designing concepts

Daniel Jackson · Autodesk Oslo Workshop · August 25-26, 2025

on details


Charles Eames Ray Eames 13721
The details are not details. They make the design. *Charles Eames*
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

Terra - Eataly Boston

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

Overview | Experiences | Popular dishes | Photos

how many steps to enter data?

About this restaurant

Charming | Lively | Good for special occasions

should available slots be red?

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

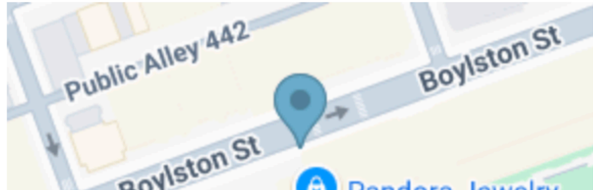
Booked 110 times today

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

is this helpful?

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

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

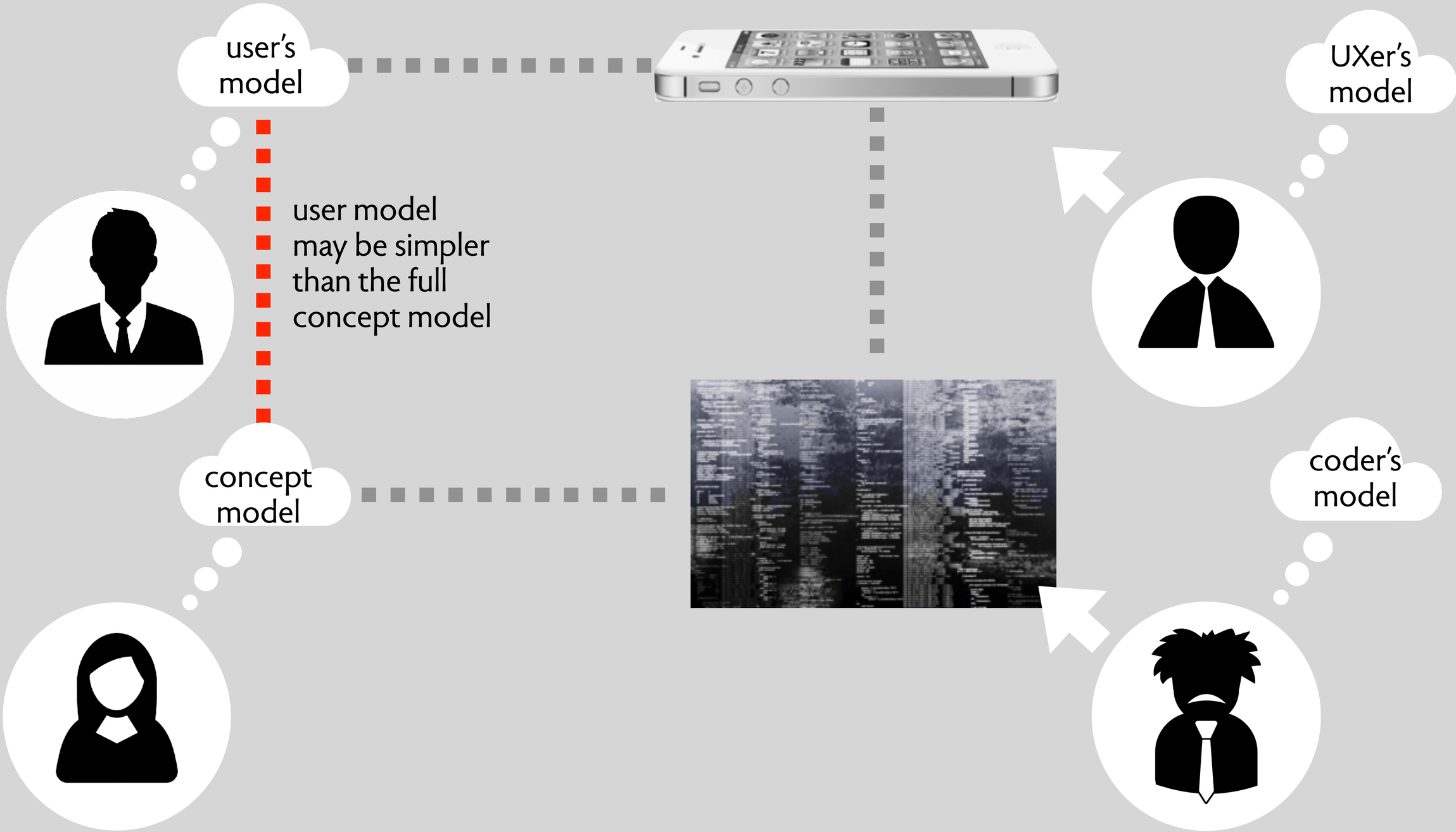
they can be a distraction
color of slots before we've
decided that we have slots?

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

shared understanding
between UX & engineering
capturing the overlap

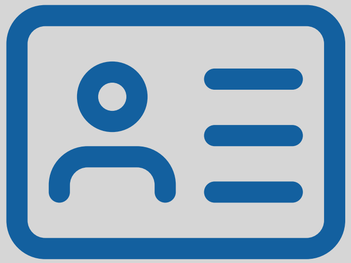
which steps are concept actions?

many models playing different roles



a full example
a reservation concept

how to design a concept

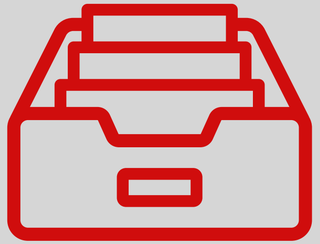


pick a name
specific to function
but for general use

describe purpose
why design or use it?
value to stakeholders

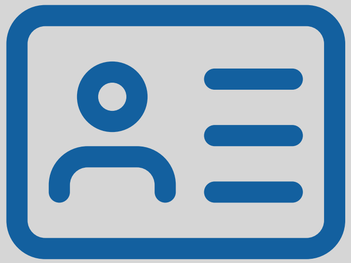
tell story
a simple scenario
of how it's used

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

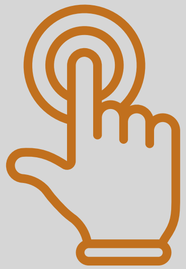
tracking occupancy patterns

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

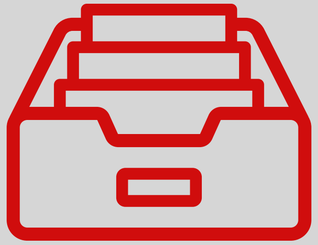
deleteSlot

defining action arguments

createSlot	createSlot (time)
reserve	reserve (user, time): reservation
seat	seat (reservation)

cancel	cancel (reservation)
noShow	noShow (reservation)
deleteSlot	deleteSlot (slot)

devising the state



specify state
what's remembered
enough for actions

a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot

defining the actions

state

a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
  a seated Flag

actions

createSlot (time)
effect
  creates a fresh slot for the time

reserve (user, time): reservation
requires
  some slot at this time not yet reserved
effect
  creates & returns a fresh reservation
  associates it with user and the slot

"precondition"
what's true of state before

"postcondition"
relates state after to before

seat (reservation)
requires
  reservation is for about now
effect
  marks reservation as seated

explaining state notation

state

a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
  a seated Flag

this means each slot has a property called *time* that is a Time

this means each reservation has a property called *seated* that is a Flag

state

a set of Slots with
- a Time

a set of Reservations with
- a User
- a Slot
- a seated Flag

actions

createSlot (time)
effect creates a fresh slot for the time

reserve (user, time): reservation
requires some slot at this time not yet reserved
effect creates & returns a fresh reservation
associates it with user and slot

seat (reservation)
requires reservation is for about now
effect marks reservation 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

