a new approach to software design analysis

Daniel Jackson · MIT CSAIL · ISSTA, Baltimore · July 15, 2015

a retraction

ISSTA, 1996

Elements of Style: Analyzing a Software Design Feature with a Counterexample Detector

Daniel Jackson and Craig A. Damon
School of Computer Science
Carnegie Mellon University

Abstract

We illustrate the application of Nitpick, a specification checker, to the design of a style mechanism for a word processor. The design is cast, along with some expected properties, in a subset of Z. Nitpick checks a property by enumerating all possible cases within some finite bounds, displaying as a counterexample the first case for which the property fails to hold. Unlike animation or execution tools, Nitpick does not require state transitions to be expressed constructively, and unlike theorem provers, operates completely automatically without user intervention. Using a variety of reduction mechanisms, it can cover an enormous number of cases in a reasonable time, so that subtle flaws can be rapidly detected.

'design' because, unless the specification is trivial, its development inevitably involves design decisions.

Our approach goes back to Guttag and Horning's paper 'Formal Specification as a Design Tool' [GH80], in which they show that, having characterized a design in a formal language, quite subtle questions can be framed about the design in the same formalism. Despite advances in theorem proving technology, however, it is still not possible just to feed such questions to a tool that will answer yes or no. Sobered by the apparent intractability of any specification language rich enough to describe interesting properties, researchers have since then turned their attention more to languages and away from analysis.

The analysis of hardware designs, in contrast, has advanced steadily since the early 1980s. Clarke's method, known as

alloy: a language & tool for relational models

about alloy

Alloy is a language for describing structures and a tool for exploring them. It has been used in a wide range of applications from finding holes in security mechanisms to designing telephone switching networks.

An Alloy model is a collection of constraints that describes (implicitly) a set of structures, for example: all the possible security configurations of a web application, or all the possible topologies of a switching network. Alloy's tool, the Alloy Analyzer, is a solver that takes the constraints of a model and finds structures that satisfy them. It can be used both to explore the model by generating sample structures, and to check properties of the model by generating counterexamples. Structures are displayed graphically, and their appearance can be customized for the domain at hand.

At its core, the Alloy language is a simple but expressive logic based on the notion of relations, and was inspired by the Z specification language and Tarski's relational calculus. Alloy's syntax is designed to make it easy to build models incrementally, and was influenced by modeling languages (such as the object models of OMT and UML). Novel features of Alloy include a rich subtype facility for factoring out common features and a uniform and powerful syntax for navigation expressions.

The Alloy Analyzer works by reduction to SAT. Version 4 was a complete rewrite that included Kodkod, a new model finding engine that optimizes the reduction, and a new front end.

contact us!

news

A Japanese translation of book published!

Revised edition of book now out!
Available from MIT Press.

a sad story

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!

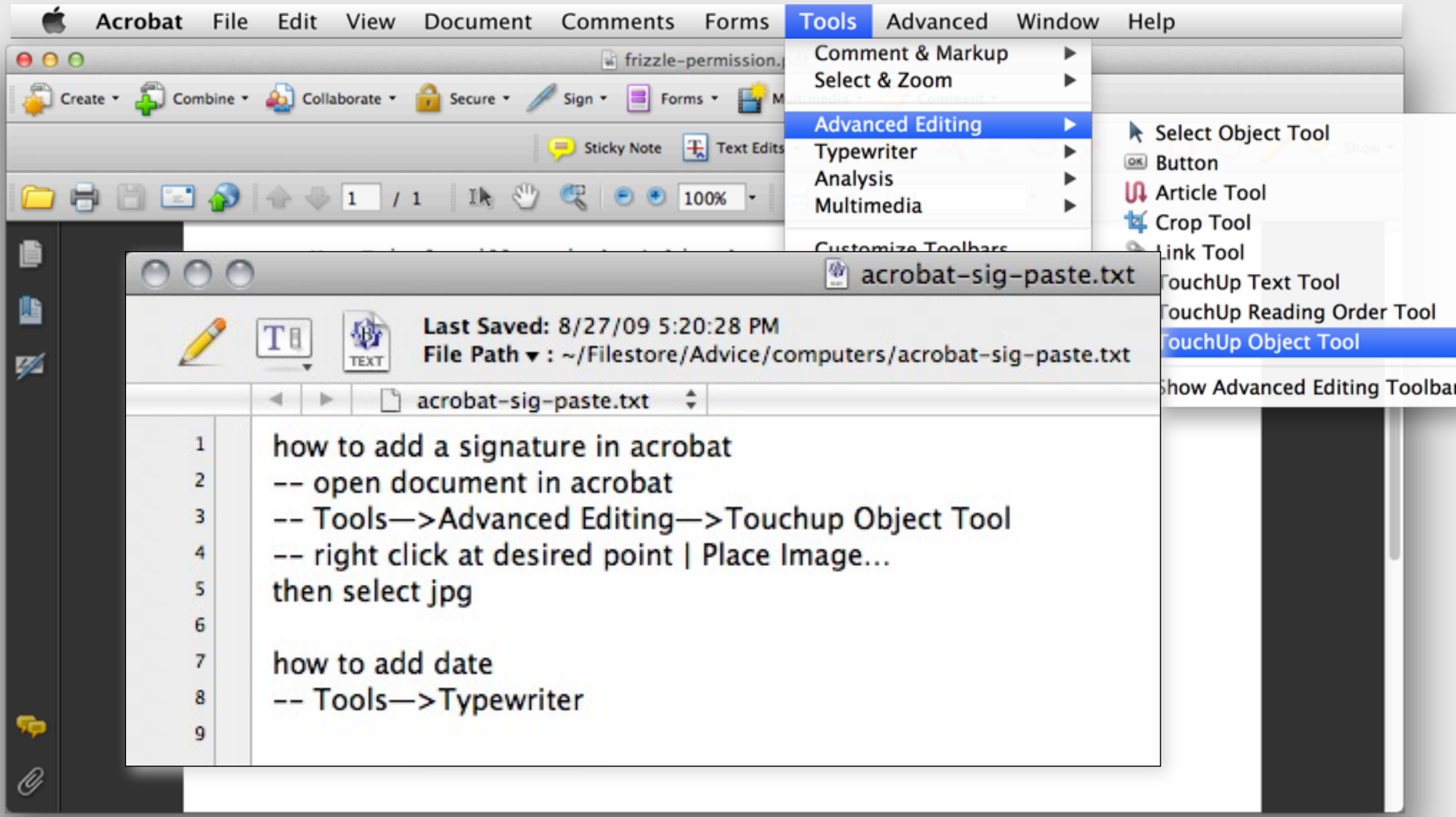
Yes, I give permission for my child to go on the second grade "Touch and Feel" trip on Friday February 13th to the NastyCo Nuclear Dump. I understood that my child may encounter the normal risks of childhood play, including grazed knees, hurt feelings and exposure to toxic waste.

Count Olaf

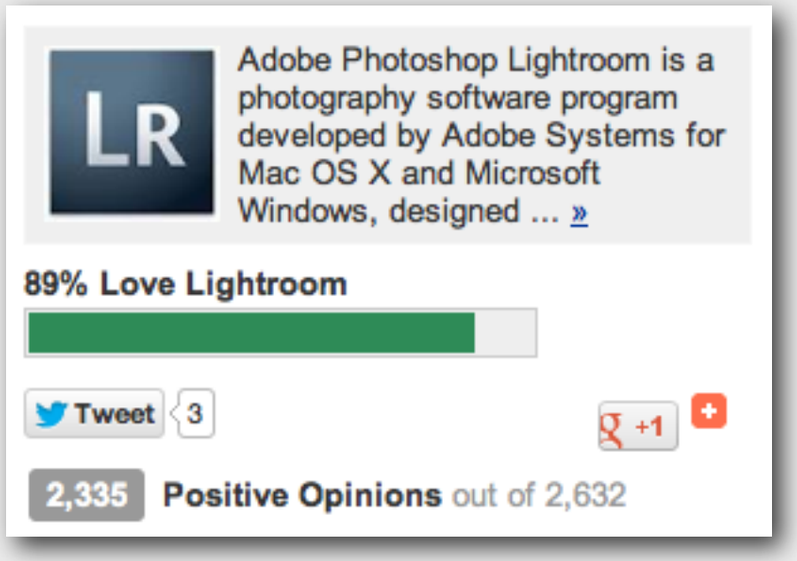
February 11, 2013

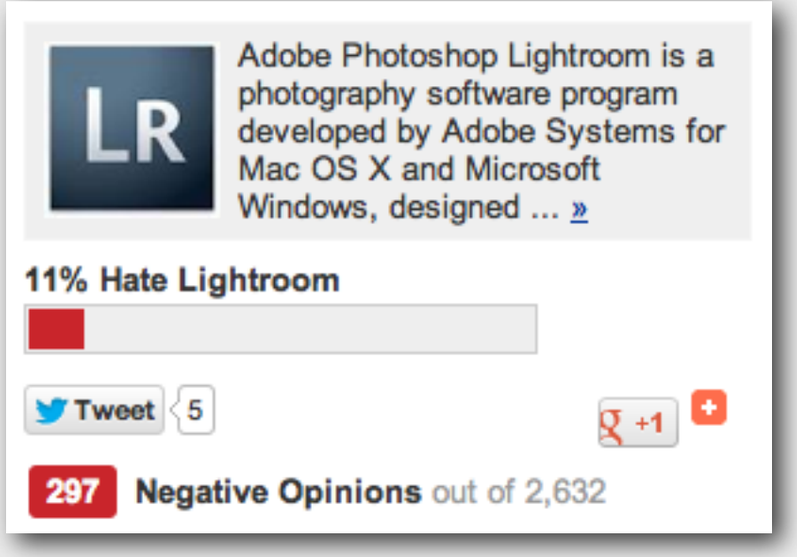
Parents signature Date

acrobat to the rescue



what we hate ... & love





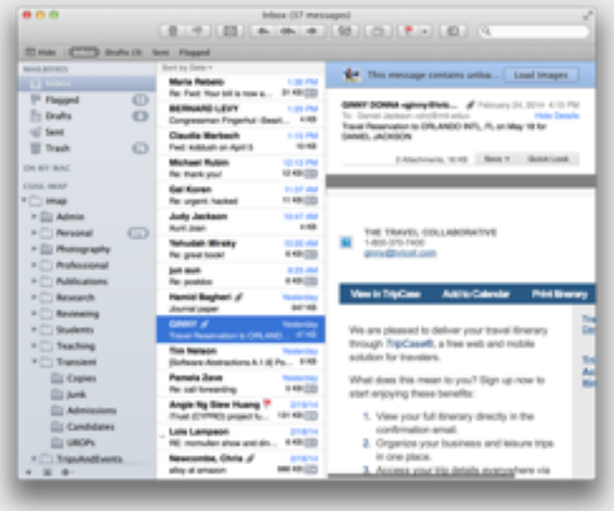
from http://amplicate.com

concepts

what characterizes an app?

