6.104 · Software Design · MIT · Fall 2025

modularity in design

Daniel Jackson

your goals for today's class

know what goes wrong when modularity is poor
confuses users, restricts & breaks functionality, prevents reuse, ...

grasp idea of separation of concerns
organizing functions around concerns, not objects

learn from examples of better modularity
how aspects of an object are split across concepts

have the specificity principle in mind
one purpose :: one concept

de/composition:
design is breaking up & putting together

decomposing into parts with purposes

teapot

keeping hot

lid

handle

holding

spout

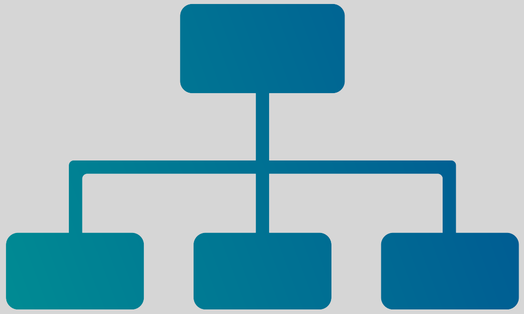
pouring

body

brewing

how does decomposition help?

what do you think?



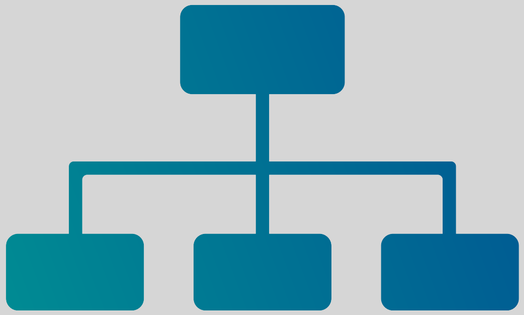
incrementality
division of labor
steady progress

reuse
build on experience
within & across products

usability
identify familiar parts
learn what you need

focus
one part at a time
localize changes

the two watchmakers



incrementality
division of labor
steady progress

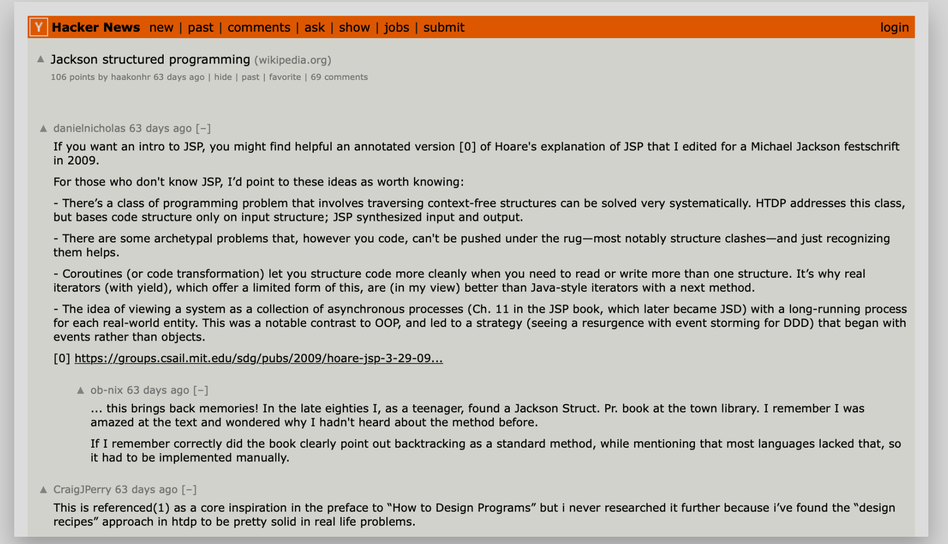
Herb Simon, The Architecture of Complexity (1962)

cartoon by ChatGPT

how unique is it?

reuse

build on experience
within & across products



no other app is the same as HackerNews

HackerNews = Post + Comment + Upvote + Karma + ...

but its concepts are mostly identical to the concepts in other apps

Dijkstra: separation of concerns

focus
one part at a time
localize changes

"Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects.

It is what I sometimes have called "the separation of concerns", which, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts, that I know of. This is what I mean by "focussing one's attention upon some aspect": it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously.

Edsger Dijkstra, On the role of scientific thought (EWD447, 1974)

example: separating concerns for a modular design

a bad concept with poor modularity

concept UserAccount

purpose ????

state
a set of User with
- a username String
- a password String
- an email String
- a phone String
- a displayName String
- a profile Image

is this good modularity?

not incremental
modules like this often >10kloc
can it be tested before it's all done?

not reusable
a dumping ground for all user-related function
more & more app-specific over time

not focused
what if user wants a different email for messages?
how & where to make this change?

separating concerns

concept UserAccount

purpose ????

state
a set of User with
 a username String
 a password String
 an email String
 a phone String
 a displayName String
 a profile Image

concept UserNaming

purpose name users

state
a set of User with
 a username String

concept PasswordAuth [User]

purpose authenticate users

state
a set of User with
 a password String

concept Notification [User]

purpose notify users

state
a set of User with
 an email String
 a phone String

concept Profile [User]

purpose share user info

state
a set of User with
 a displayName String
 a profile Image

a more modular design

concept UserNaming

purpose
let users refer to each other by name

principle
after registering with a name, the user can be found by looking up by that name

state
a set of User with
a username String

actions
register (n: String): (u: User)

concept PasswordAuth [User]

purpose
authenticate users with passwords

principle
after setting a password for a user, the user can authenticate with that password

state
a set of User with
a password String

actions
setPassword (u: User, p: String)
authenticate (u: User, p: String)

concept UserProfile [User]

purpose
let users share personal info

principle
after setting a name and image for a user, other users can see them

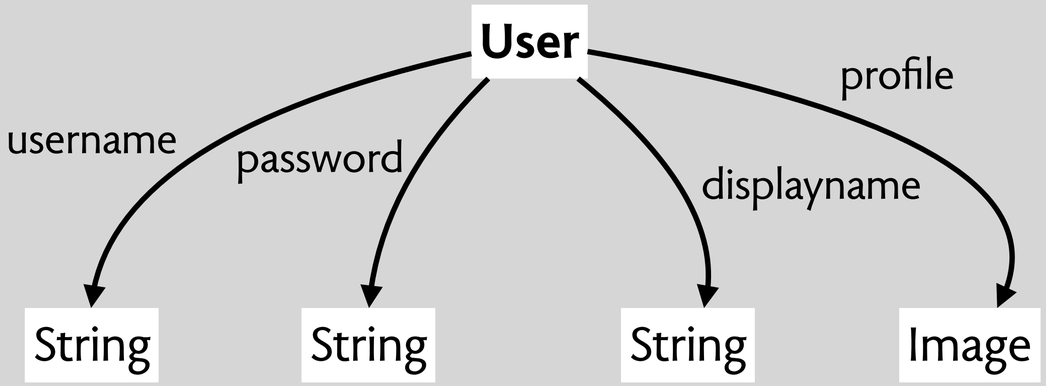
state
a set of User with
a displayname String
a profile Image

actions
setName (u: User, n: String)
setImage (u: User, i: Image)

when
Request.createAccount (name)
then
UserNaming.register (name)