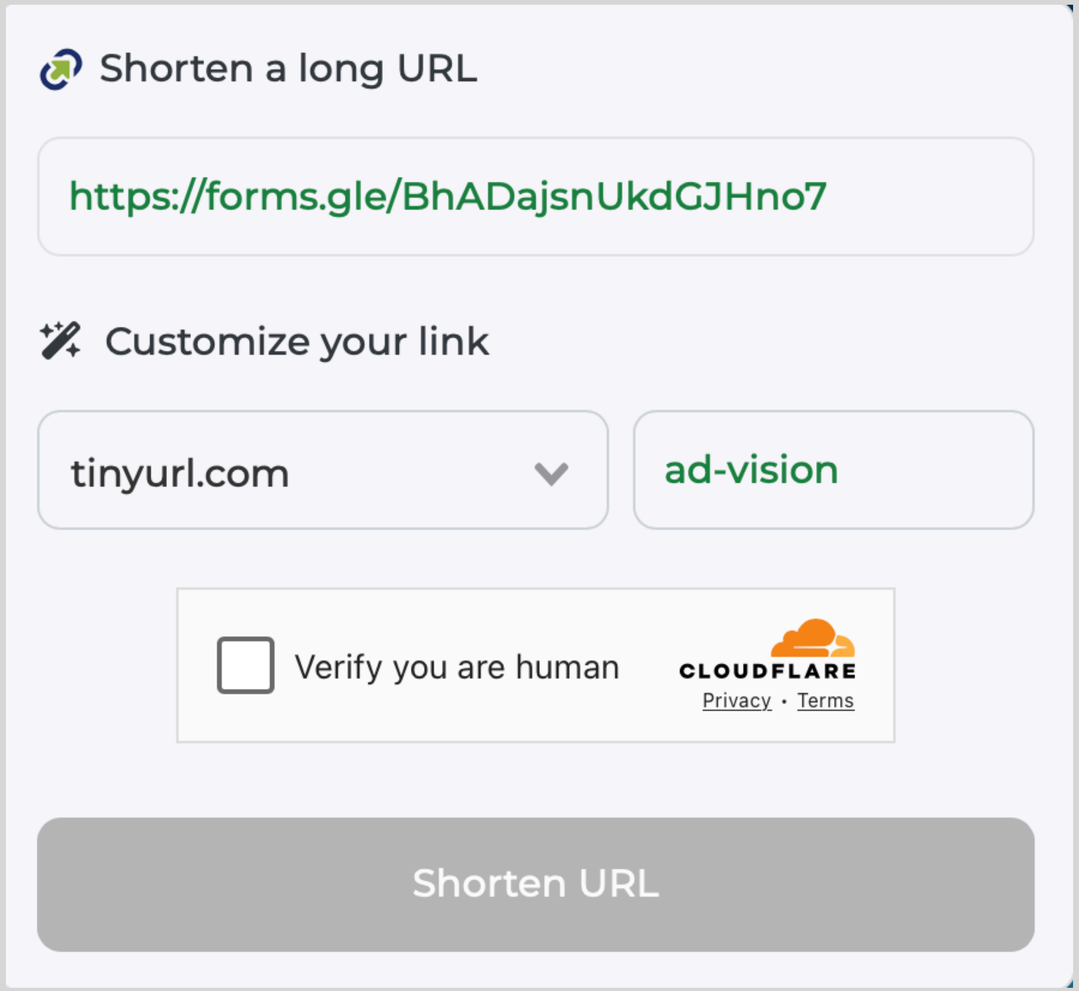
state

a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
  a seated Flag

actions

createSlot (time)
  effect creates a fresh slot for the time
reserve (user, time): reservation
  requires some slot at this time not yet reserved
  effect creates & returns a fresh reservation
    associates it with user and slot
seat (reservation)
  requires reservation is for about now
  effect marks reservation as seated

your turn: designing a URL shortening concept



define the core concept

for a URL shortener like tinyurl
consider simple enhancements

elements of your concept

name, purpose, principle
then actions, then states

for each action

say how it updates the state

one possible solution

concept UrlShortening
purpose shorter or more memorable way to link
principle after create generates a short url, lookup will return the original url
state
    a set of Shortenings with
        a targetUrl String
        a shortUrl String
actions
    register (shortUrlBase, targetUrl: String): (shortUrl: String)
        effect pick any shortUrl of the form shortUrlBase/foo that has not been used
        return it and create a shortening for it
    lookup (shortUrl: String): (targetUrl: String)
        requires some shortening with shortUrl
        effect returns targetUrl corresponding to it
    delete (shortUrl: String)
        requires some shortening with shortUrl
        effect removes the shortening

state invariants

aka integrity constraints

designing invariants for concepts

concept PasswordSession

state
a set of Users with
  a username String
  a password String

invariants?

at most one user with a given username

what goes wrong if violated?

concept RestaurantReservation

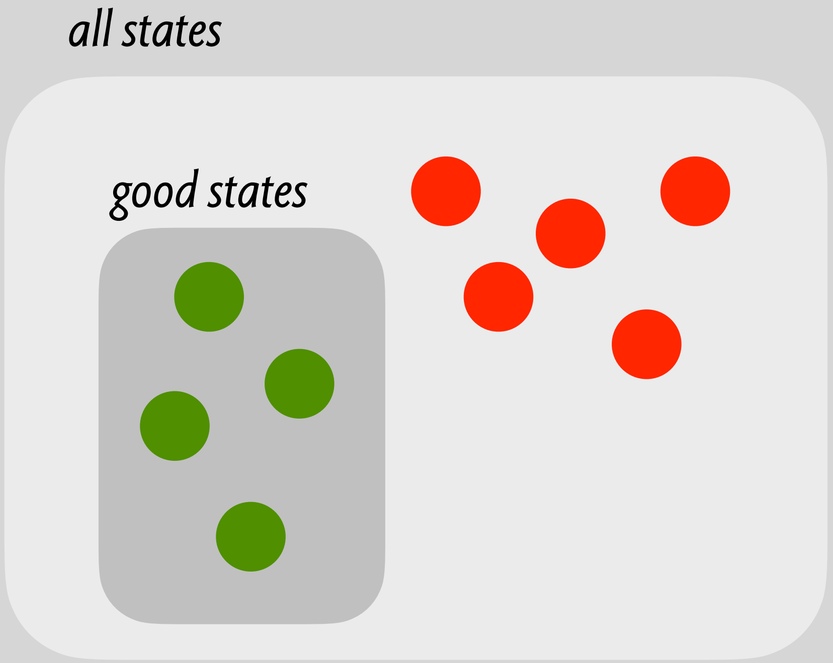
state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
  a seated Flag

at most one reservation for a given slot

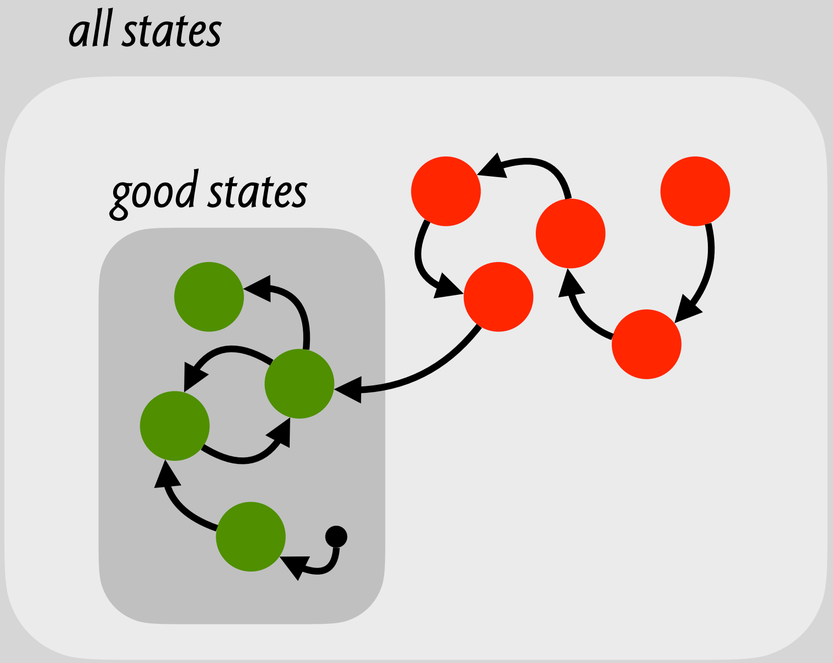
at most one reservation for a given user ?

reservation not seated if slot time before now? ?