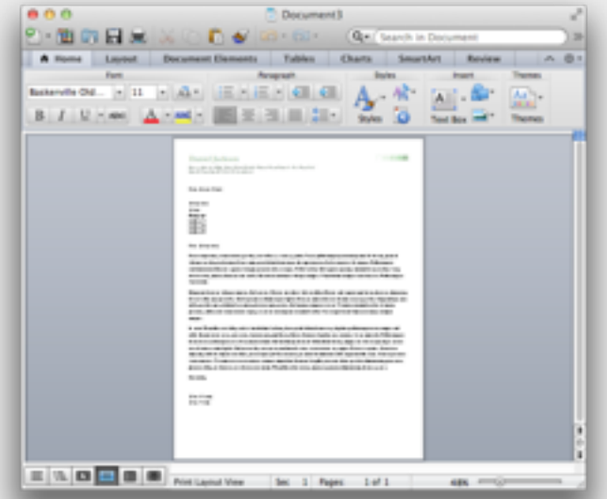
concepts!

Apple Mail



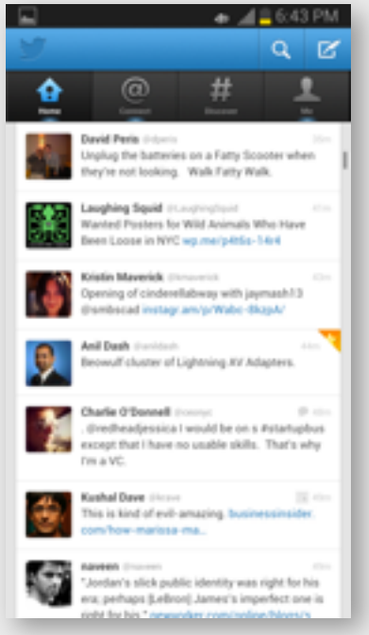
EmailAddress
Message
Folder or Label

Microsoft Word



Paragraph
Format
Style

Twitter



Tweet
Hashtag
Following

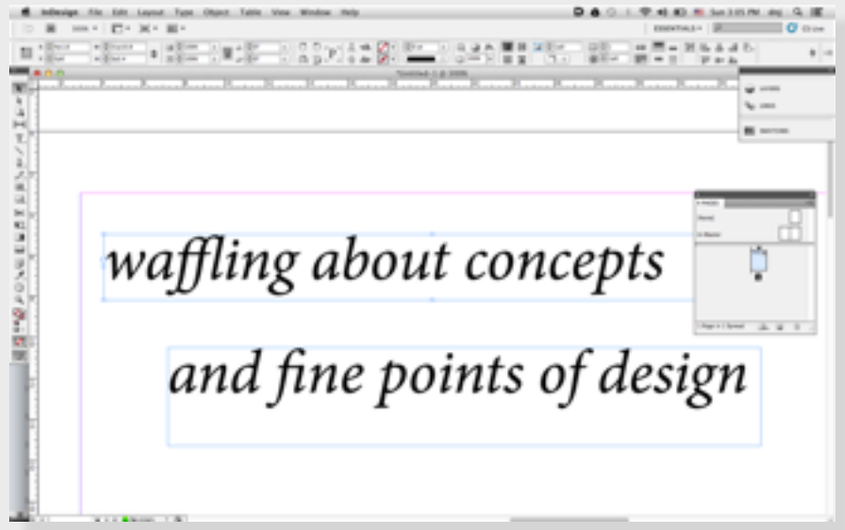
Photoshop

PixelMap
Layer/Mask
Adjustment

concepts define classes

text editor
line, buffer,
character set

word processor
paragraph,
format, style



desktop publishing app
text flow, link,
page template

jamonh
Oct 22, 2013 7:19 PM

Just upgraded to the new Pages and can't find a way to link text boxes anymore like

http://www.macobserver.com/tmo/article/pages-linking-text-boxes

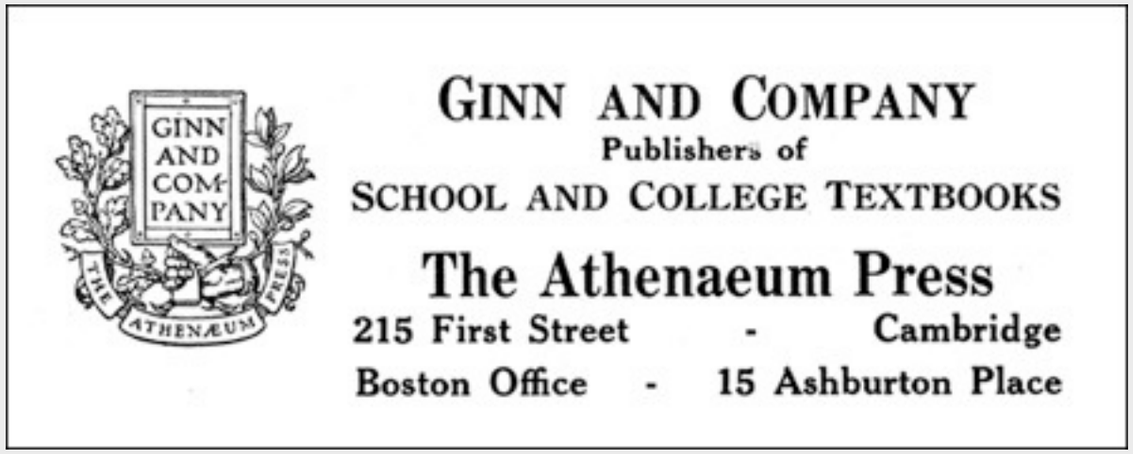
Am I missing something, or is it really not possible anymore?

- Jamon

where are Word's concepts from?

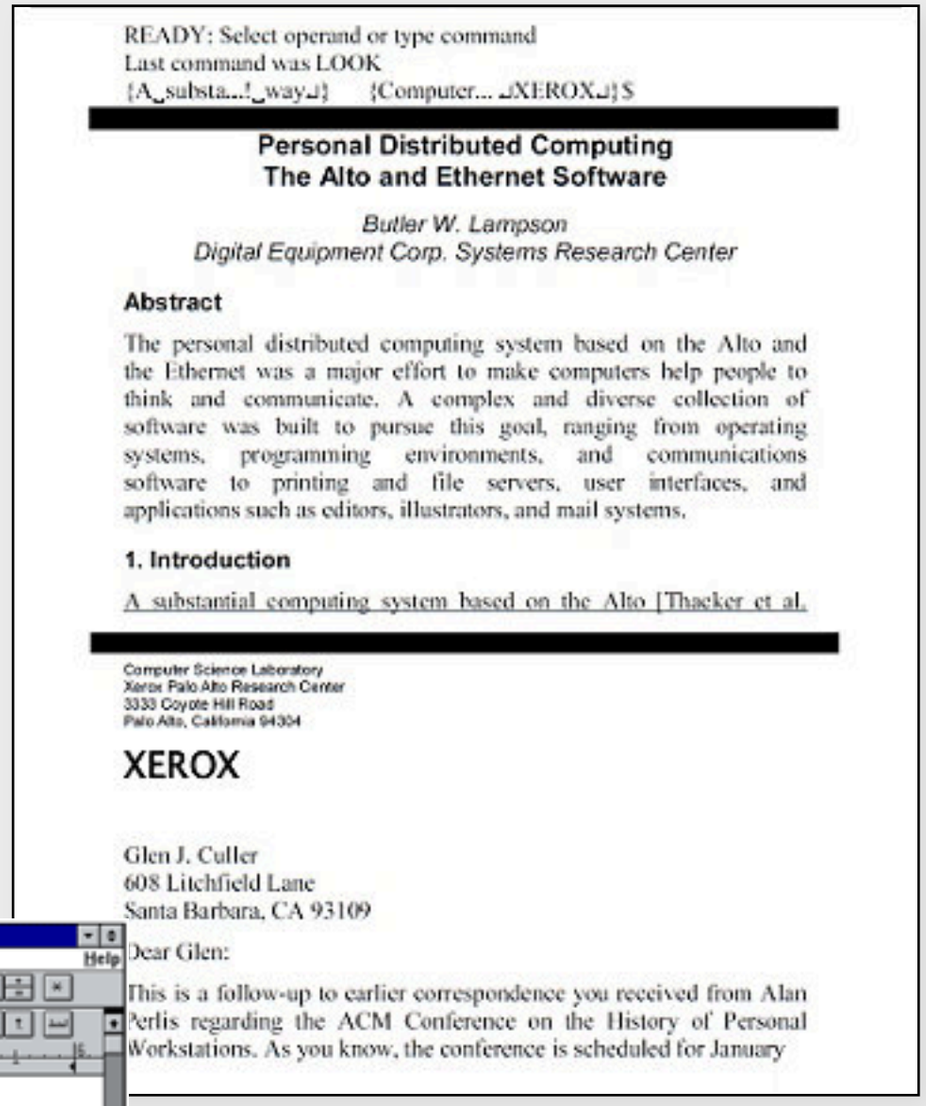
Charles Simonyi: brought key concepts to Word from Xerox PARC

rich concepts have long journeys

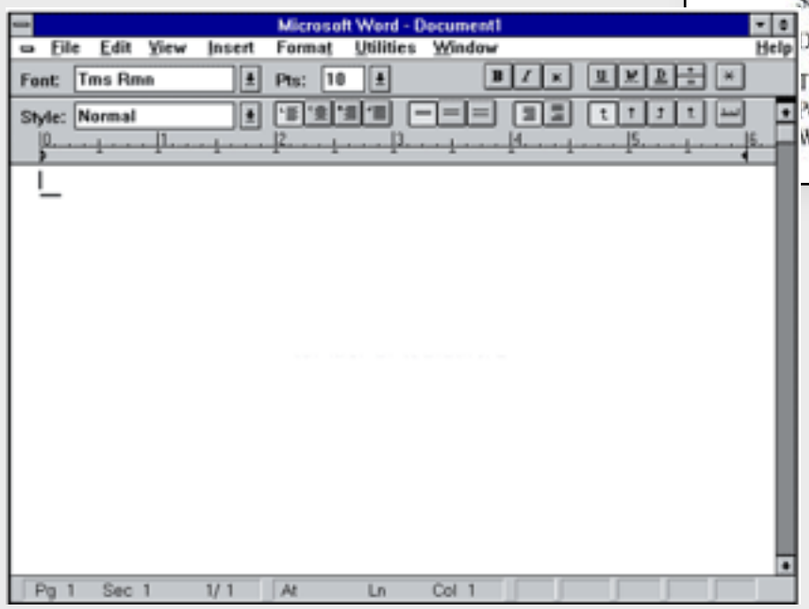


Ginn & Co, since 1868

Apple Pages, 2005



Bravo, 1974



Microsoft Word, 1983

new, old & refurbished concepts

even these were **invented**

repurposed with a new role

often **enablers** of new technology

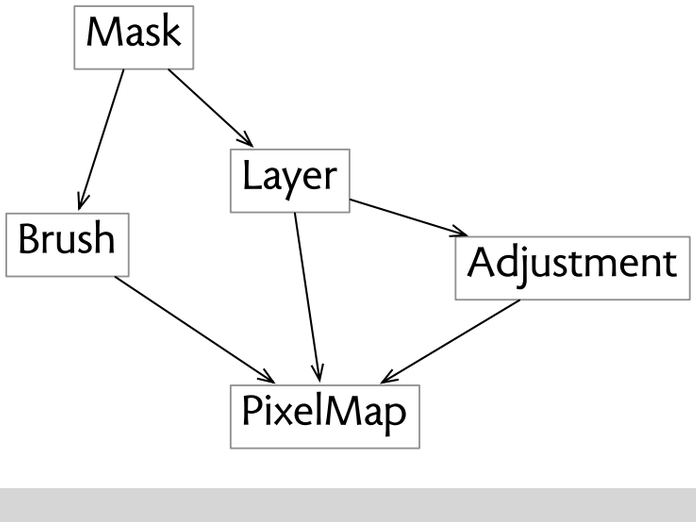
pre-existing concepts	analogic concepts	synthetic concepts: entirely new
electoral vote	comment, tweet	relative reference
purchase order	folder, label	vacation bounce
social security number	layer, mask	hashtag
calendar event	friend, follower	public key

so instead of this...