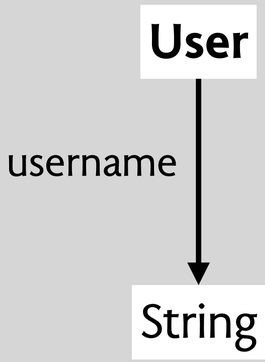
when
UserNaming.register (name): (user)
Request.createAccount (password)
then
PasswordAuth.setPassword (user, password)

factoring the data model

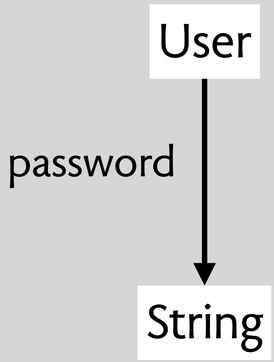


a set of User with
- a username String
- a password String
- a displayname String
- a profile Image

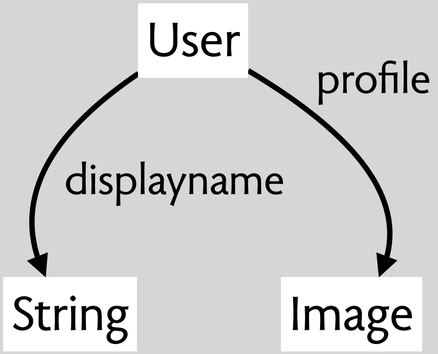
a set of User with
a username String



a set of User with
a password String



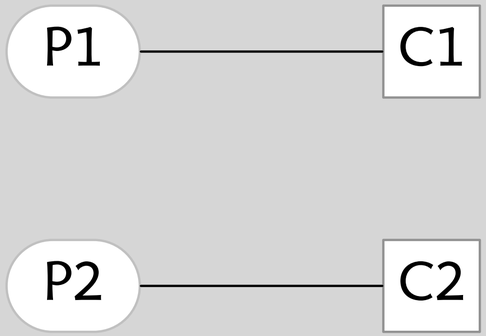
a set of User with
a displayname String
a profile Image



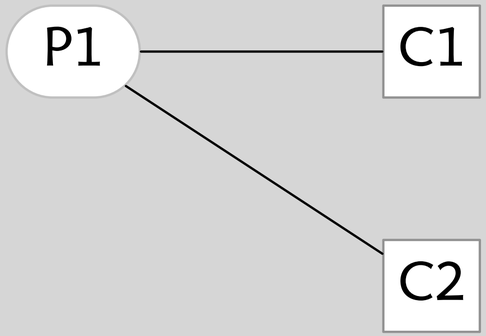
User = {u0, u1}
username = {(u0, n0), (u1, n1)}
password = {(u0, p0), (u1, p1)}