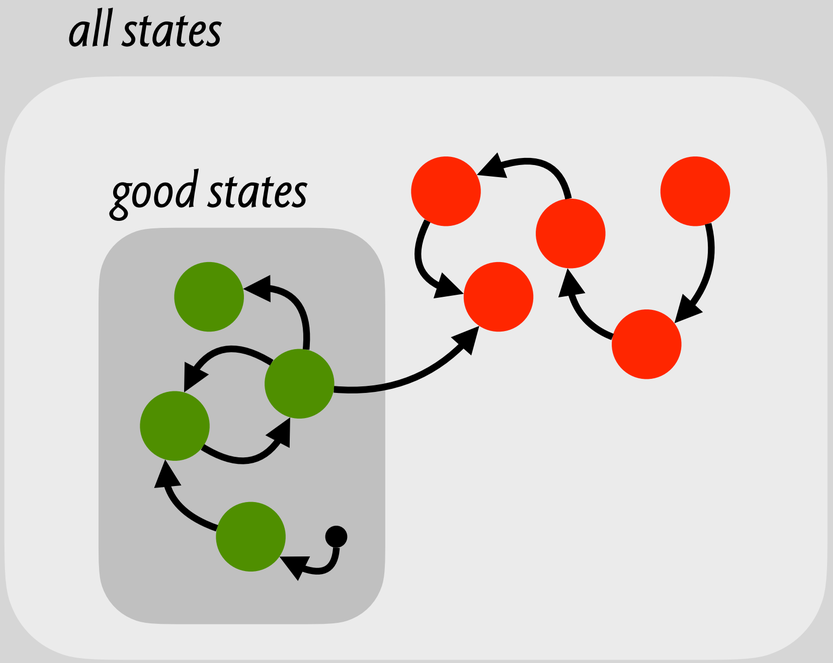
classifying states



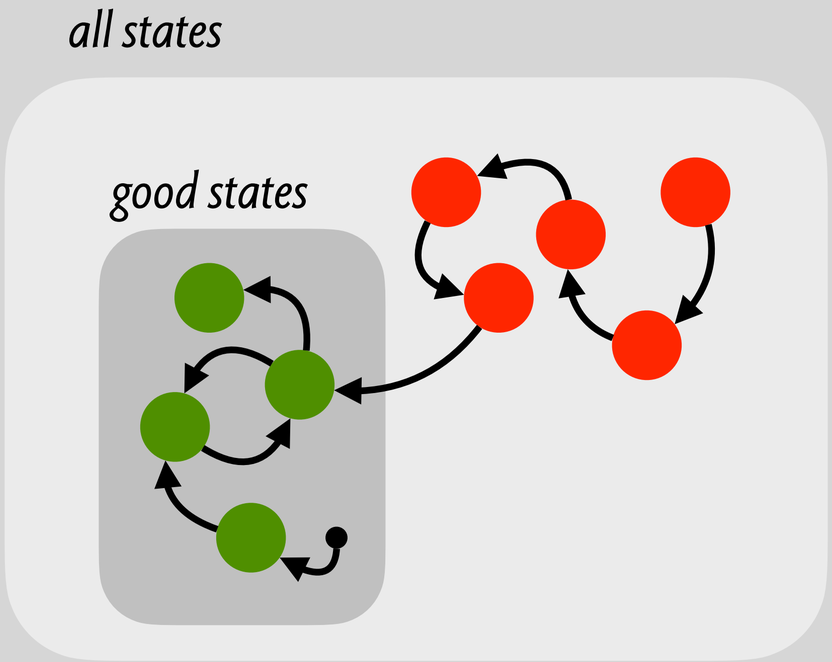
a safe design



an unsafe design



inductive reasoning strategy



what we want to avoid
reasoning about all scenarios
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 with
  a Time
a set of Reservations with
  a User
  a Slot
  a seated Flag

actions
  createSlot (time)
    effects creates a fresh slot for the time
  reserve (user, time): reservation
    requires some slot at this time not yet reserved
    effects creates & returns a fresh reservation
        associates it with user and slot
  seat (reservation)
    requires reservation is for about now
    effects marks reservation as seated

the invariant we want to check

at most one reservation for a given slot

check that the invariant holds in initial state

initially, no reservations

check each action preserves invariant

only the reserve action modifies set of reservations

reserve action ensures slot is not reserved

did your URL shortener have any invariants?

states & data models

getting more precise

simplifying the state

concept RestaurantReservation

state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot

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	Reservation	User
s0	r0	u1

these are SETS

time		user		slot	
s0	Ju..	r0	u1	r0	s0

these are BINARY RELATIONS

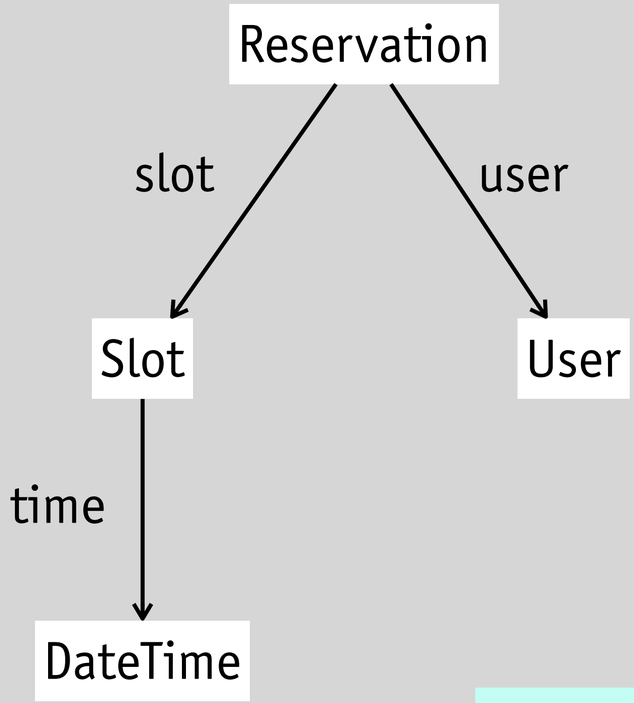
a diagrammatic form

Slot	Reservation	User
s0	r0	u1

these are SETS

time		user		slot	
s0	Ju..	r0	u1	r0	s0

these are BINARY RELATIONS



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

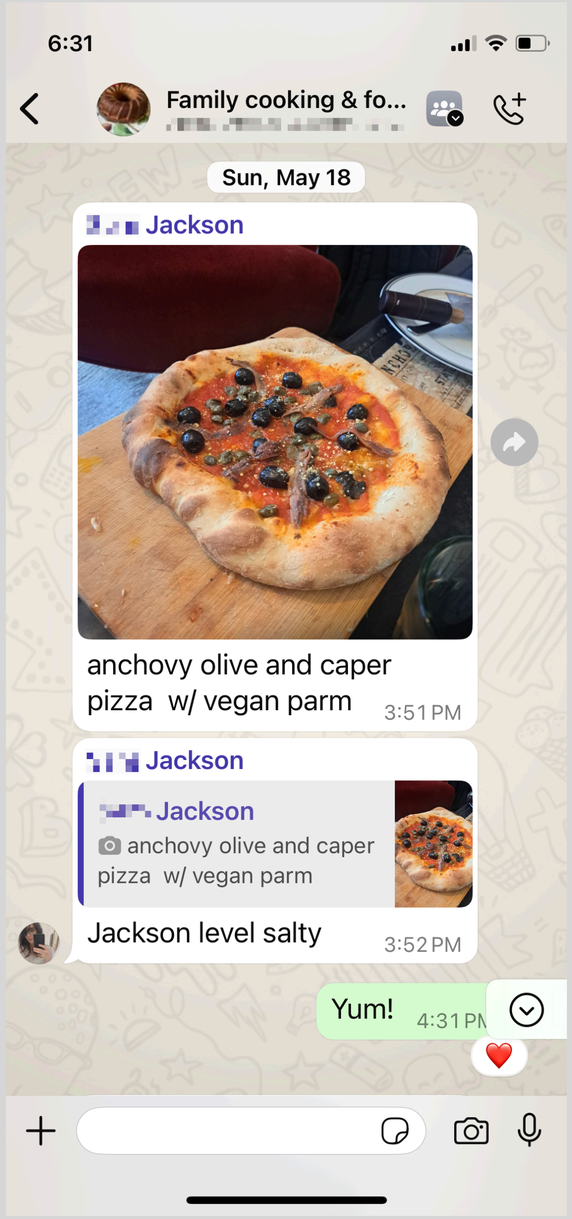
states can be represented as just sets & binary relations
never need tables with more than two columns