UI design
soft & human
about presentation

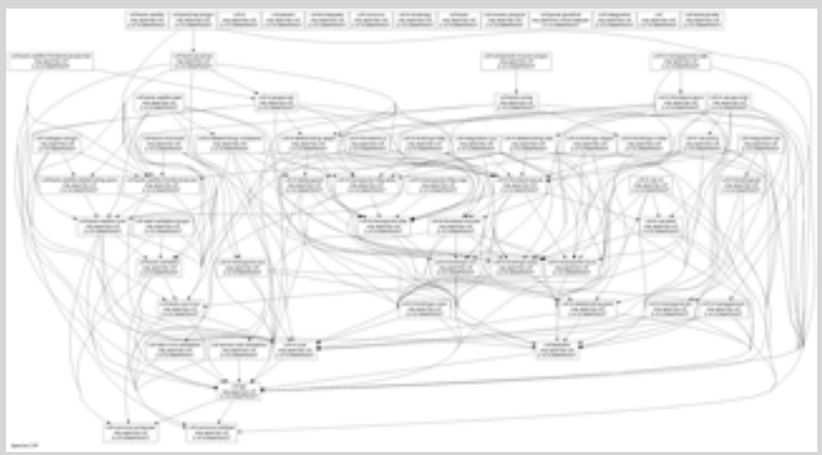
programming
hard & technical
about content

… a new (old) view



conceptual design:
essential concepts
& behavior

Fred Brooks
Essence & accident



representation design:
organization &
performance

it's all about the concepts

Acrobat
text box
object
document text

incoherent concepts, no clear **purpose**

Photoshop
channel
layer
mask

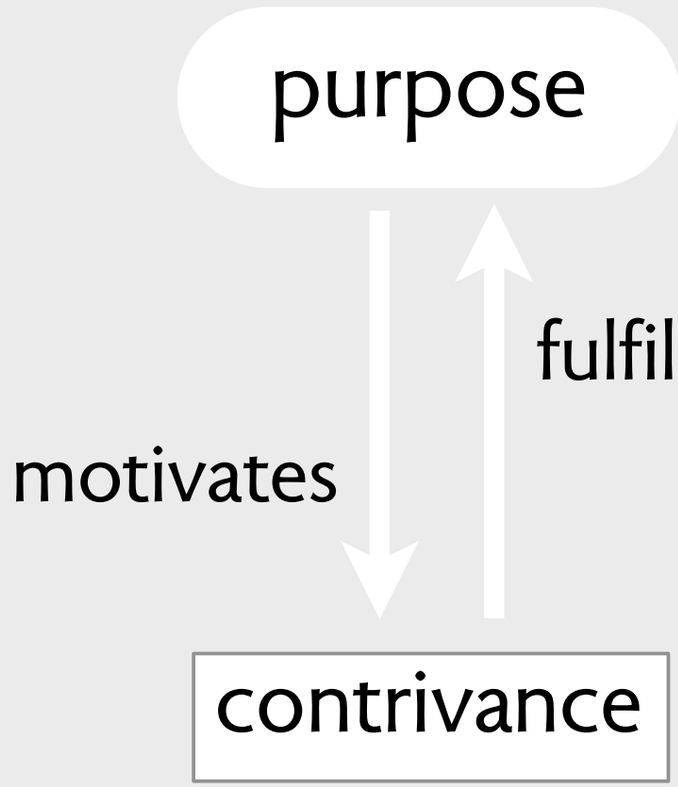
powerful concepts with low level **purposes**

Lightroom
action
treatment
effect

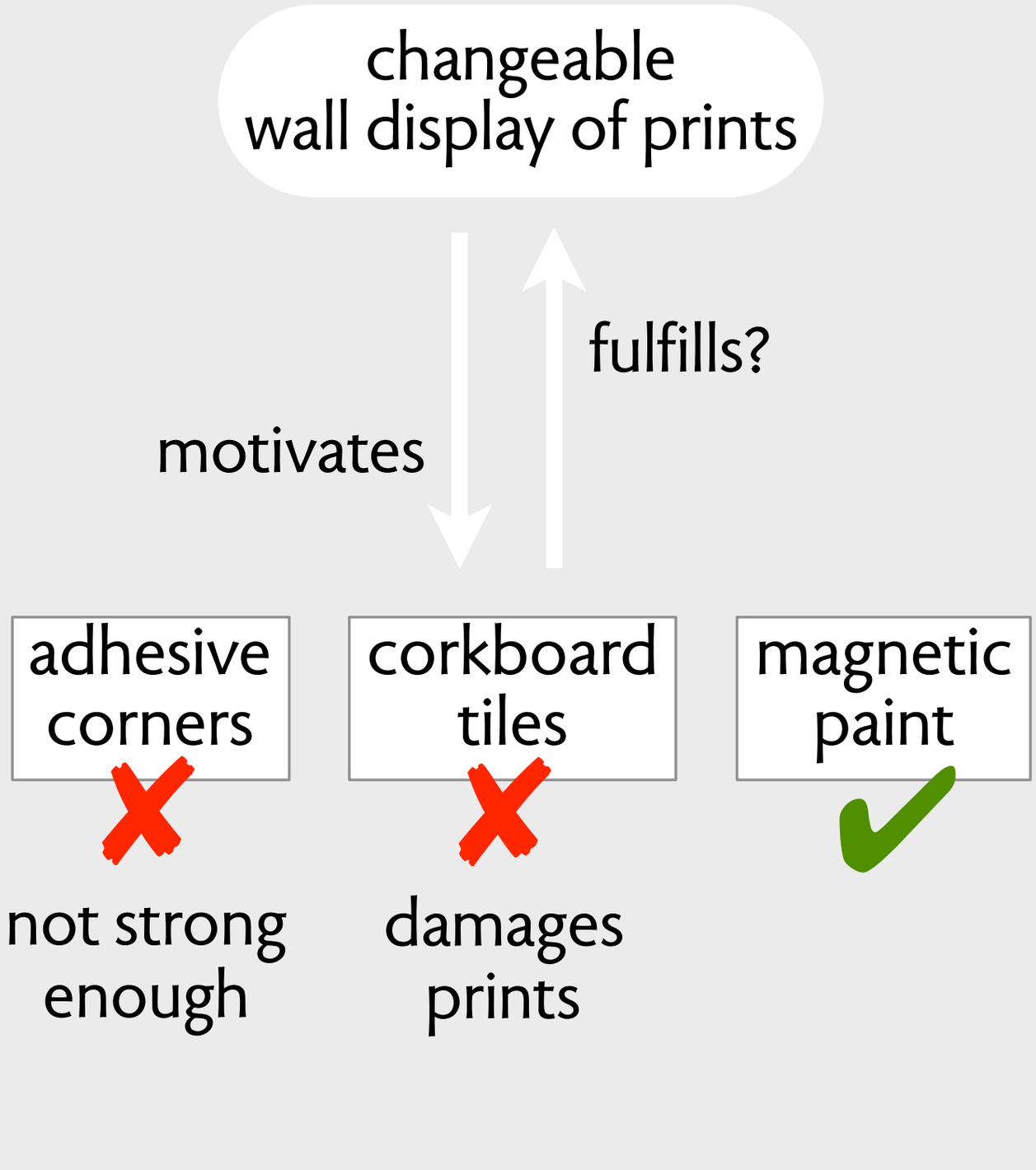
simple concepts with **purposes** aligned to common tasks