concept design principles

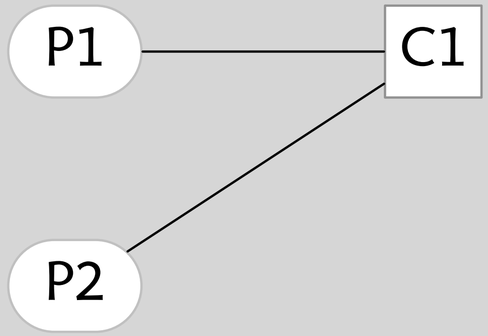
specificity
purposes:concepts are 1:1



redundancy
>1 concept per purpose



overloading
>1 purpose per concept

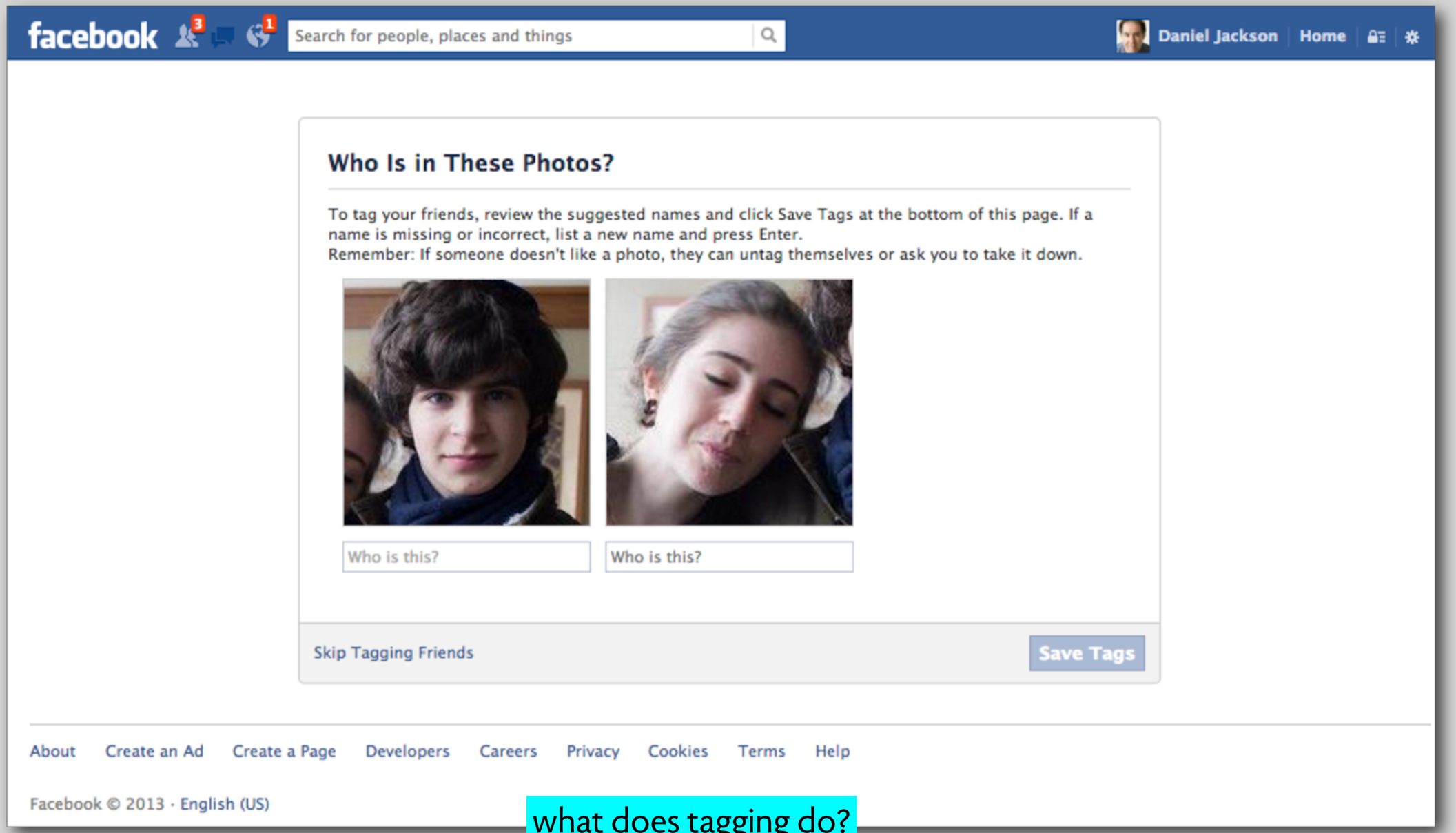


Mitchell and Webb on “unity of purpose”

https://www.youtube.com/watch?v=vlN17gMhnEk

example:
how non-modularity confuses users

tagging in facebook



what does tagging do?

so who sees a post you're tagged in?

Who can see my photos and photos I'm tagged in on Facebook?

Computer Help | Copy Link

Your photos and the photos you're tagged in are visible to:

- The audience they're shared with.
- The people tagged in the photo.
- Friends that the people tagged choose to add to the audience.

To adjust who can see photos you've uploaded, change the audience or remove them. If you've been tagged in a photo you don't like, you can remove the tag or ask the person to take it down.

Keep in mind, photos and posts hidden from your timeline are still visible to these people other places on Facebook, such as in News Feed and search. To see and review posts you've hidden from your timeline, go to your activity log.

what does “choose to add” mean?

If I'm tagged in a post on Facebook, will my friends be able to see it, even if it wasn't shared with them?

Computer Help ▾ | Copy Link

When someone tags you in a post, it will be visible to:

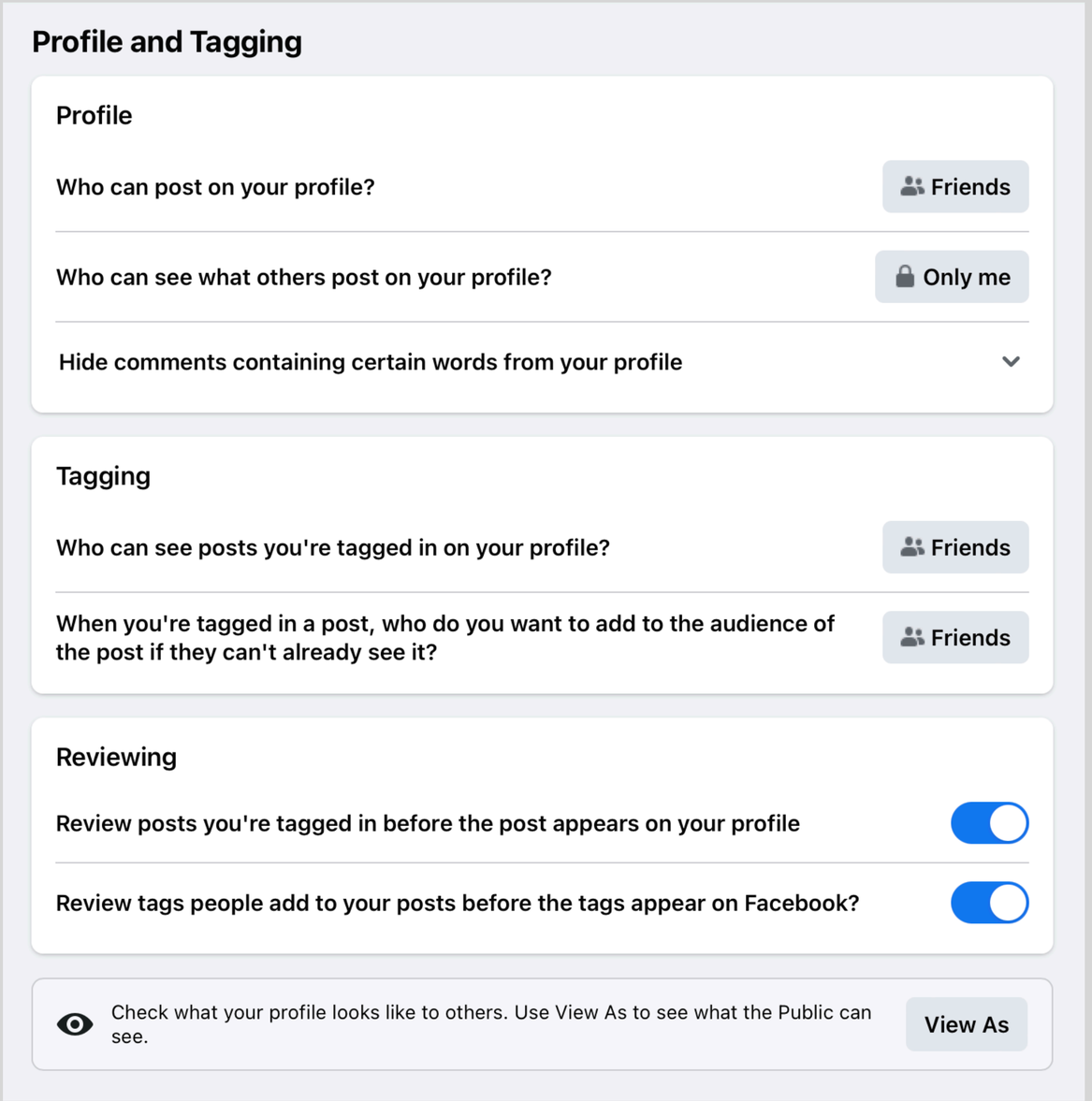
- The audience selected by the person who made the post.
- The audience you indicate in your Profile and Tagging settings. You can choose to automatically add your friends, select specific friends or not add anyone to the audience of the post you're tagged in.

To choose your audience for posts you're tagged in:

1. Click ▾ in the top right of Facebook.
2. Select **Settings & Privacy**, then click **Settings**.
3. In the left column, click **Profile and Tagging**.
4. Look for the setting **When you're tagged in a post, who do you want to add to the audience if they aren't already in it?** and click **Edit** to the far right.
5. Select an audience (example: **Only Me**).