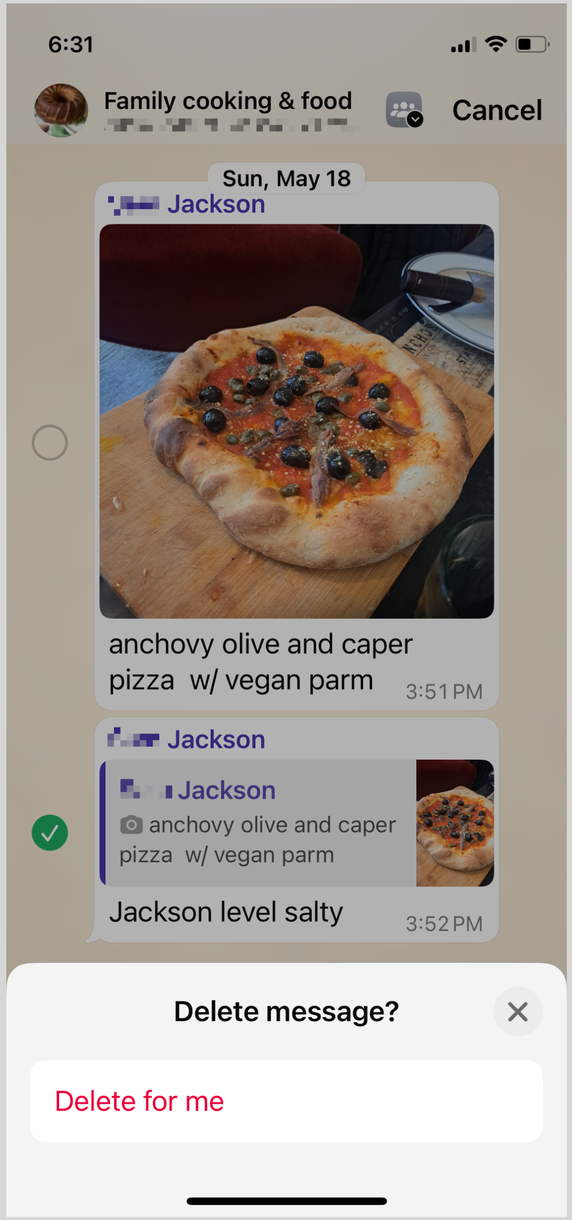
this allows a nice diagrammatic representation
this is the “entity relationship diagram”

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)

your turn: designing state for WhatsApp





some features shown here

sent & received messages

replies to messages

deleting messages

why might group members see different messages?

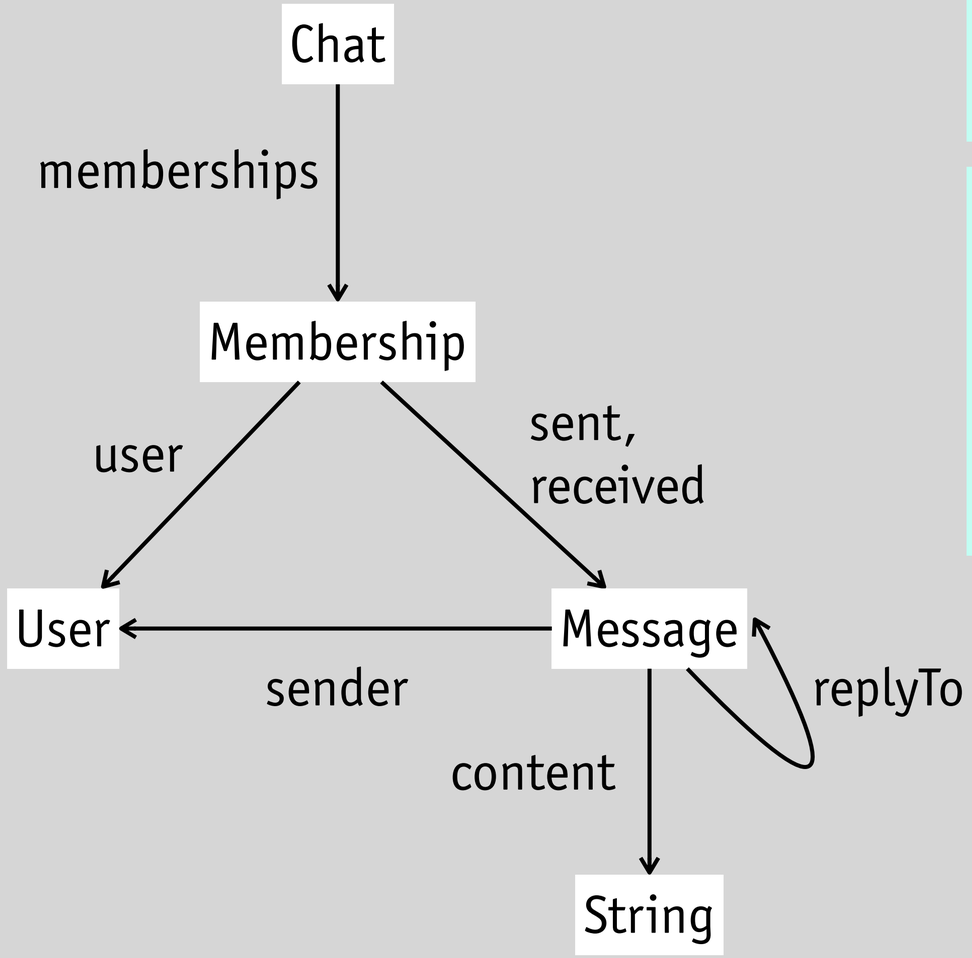
only see messages when member you can "delete for me"

can you define the state of WhatsApp's **GroupChat** concept? use a diagram or text

a possible solution

concept GroupChat

state
a set of Chat with
 a set of Memberships
a set of Memberships with
 a User
 a sent set of Messages
 a received set of Messages
a set of Messages with
 a sender User
 a content String
 an optional replyTo Message



why memberships vs. users as members?

user has different messages in each chat

why sender if have user's sent messages?

user may have deleted message others still see

heuristics
for states & actions

do you have enough actions?

is purpose/value delivered?
note that have info in 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?

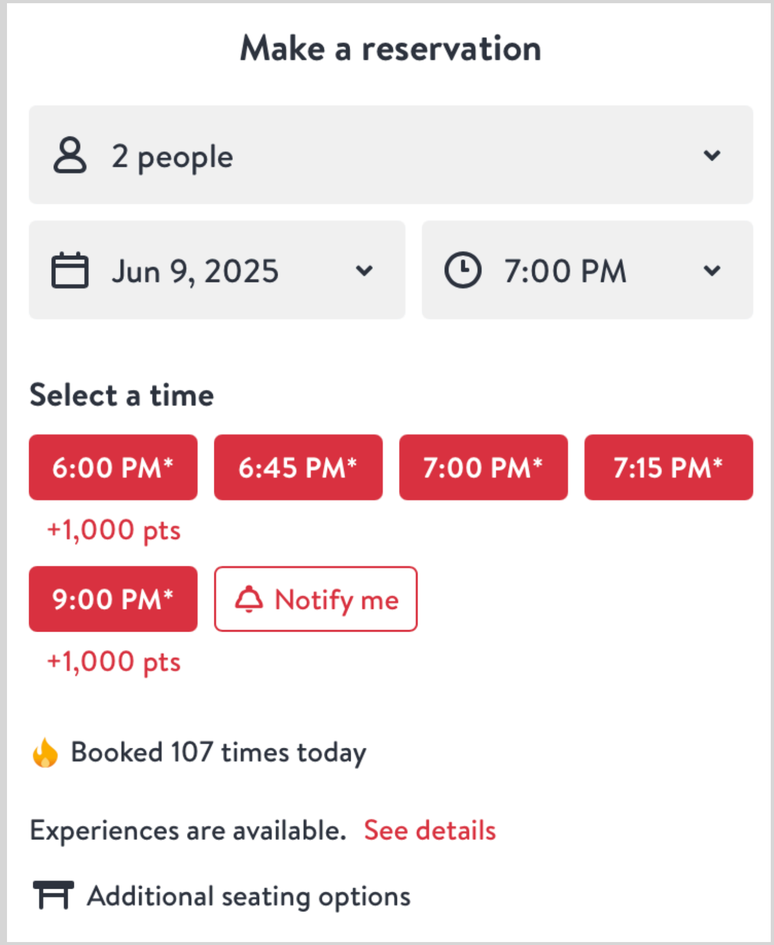
do all nouns have create, update, delete?
for associated state?

seat action?

create slots?

unseat?
cancel reservation?

change reservation?



concept Reservation
actions reserve...

do you have a rich enough state?