purposes

design is driven by purpose



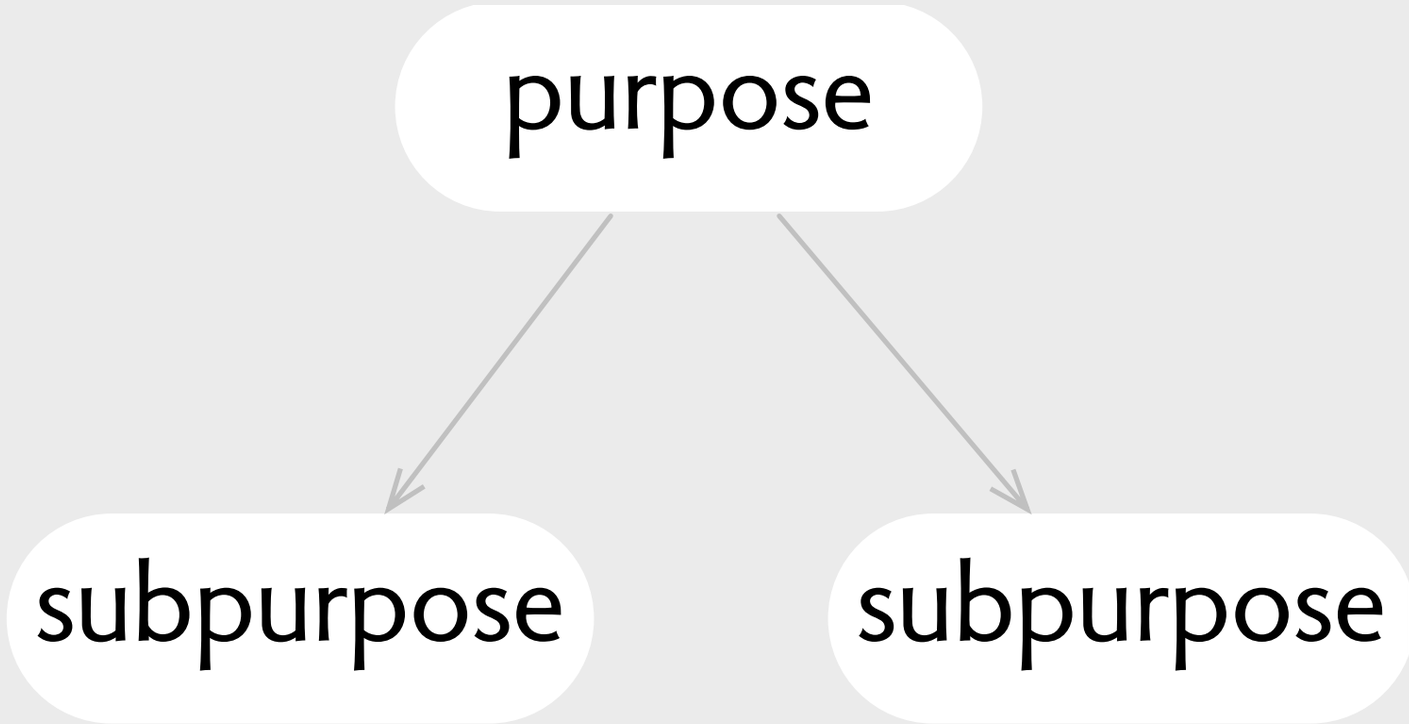
example: a photo wall



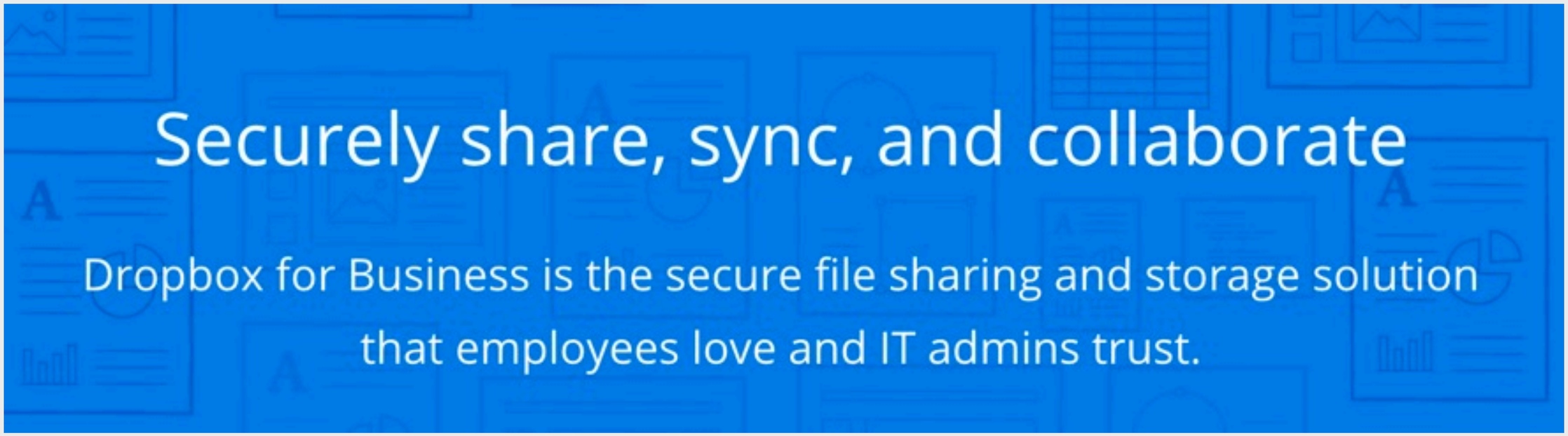
final design, executed

purpose elaboration

in complex systems
purpose is elaborated into subpurposes



dropbox



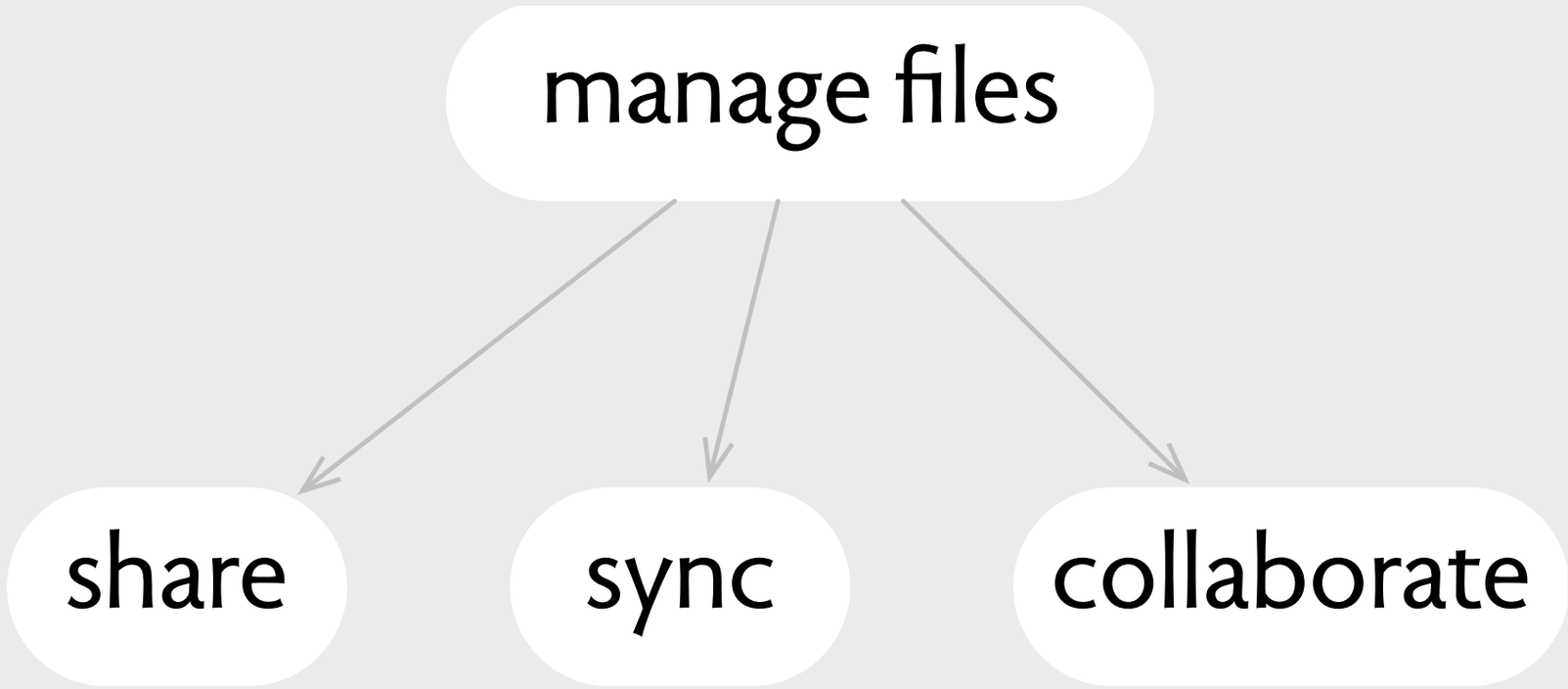
share: control who can read your files

sync: keep files on multiple machines consistent

collaborate: support multi-user editing of documents

store: expand space available by storing files in the cloud

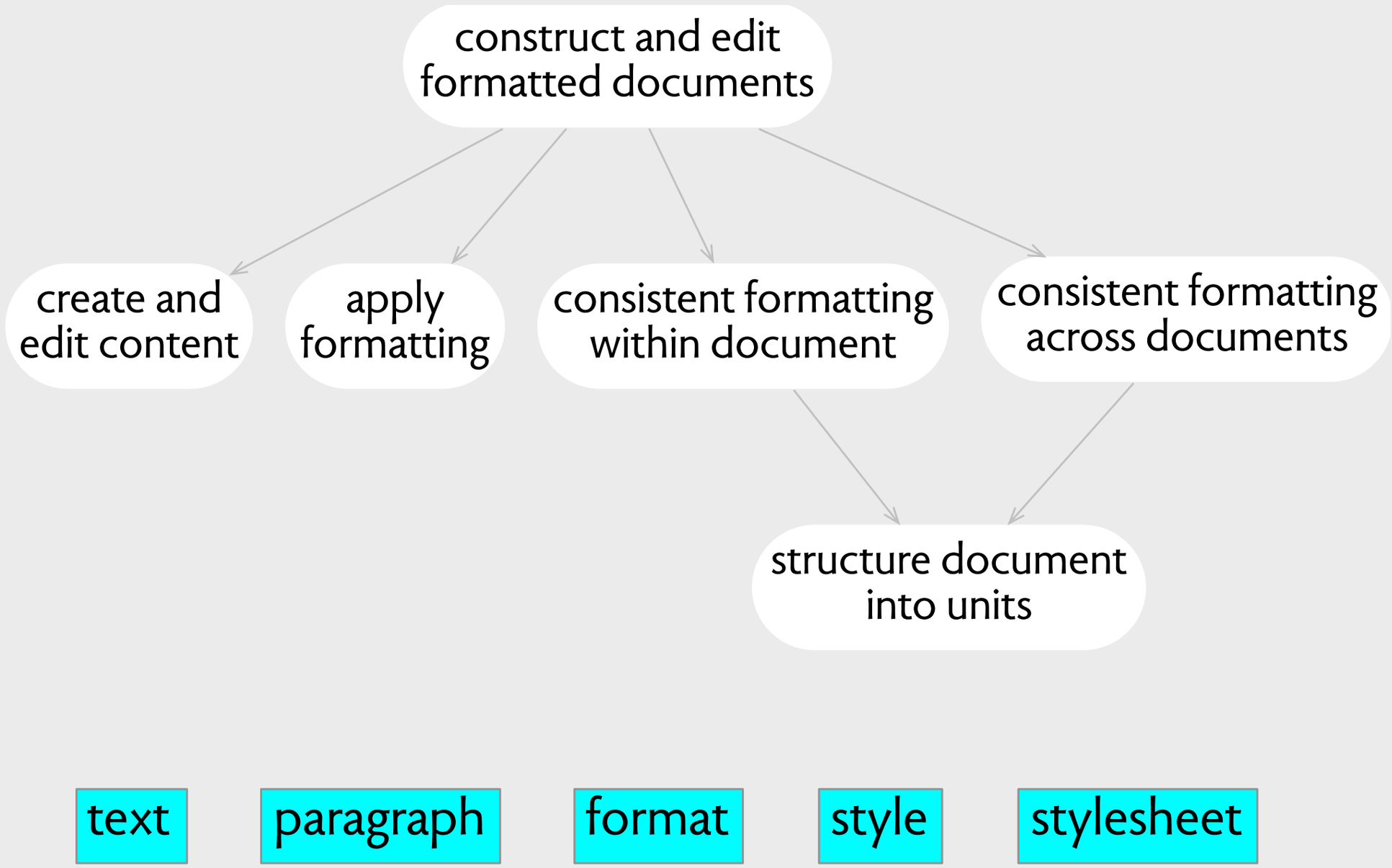
a hierarchy of purposes



the fundamental idea

in a well designed system
each concept is motivated by one purpose

example: word processor



but what exactly is a concept?