Keep in mind:

- This setting defaults to **Friends**, which means when you’re tagged in a post, you and your friends can see the post, even if they weren’t in the original audience.
- The post you're tagged in may be shared with the original audience, as well as the friends you suggest. These people may see it in News Feed, search and other places on Facebook.



your turn

how would you fix the tagging problem?

goals might include
preserving modularity
not violating users' privacy
giving users flexibility

a more modular design

concept Tagging [Image, User]

purpose
share who is in an image

principle
after a user tags another user in an image, viewers can see the tag and identify the user

state
a set of Image with
a set of Tag
a set of Tag with
a tagging User
a tagged User

actions
tag (by: User, i: Image, who: User): Tag

concept Friending [User, Item]

purpose
let users limit access to their items

principle
after a user adds another friend as a user, and then publishes an item, the friend can access it

state
a set of User with
a friends set of User
a published set of Item

actions
addFriend (u: User, friend: User)
publish (u: User, i: Item)
access (u: User, i: Item)
requires item i is published by a user who is a friend of the user u

a reasonable sync

when
Tagging.tag (by, image, who): (tag)
then
Friending.publish (by, tag)

an unreasonable sync:
acting on the user's behalf

when
Tagging.tag (by, image, who): (tag)
then
Friending.publish (who, image)

example:
how non-modularity restricts functionality

a lovely camera fuji x100

complex menu system: image quality setting

aspect ratio

image size setting

non-standard ratio + raw?

problem #1: no non-standard ratio unless also save JPG!

raw image showing non-destructive aspect ratio crop

problem #2: very few ratio options

change.org

Petition details Comments Updates

Fuji, give us 4:3, 5:4, and 6:7 aspect ratios on X-series cameras

636 have signed. Let's get to 1,000!

At 1,000 signatures, this petition is more likely to be **featured in recommendations!**

how would you fix this problem?

a more modular design

concept ImageQuality

purpose
set quality and format for images

principle
after setting the quality and format, photos taken will use that setting

state
an element QualitySetting with
a resolution of SMALL or MED or LARGE
a compression of SUPER or FINE or NORMAL
a format of RAW or JPEG or BOTH

actions
setCompression (...)
setFormat (...)
setResolution (...)