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

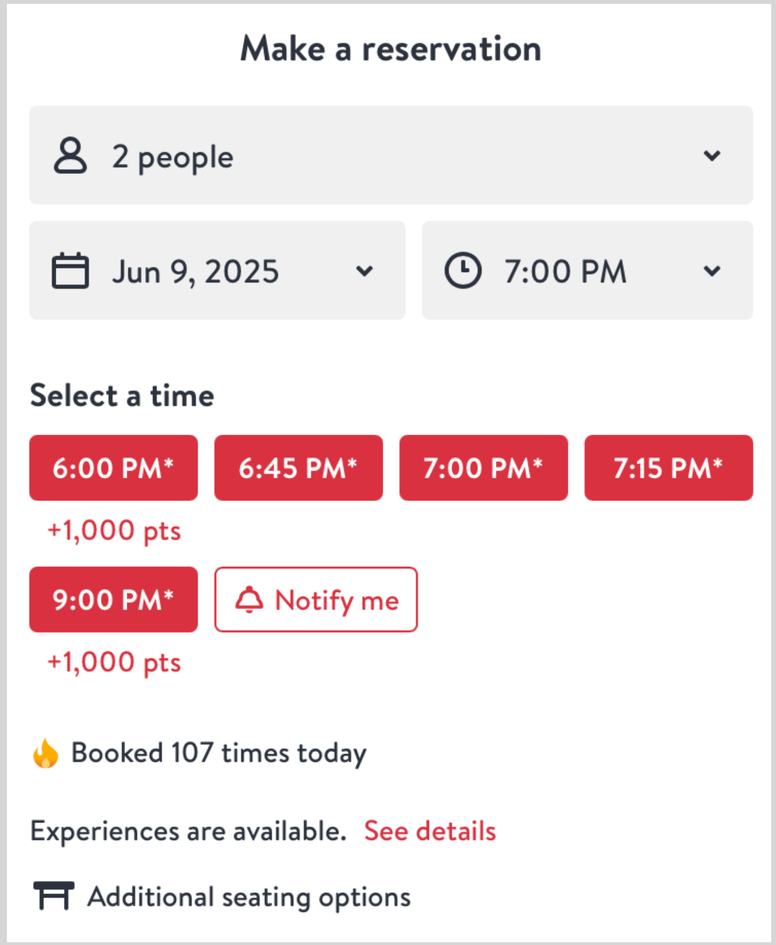
should you track history?
remember completions, deletions, undos?

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

table sizes?

retain after seat?

by vs. for?
time of reservation?



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

check your understanding

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

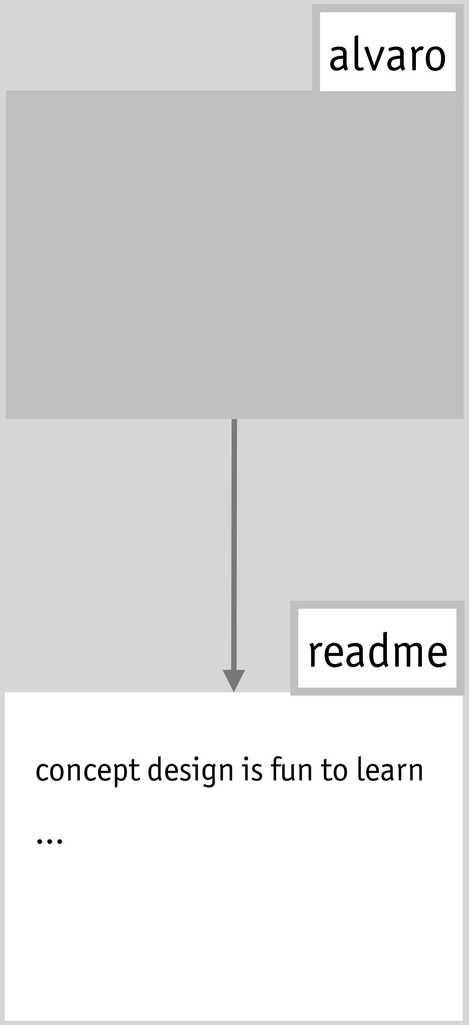
(a) User behavior will generally be more constrained

(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 with
  a name String
  a contents set of Files or Folders
a set of Files with
  a name String
  a body Text

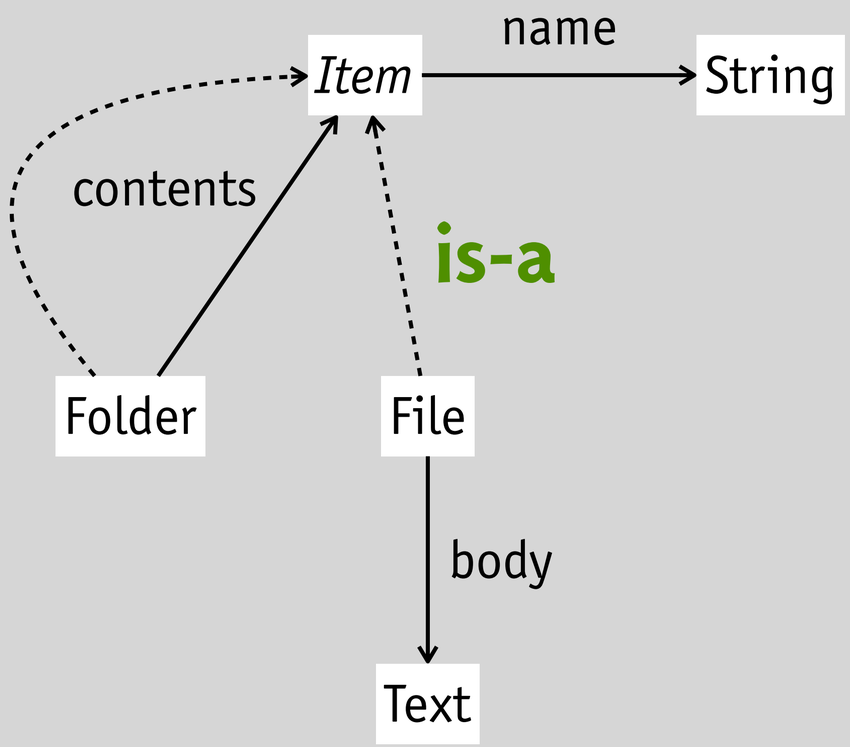


diagram introduces a new trick
an arrow for is-a (aka subset)
allowing sets for generalization

what invariants?

alvaro

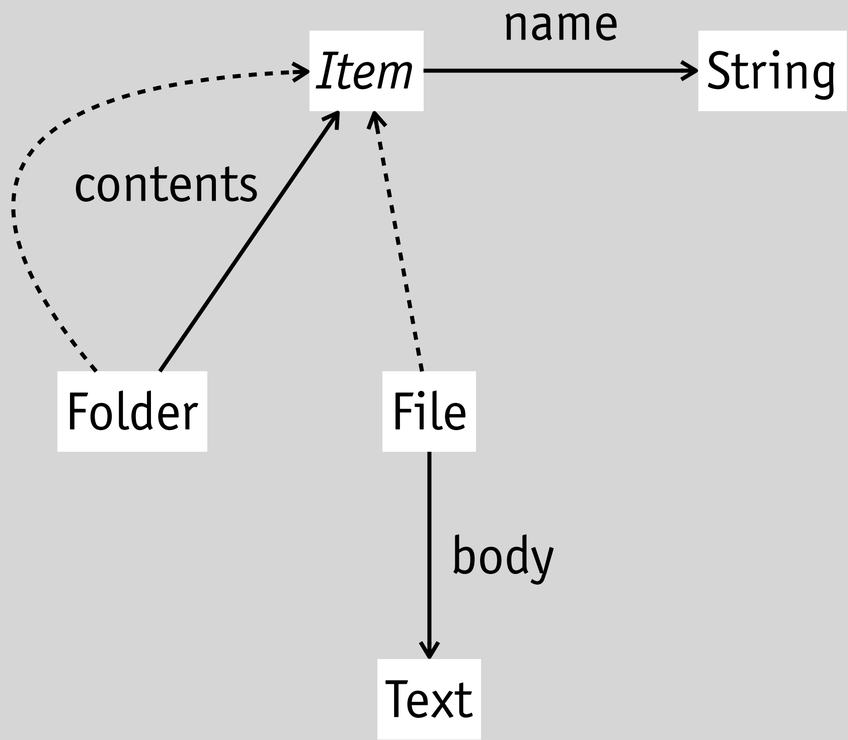
readme

concept design is fun to learn

...