a timer

a conceptual explanation

on: **bool**
time: Slot

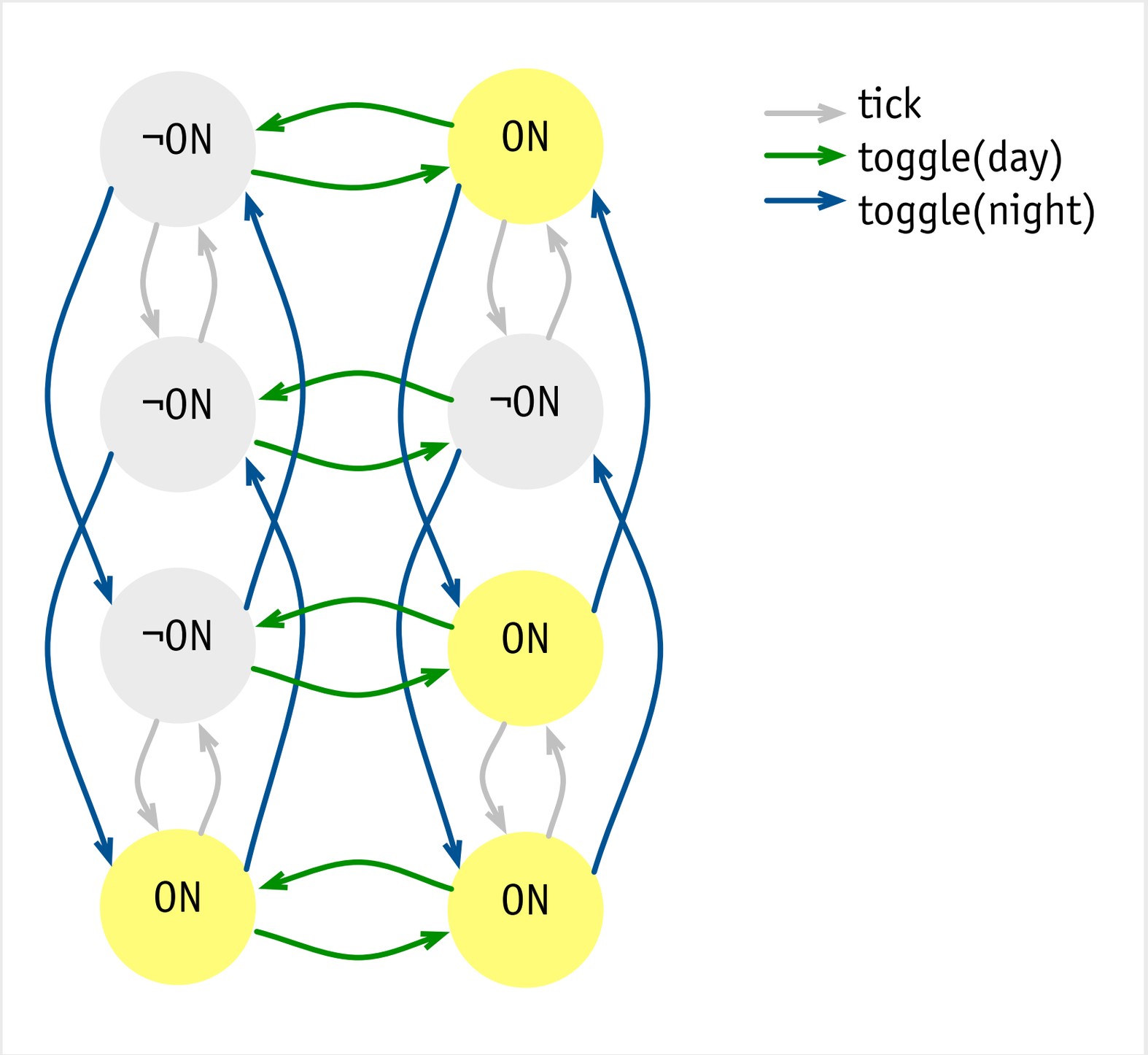
schedule: **set** Slot

inv on = (time ∈ schedule)

tick ≜ time := next(time)

toggle (s: Slot) ≜
    if s ∉ schedule **then** schedule := schedule ∪ {s}
    else schedule := schedule \ {s}

a non-conceptual description



a concept is...

an increment of functionality
can be included independently of others

that fulfills a purpose
contributing to the system's overall purpose

with its own state
visible to the user

with its own actions
performed by the user

affecting the external world
but often only indirectly

formal models of concepts

```
on: bool
time: Slot
schedule: set Slot
inv on = (time ∈ schedule)
tick ≜  time := next(time)
toggle (s: Slot) ≜
    if s ∉ schedule then schedule := schedule ∪ {s}
    else schedule := schedule \ {s}
```

what's good	what's bad
every behavior (helps get it all right)	every behavior (irrelevant ones too)
just what, not why (separation of concerns)	just what, not why (no real meaning)

the operational principle

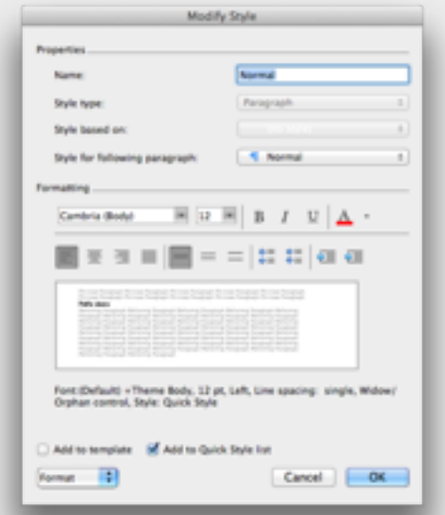
a better way to define & explain a concept

an archetypal scenario
separates essential from accidental aspects
shows how purpose is fulfilled
by combination of user & system actions

Michael Polanyi

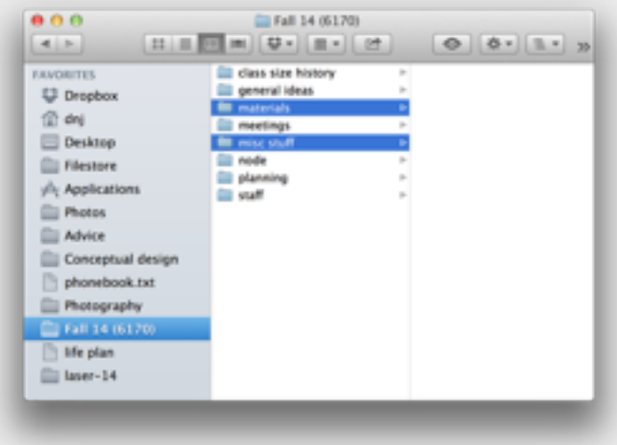
"**if** you pull a tab out, **then** when that time slot comes around, the light will go on"

"**if** you pull a tab out, **then** when that time slot comes around, the light will go on"



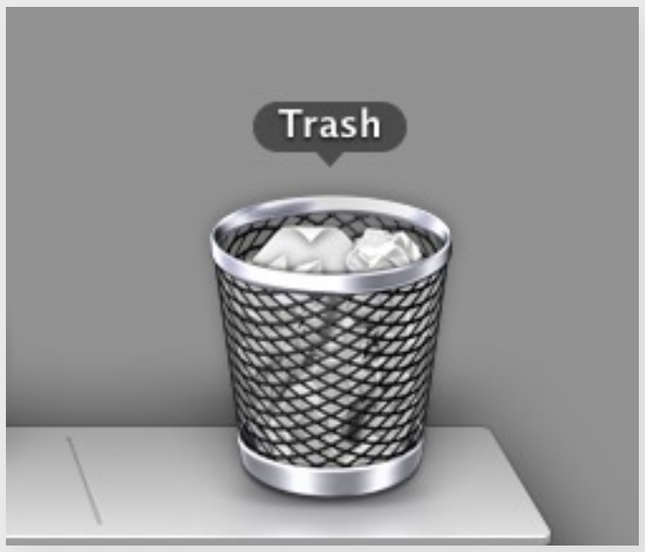
"**if** you change a style's format, **then** all paragraphs of that style will change format accordingly"

"**if** you tag a photo, **then** all friends of the person tagged will be able to see the photo"



"**if** you select a file and it belongs to a folder with keyboard focus, **then** pressing delete will move the file to the trash"

purposes, principles & misfits



concept: trash

purpose: allow undo of deletions

operational principle: if you delete a file, it moves to a special folder; you can restore from there, but emptying it removes contents for good

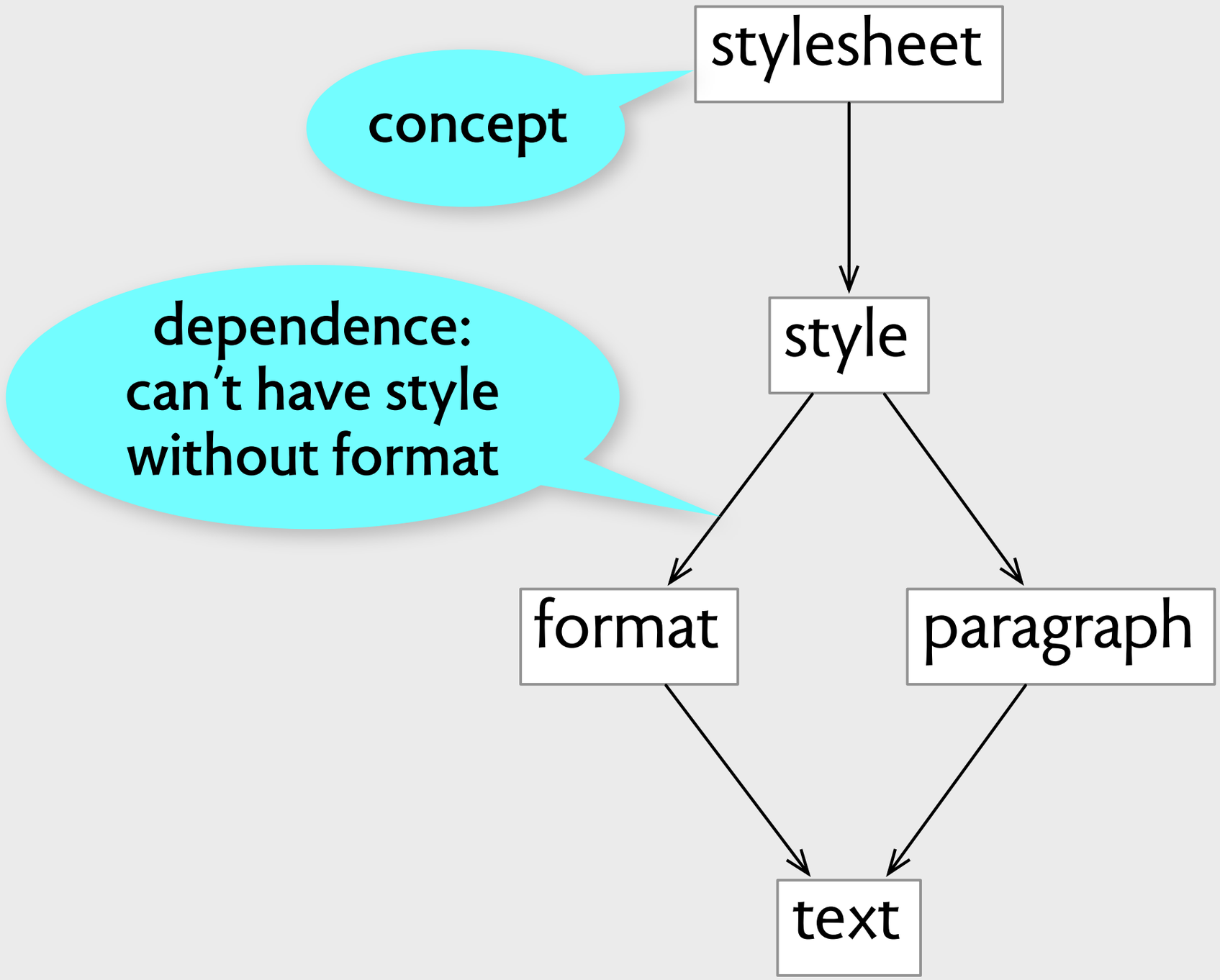
operational misfit: if you delete a file on an external drive, you cannot reclaim the space until you empty the trash, but then you'll lose the ability to restore files deleted from the main drive