concept AspectRatio

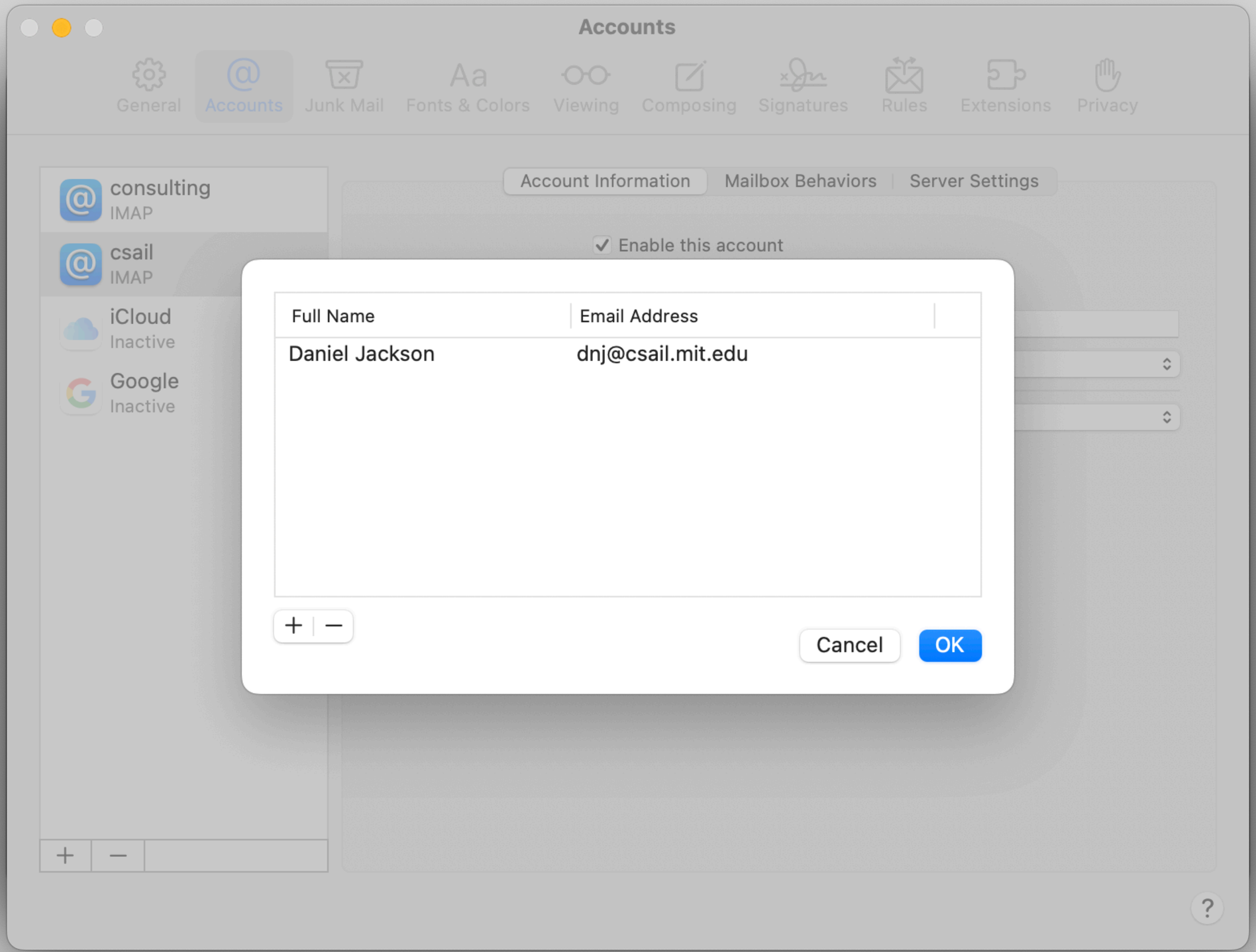
purpose
set aspect ratio for images

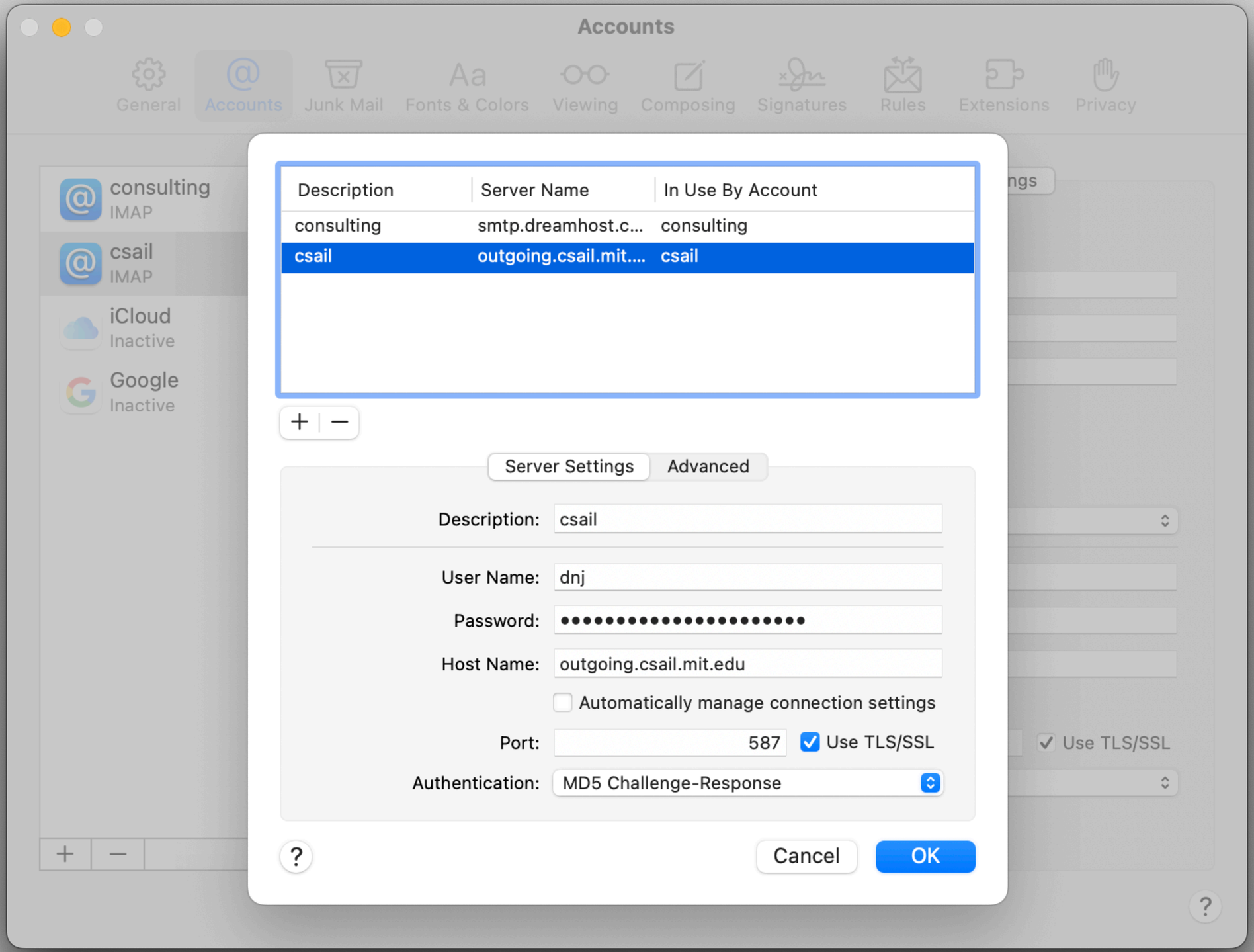
principle
after setting the aspect ratio, photos taken will use that ratio (by cropping for JPEGs and non-destructive framing for RAWs)

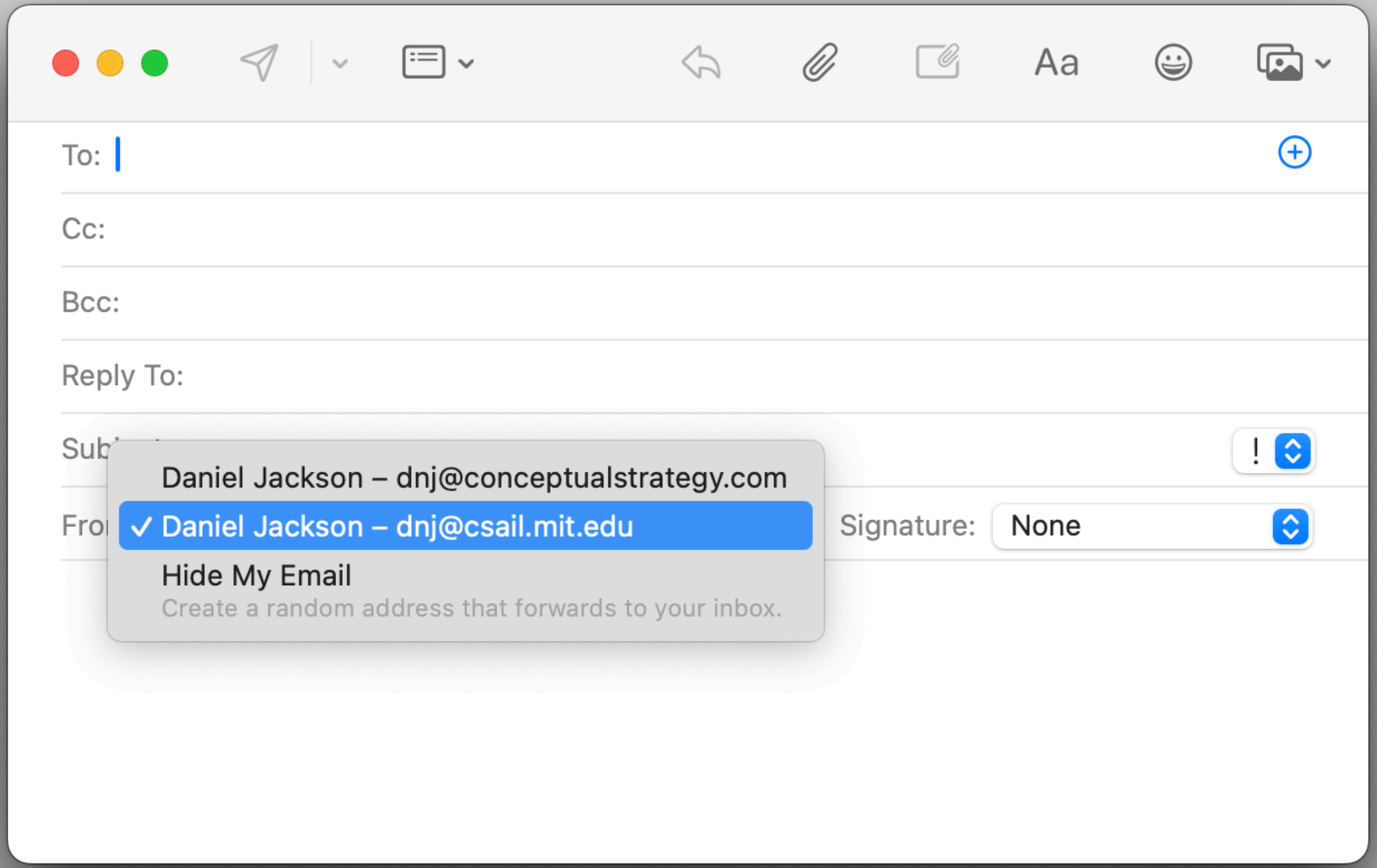
state
a set of Ratio with
a longSide Number
a shortSide Number
an element RatioSetting with
a Ratio