concept SimpleFolder

state
a set of Folders with
  a name String
  a contents set of Files or Folders
a set of Files with
  a name String
  a body Text



some invariants

✔ every file belongs to a folder

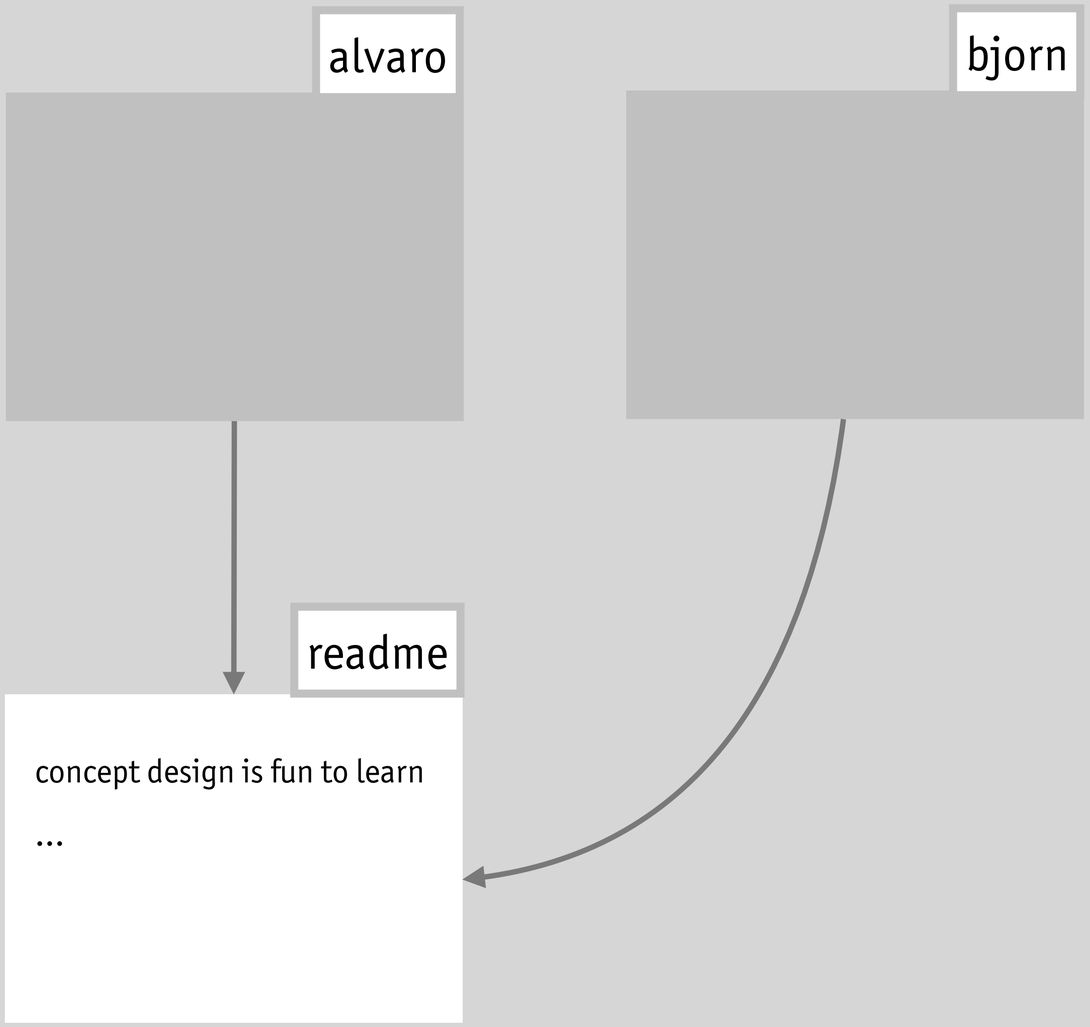
✔ no folder contains itself (directly or indirectly)

✔ 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



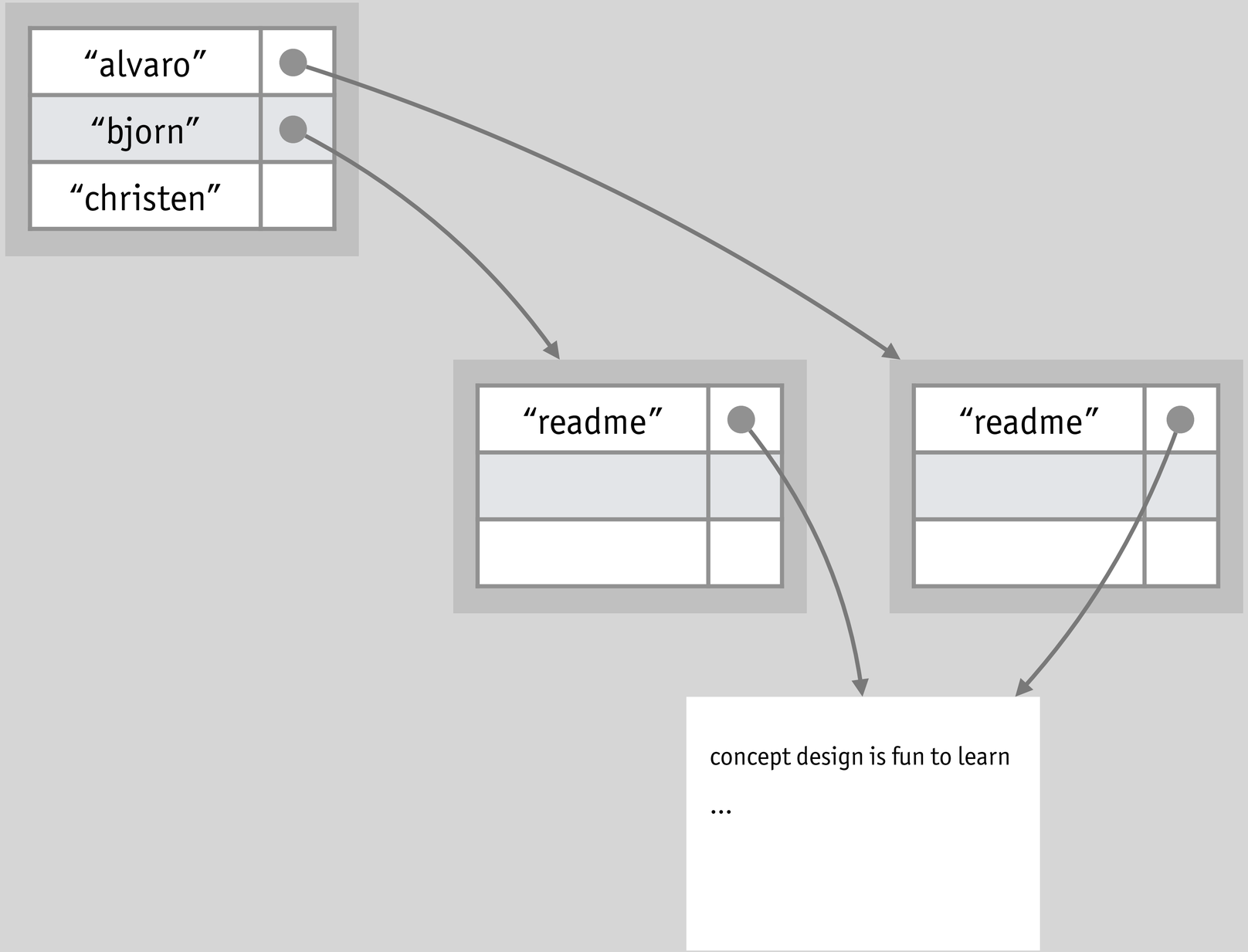
what might go wrong in this situation?