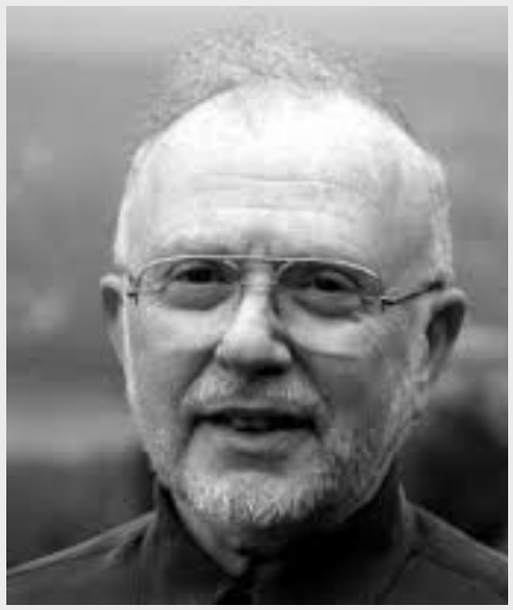
operational misfit: if you delete an old file and change your mind, you may not be able to find it again in the trash (if there are many deleted files and you forgot the file's name)

design structure

concept dependences

$\langle c,c' \rangle \in \text{depends} \Leftrightarrow \forall a: \text{apps} \cdot c \in \text{concepts}(a) \Rightarrow c' \in \text{concepts}(a)$

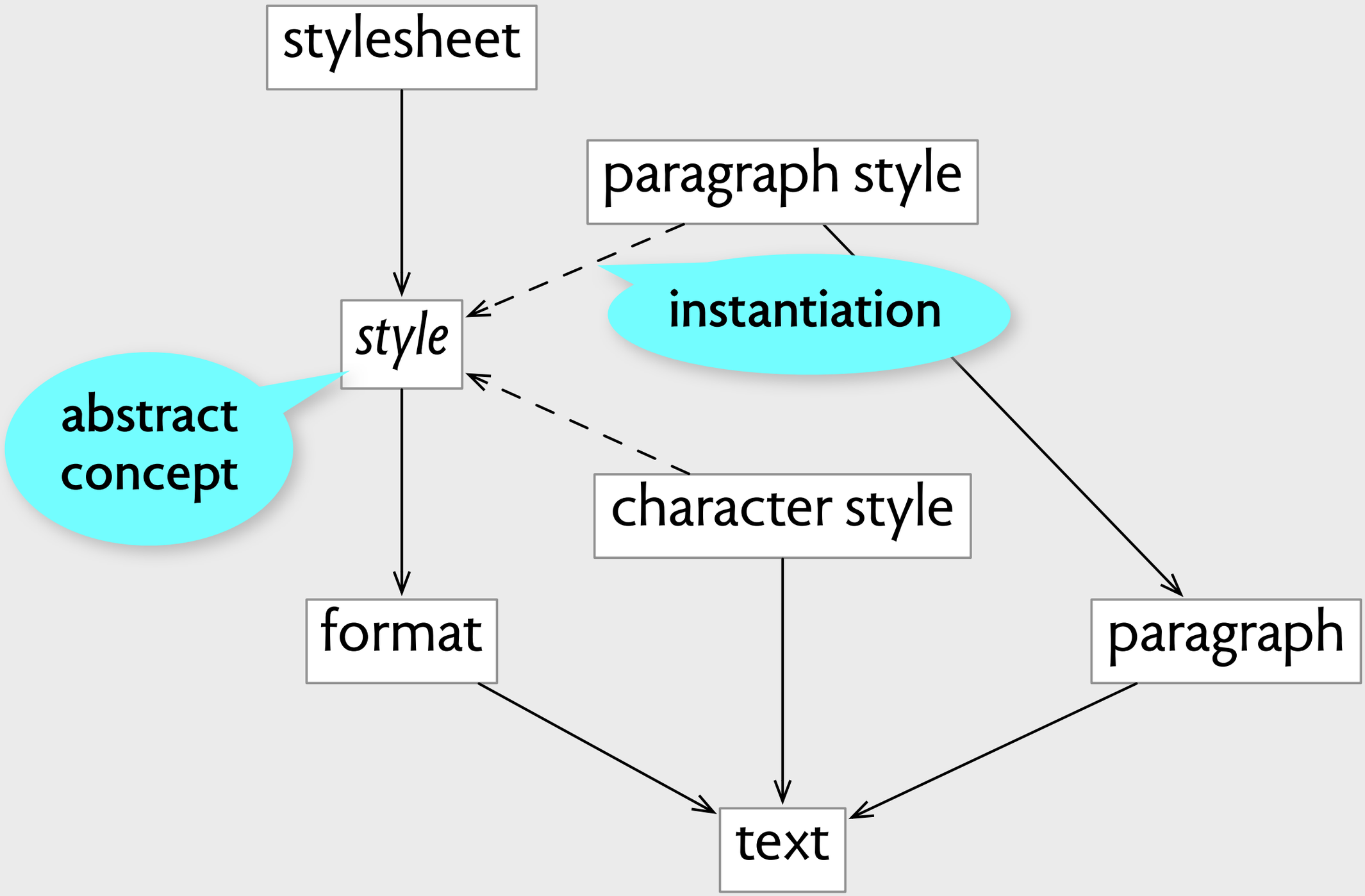




David Parnas
uses relation

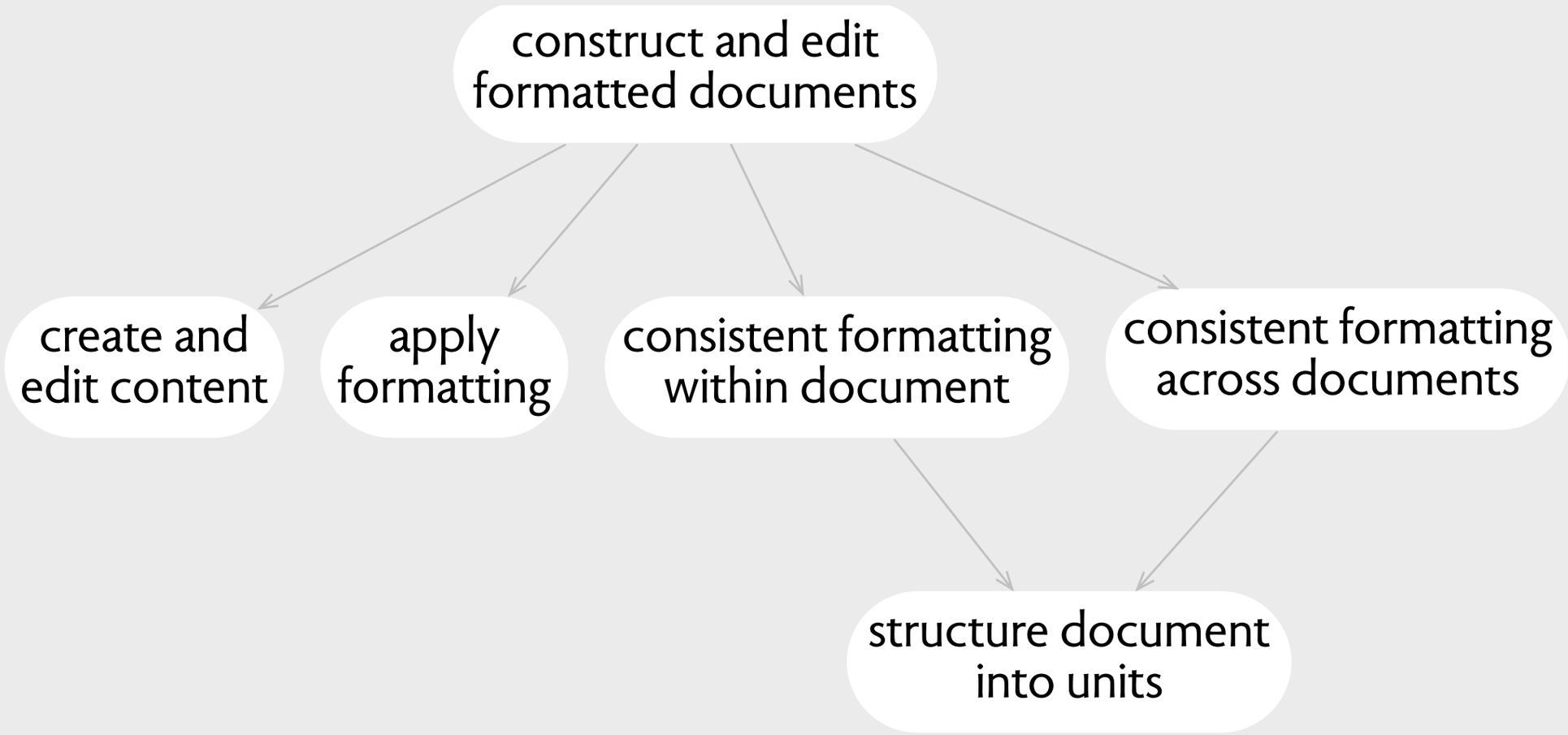
abstract concepts

$\langle c_i, c \rangle \in \text{instantiates} \Leftrightarrow (\forall a: \text{apps} \cdot c \in \text{concepts}(a) \Rightarrow \exists i \cdot c_i \in \text{concepts}(a))$