actions
addRatio (long: Number, short: Number)
setRatio (ratio: Ratio)

example:
how the wrong modularity can break functionality

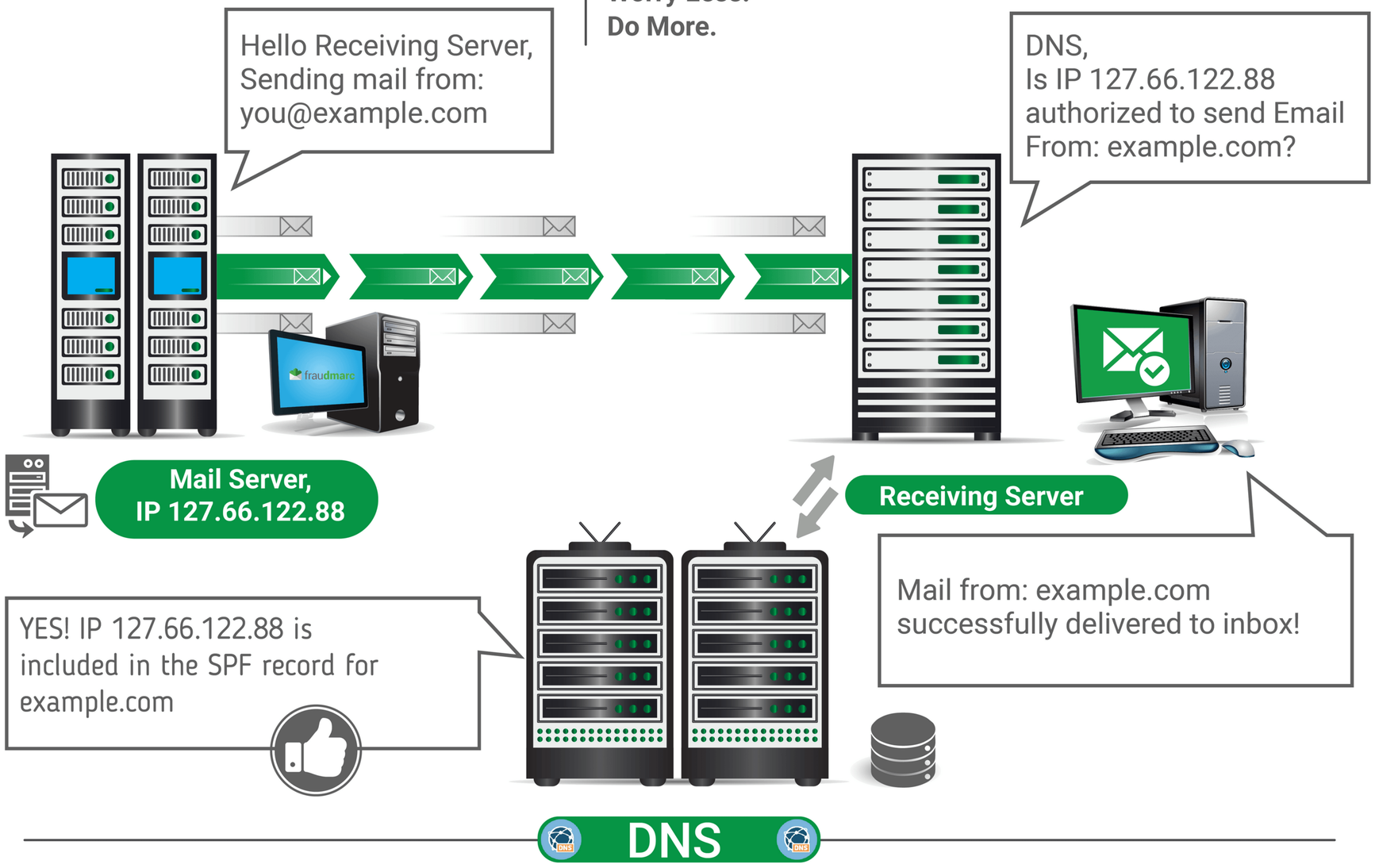

Accounts
General
Accounts
Junk Mail
Fonts & Colors
Viewing
Composing
Signatures
Rules
Extensions
Privacy
consulting
IMAP
csail
IMAP
iCloud
Inactive
Google
Inactive
Account Information
Mailbox Behaviors
Server Settings
Enable this account
Full Name
Email Address
Daniel Jackson
dnj@csail.mit.edu
Cancel
OK


Accounts
General
Accounts
Junk Mail
Fonts & Colors
Viewing
Composing
Signatures
Rules
Extensions
Privacy
consulting
IMAP
csail
IMAP
iCloud
Inactive
Google
Inactive
Description
Server Name
In Use By Account
consulting
smtp.dreamhost.c...
consulting
csail
outgoing.csail.mit....
csail
Server Settings
Advanced
Description:
csail
User Name:
dnj
Password:
Host Name:
outgoing.csail.mit.edu
Automatically manage connection settings
Port:
587
Use TLS/SSL
Authentication:
MD5 Challenge-Response
Cancel
OK
Use TLS/SSL


Aa
To:
Cc:
Bcc:
Reply To:
!
Daniel Jackson – dnj@conceptualstrategy.com
✓ Daniel Jackson – dnj@csail.mit.edu
Hide My Email
Create a random address that forwards to your inbox.
Signature:
None

fraudmarc

Worry Less.
Do More.



This shows how SPF works to authorize email. What happens to email that are not authorized by SPF can be complicated and is either dictated by the domain's DMARC policy or left up to the ISP's discretion. For more information, refer to the "About DMARC" page.

