Autodesk or not
(no need to finish it: just make a start so we can see where it’s going)
or, comment on an Autodesk concept in the sandbox

plan for last session
how to break a system into concepts
modularity, purpose and synchronization