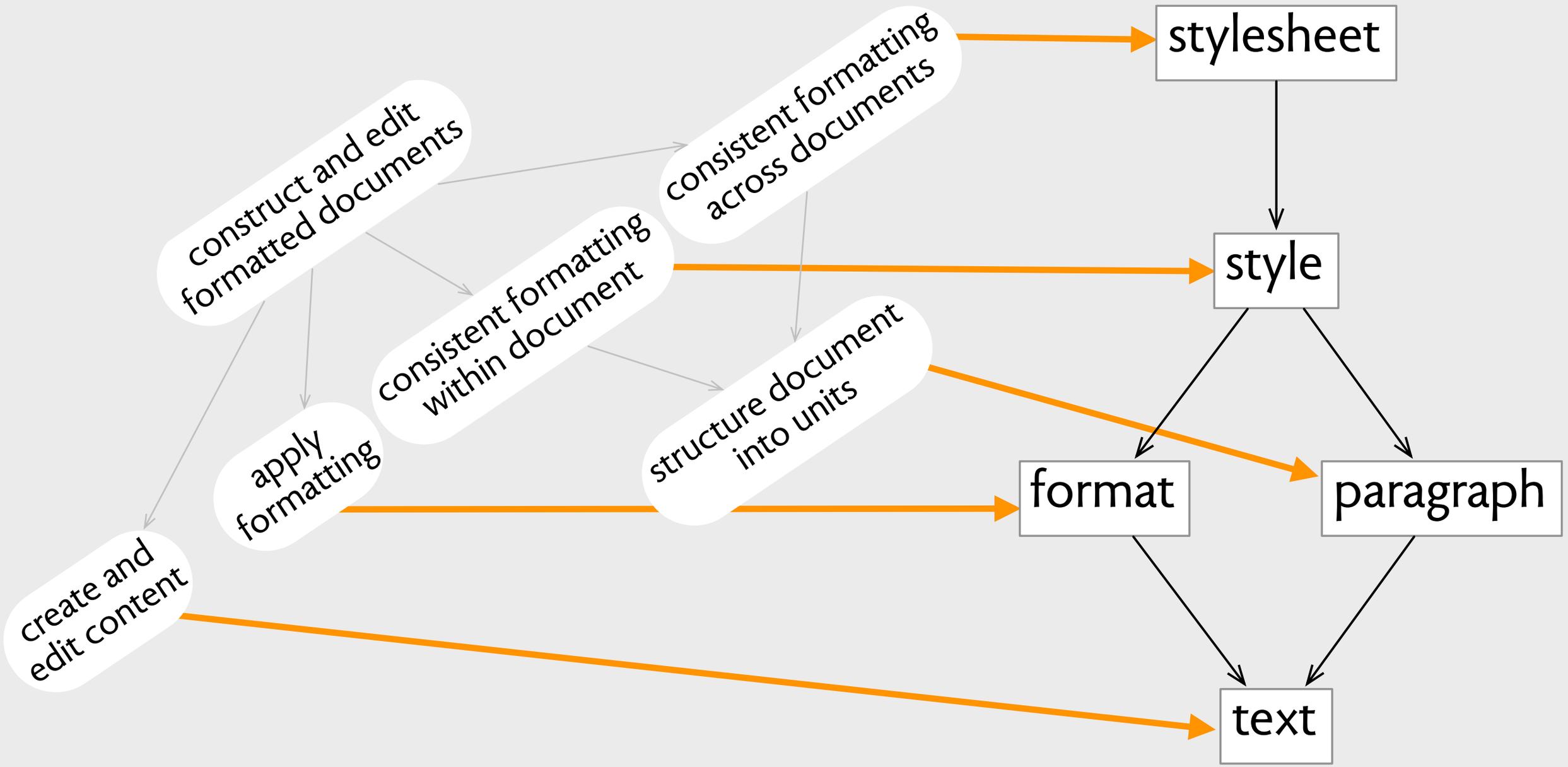
purpose hierarchy

$$\langle p,p' \rangle \in \text{requires} \Leftrightarrow \forall a\text{: apps} \cdot p \in \text{fulfills}(a) \Rightarrow p' \in \text{fulfills}(a)$$



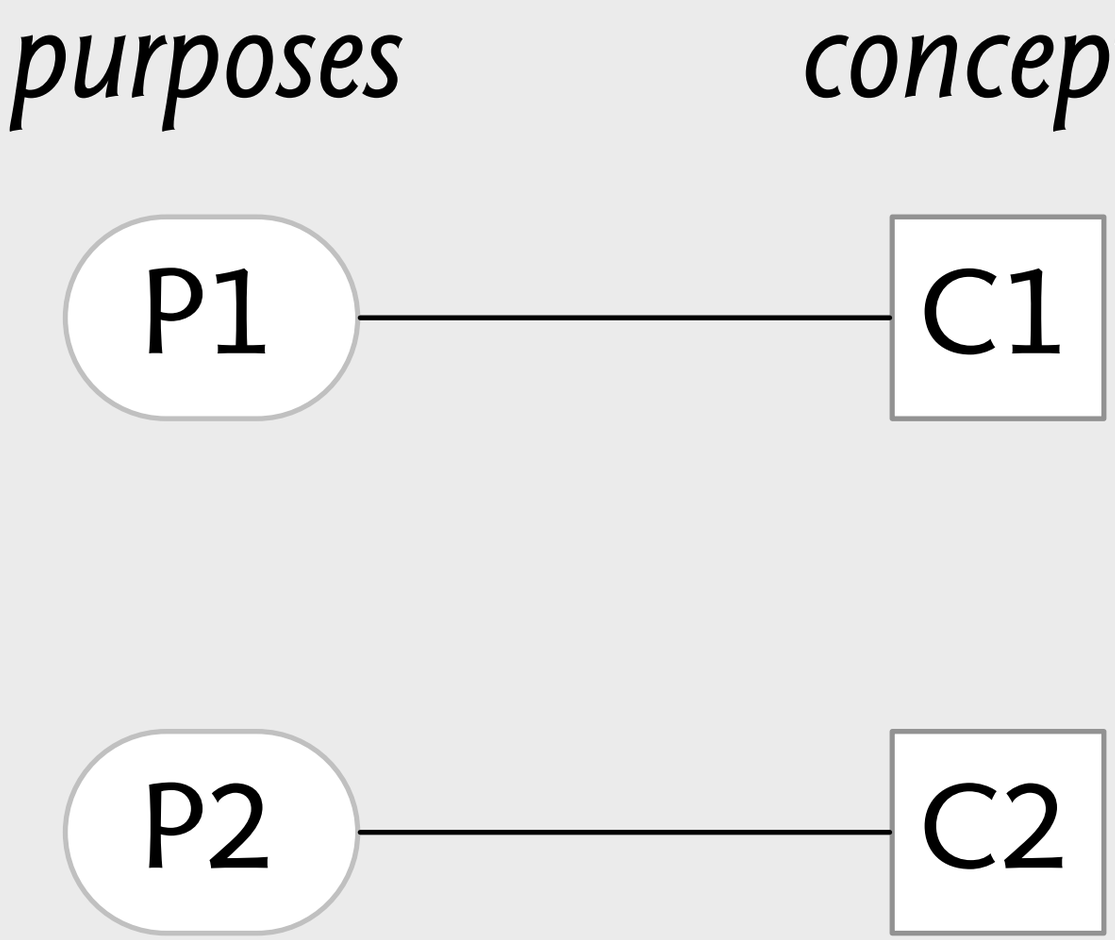
purpose-concept mapping

$\langle p,c \rangle \in \text{motivates} \Leftrightarrow \forall a: \text{apps} \cdot p \in \text{fulfills}(a) \Rightarrow c \in \text{concepts}(a)$