	SMTP server	*from email address*
select email address		dnj@csail.mit.edu
select SMTP server	outgoing.csail.mit.edu	
send message	outgoing.csail.mit.edu sends message from dnj@csail.mit.edu	
select email address		dnj@mit.edu
select SMTP server	smtp.mit.edu	
send message	smtp.mit.edu sends message from dnj@mit.edu	
receive message	imap.csail.mit.edu gets message sent to dnj@csail.mit.edu	
reply to message		dnj@csail.mit.edu
send message	smtp.mit.edu sends message from dnj@csail.mit.edu	

marked as **spam** by recipient because
IP address of smtp.mit.edu not included in SPF record for csail.mit.edu

a design with improved modularity

current design: email address is independent of choice of server

concept EmailAccount [Server]

state

a set of Account with

a displayName String

an emailAddress String

an incoming Server

an outgoing Server

actions

new (...): Account

setOutgoing (a: Account, s: Server)

better design: email address is associated with choice of server

concept EmailAccount [Server]

state

a set of Account with

an incoming Server

an outgoing Server

actions

new (...): Account

setOutgoing (a: Account, s: Server)

concept EmailSending [Server]

state

a set of Server with

an emailAddress String

a displayName String

actions

configure (..., s: Server)

can still factor out server authentication settings

concept ServerAuthentication

state

a set of Server with

a domain String

a username String

a password String

actions

register (...): Server

connect (s: Server)

...

example: how non-modularity leads to accidents

AS

Arvind Satyanarayan November 15, 2018 at 2:04 PM

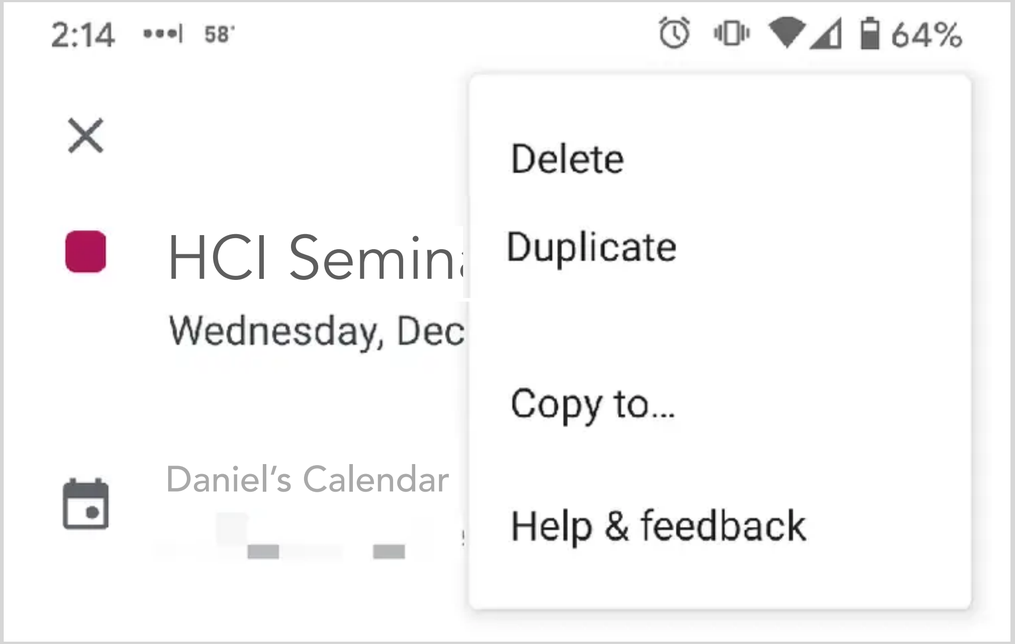
Re: TALK: Monday 11-19-2018 Kanit (Ham) Wongsuphasawat: No...

Details

Cc: seminars@csail.mit.edu, HCI-Seminar@lists.csail.mit.edu

This message is from a mailing list. Unsubscribe

Despite some erroneous messages sent to this list accidentally, Kanit's talk is happening! Please join us on Monday.



Canceling and deleting events in the Google Calendar mobile app is similar to on a desktop.

1. First, open Google Calendar.
2. Tap on the event you wish to cancel.
3. Press on the three dots in the top right corner of the event window.
4. Select Delete.
5. Tap Delete event. Google Calendar will send a cancellation email to the guests.

Mar 22, 2021

https://wpamelia.com › Blog

How to Cancel an Event in Google Calendar - Amelia booking ...

seminar announced as **email** to listserv with attached calendar event

↓

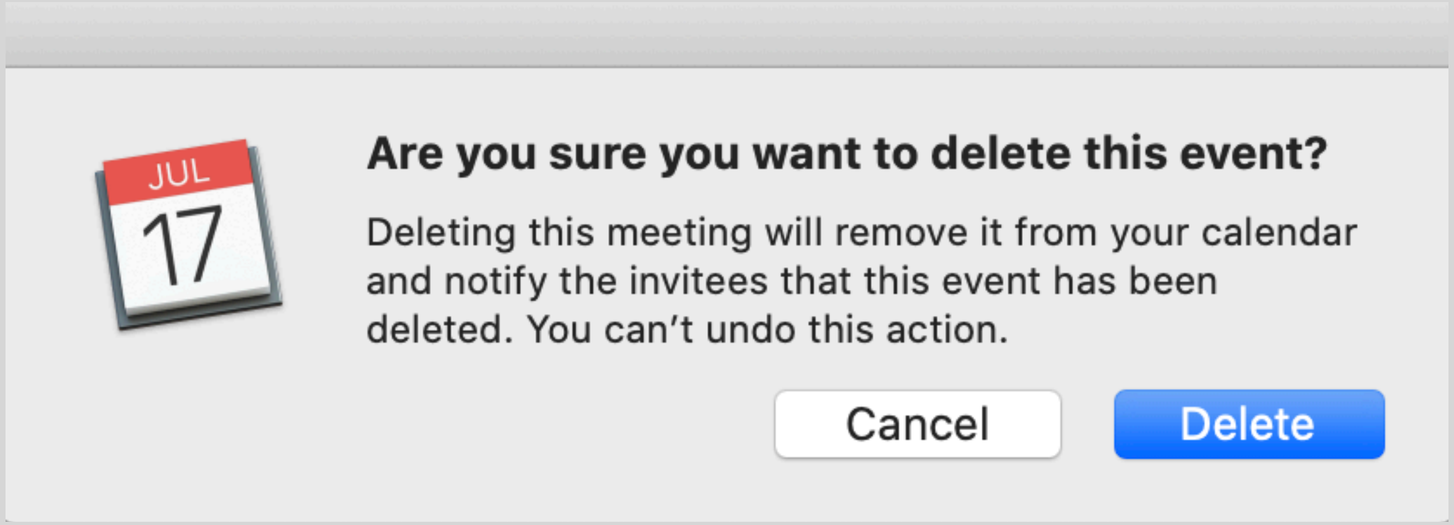
event **installed** automatically in user's calendar