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



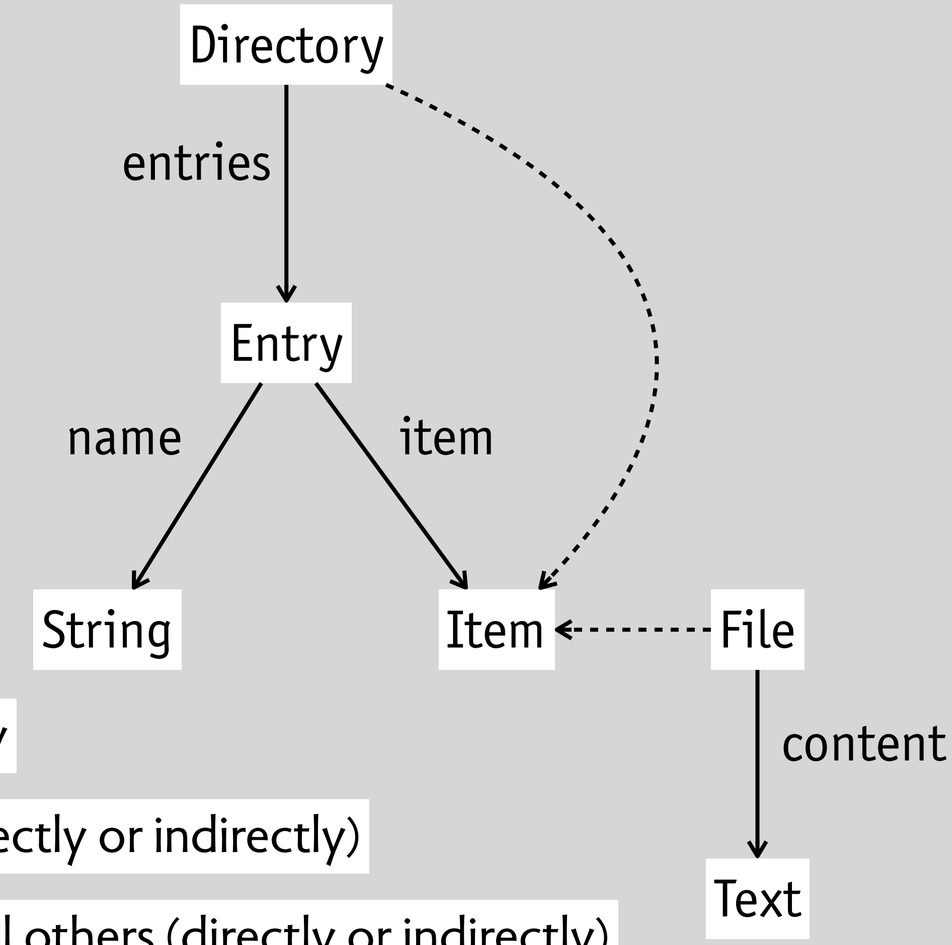
concept UnixDirectory

state
a set of Directories with
  a set of Entries
a set of Entries with
  a name String
  an item Directory or File
a set of Files with
  a body Text

the state of the Unix directory concept

concept UnixDirectory

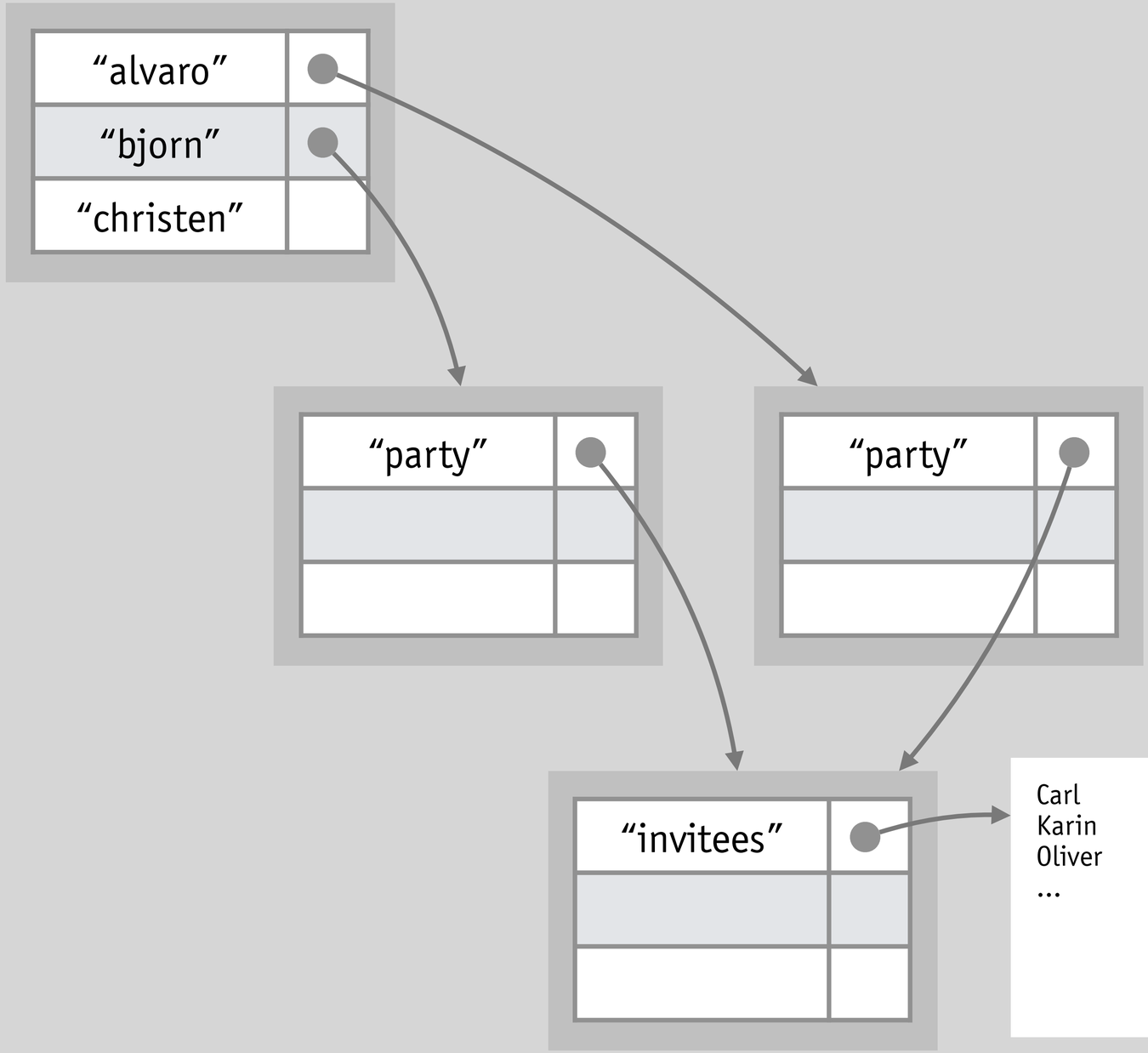
state
a set of Directories with
  a set of Entries
a set of Entries with
  a name String
  an item Directory or File
a set of Files with
  a body Text



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

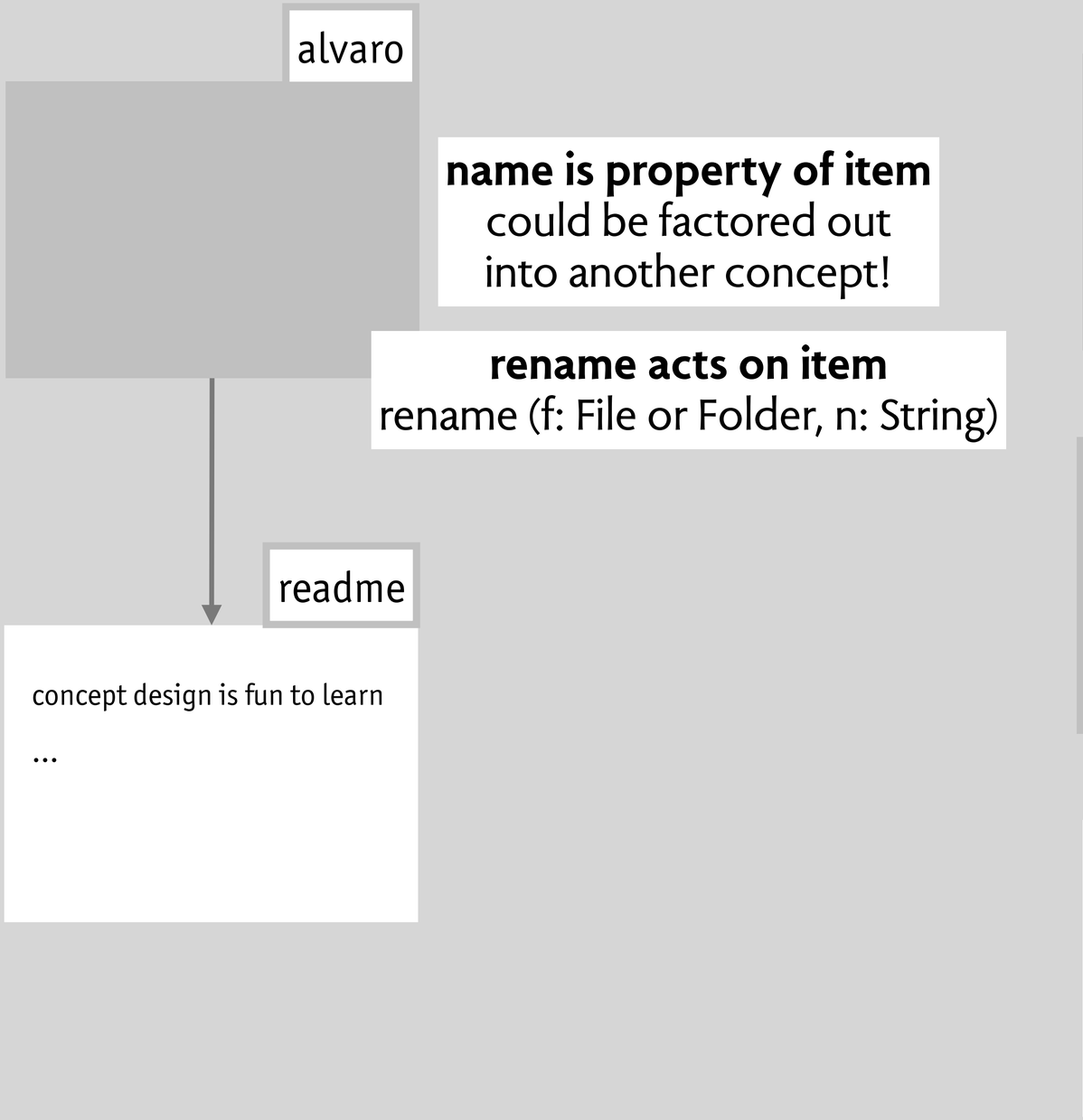
any user can change a name
only need to check uniqueness locally

what might go wrong in this situation?

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

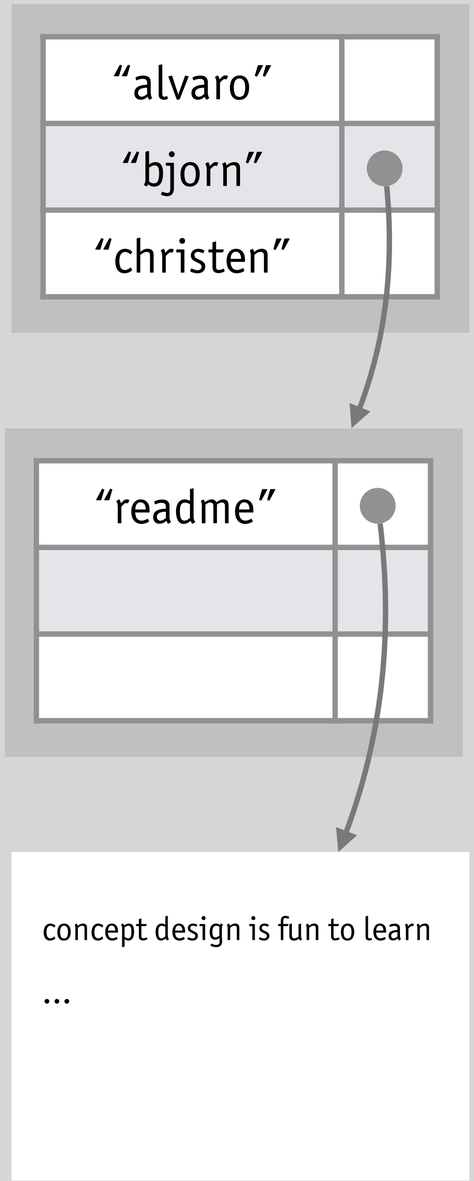
deletion removes an entry not an item
so might still be reachable!

a fine distinction with major impacts



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

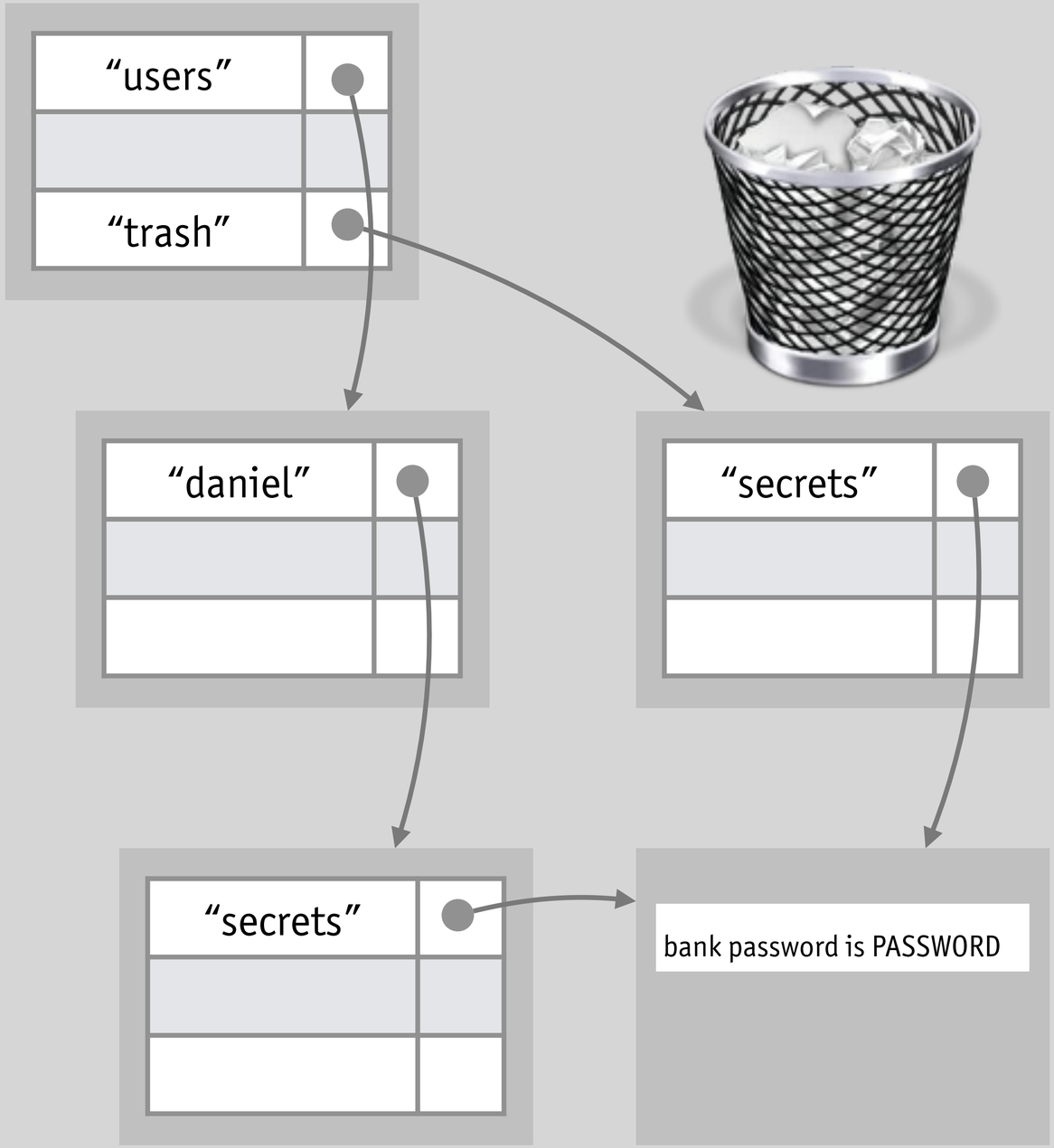
rename acts on item
rename (f: File or Folder, n: String)



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

choosing details

underlying behavior, not user interface

behavior = states + actions

a classical model, used by OOP too

actions

requires (when) & effects (what)

state models

relations, diagrams and invariants