↓

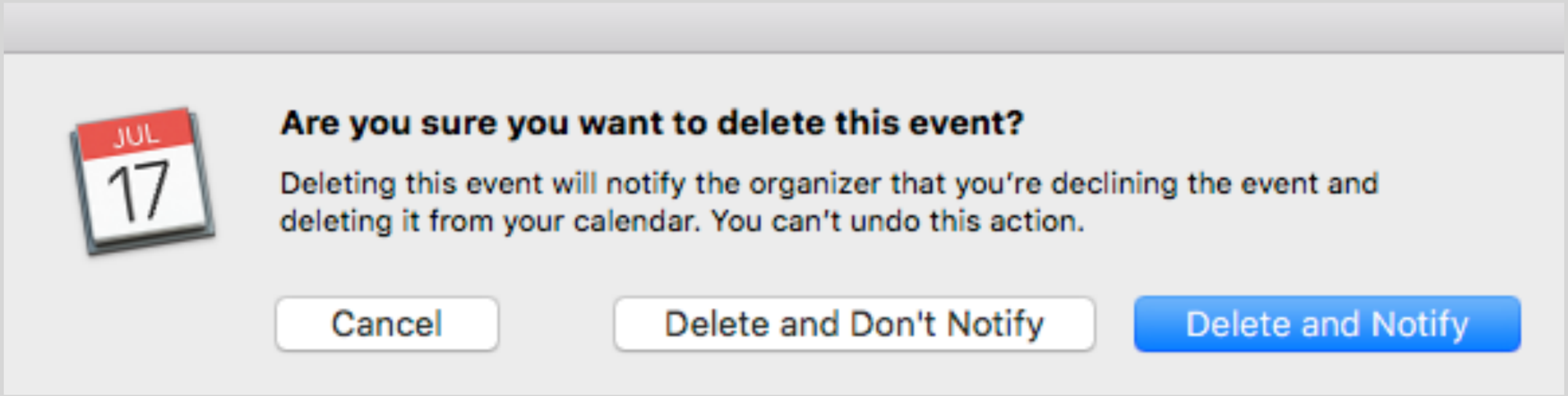
user **deletes** event from calendar

↓

cancellation email automatically sent to other invitees



a long time problem in iCal too
how to delete spam calendar events?



resolution to design problem
make sync optional

a more modular design

concept CalendarEvent [User]

state

a set of Calendar with
- a name String
- an events set of Event

a set of Event with
- a date Date
- a title String

actions

newCalendar (n: String): (Calendar)
newEvent (c: Calendar, d: Date, t: String)
deleteCalendar (c: Calendar)
deleteEvent (e: Event)

concept Inviting [User, Event]

state

a set of Event with
- a host User
- an invited set of User
- an accepted set of User
- a declined set of User
- a canceled Flag

actions

invite (host: User, u: User, e: Event)
accept (u: User, e: Event)
decline (u: User, e: Event)
cancel (e: Event)

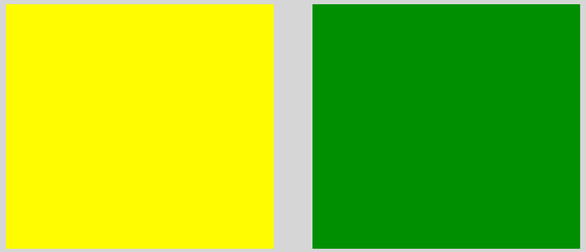
delete and cancel are now decoupled

modularity

3 criteria

defining modularity

separation
a single module doesn't
conflate unrelated concerns



separated: not conflated

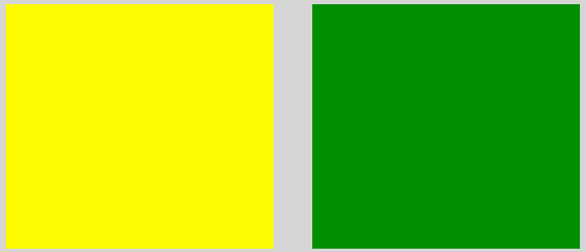
conflated

tagging/access

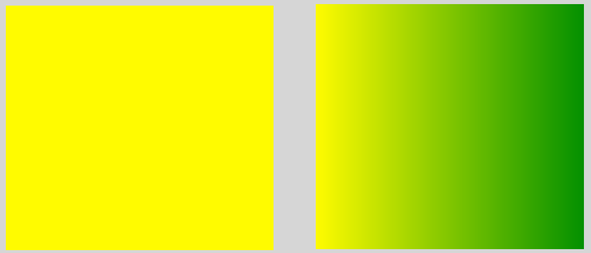
quality/ratio

invitation/event

completeness
a single module contains
all of a concern's behavior

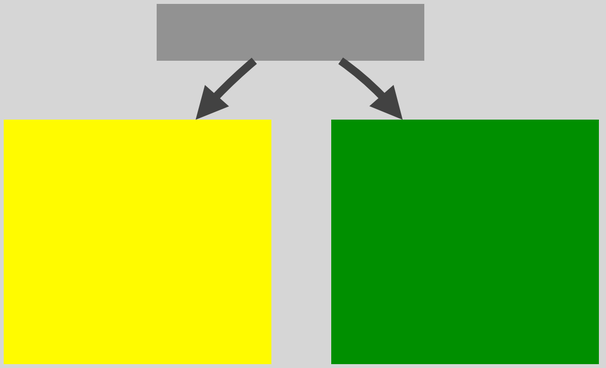


complete: not fragmented

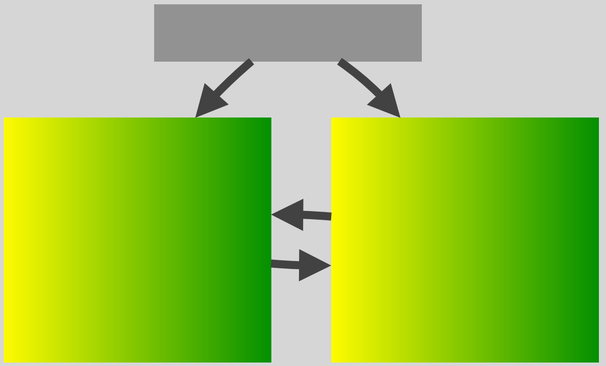


fragmented

independence
one module doesn't
rely on another



independent



dependent

takeaways

key ideas from this lecture

modularity matters

not just localizing change: user flexibility too

separation of concerns

not just grouping functions around objects

so far all about separation/decoupling

next time we'll talk about completeness