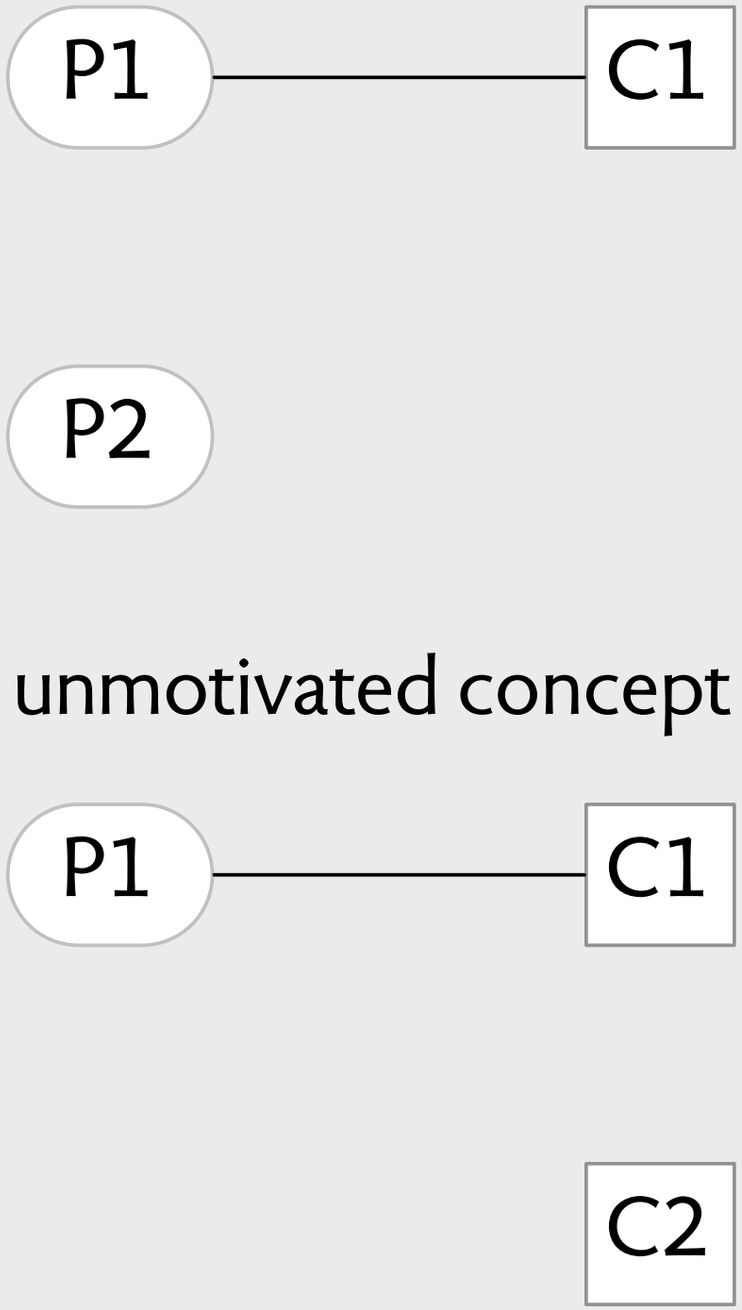
analyzing designs

the ideal mapping

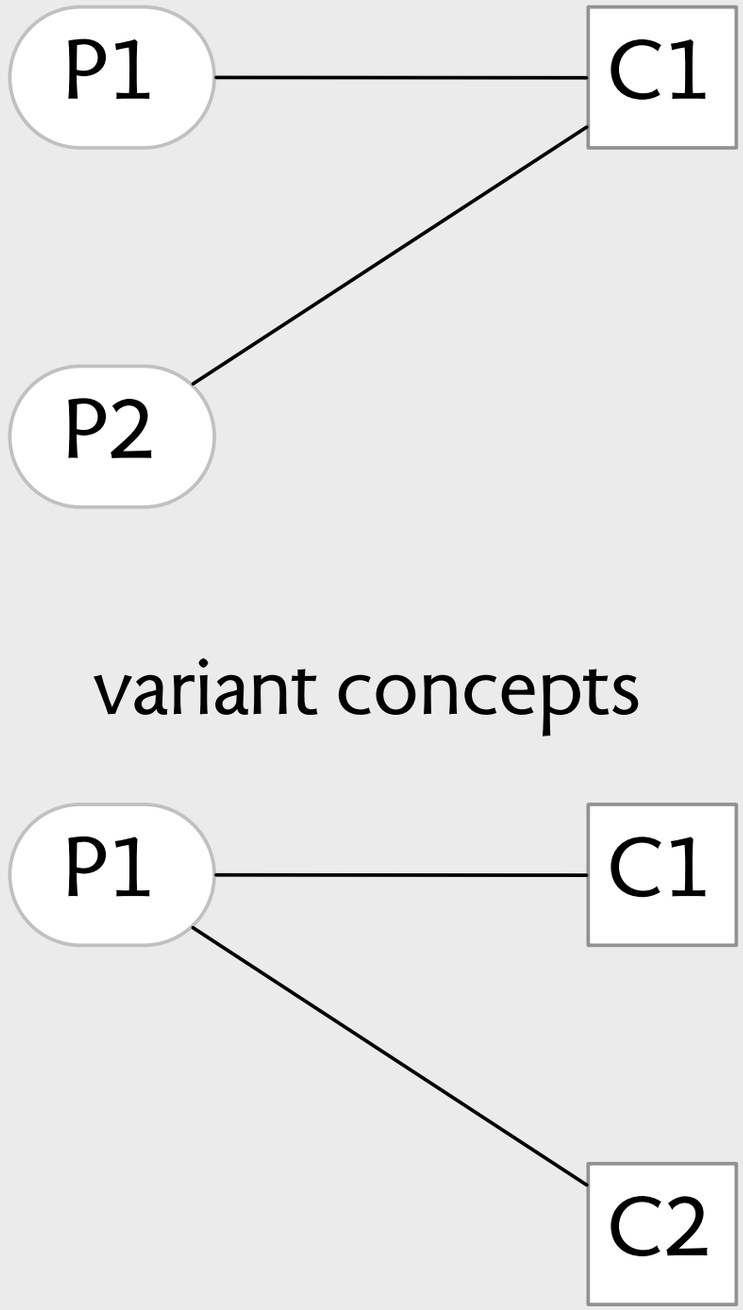


4 bad smells

unfulfilled purpose



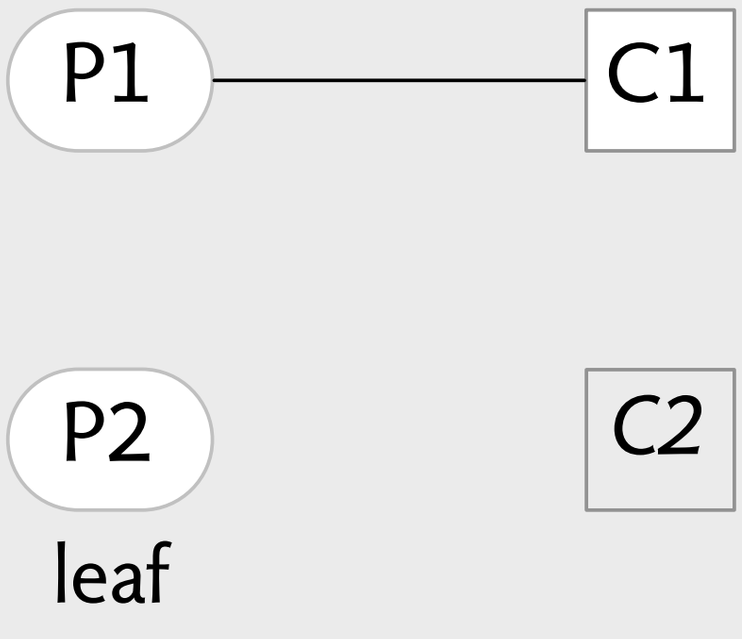
overloaded concept



unmotivated concept

variant concepts

unfulfilled purposes



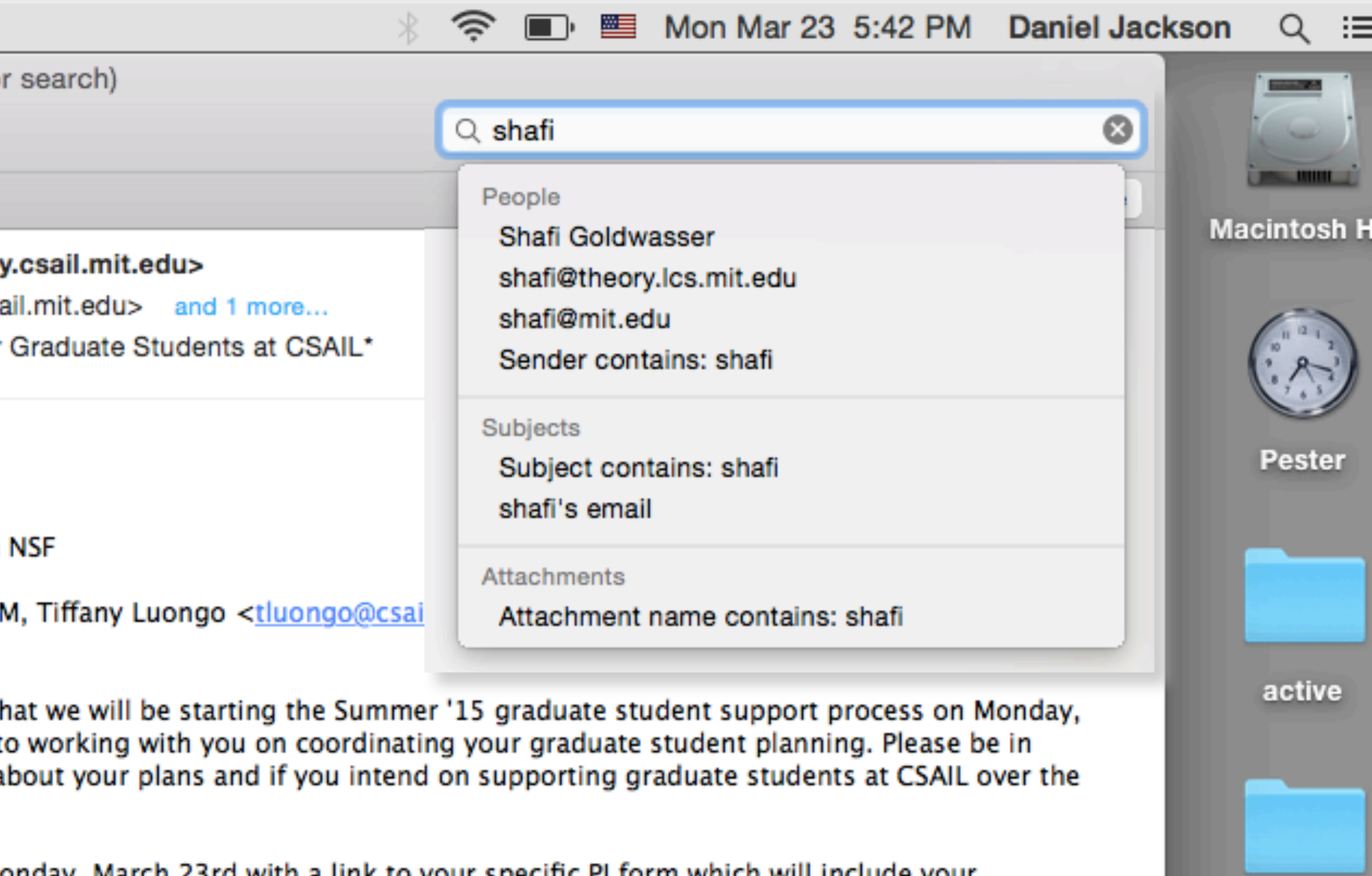
user (Apple Mail, Gmail)
› ‘identify parties to communication’
› weak search, no authentication

slide hierarchy (Powerpoint)
› ‘structure slides in a tree’
› sections provide just one level

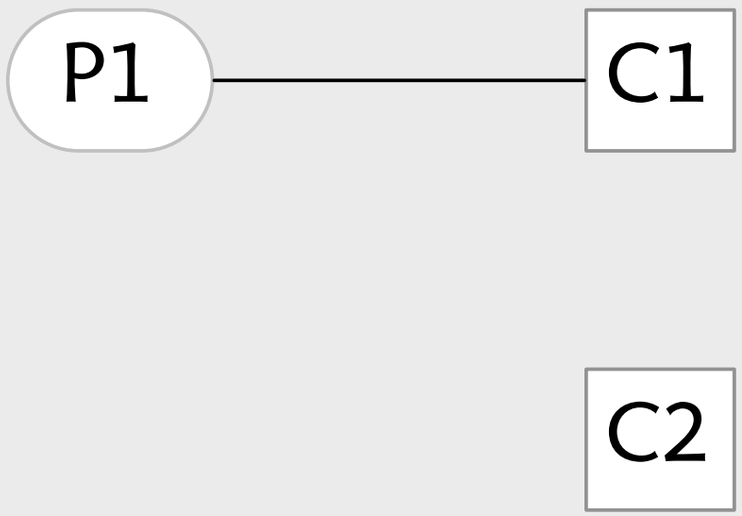
aspect ratio (Sony A7Rii, Canon 5D3)
› ‘take square image’
› can’t view in finder or save setting

binder (Preview, Acrobat)
› ‘maintain composite PDF doc’
› can insert pages, but forgets source

unfulfilled purpose Apple Mail



unmotivated concepts



performance

buffer (emacs)

› no reason not to save to file

kludge

stash (Git)

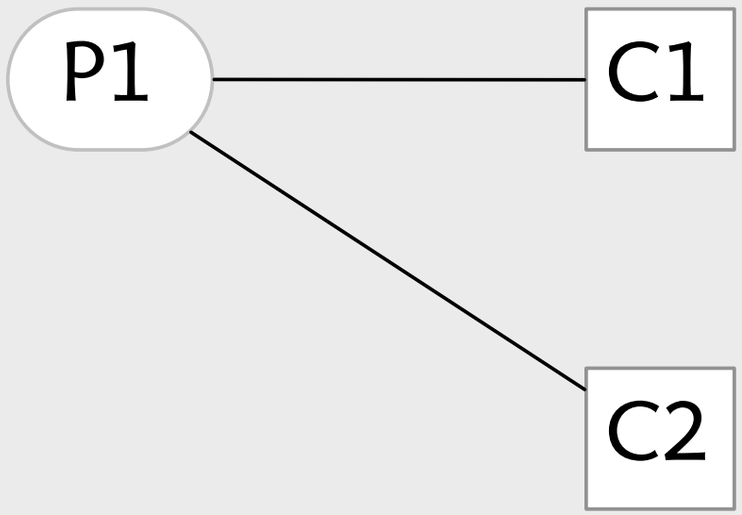
› addresses misfit in branching

glue records (DNS)

› addresses misfit of circular deps

kludge

variant concepts



rating stars (Lightroom)
› colors, flags, stars, oh my!

rules & **searches** (Apple Mail)
› two ways to specify set of messages

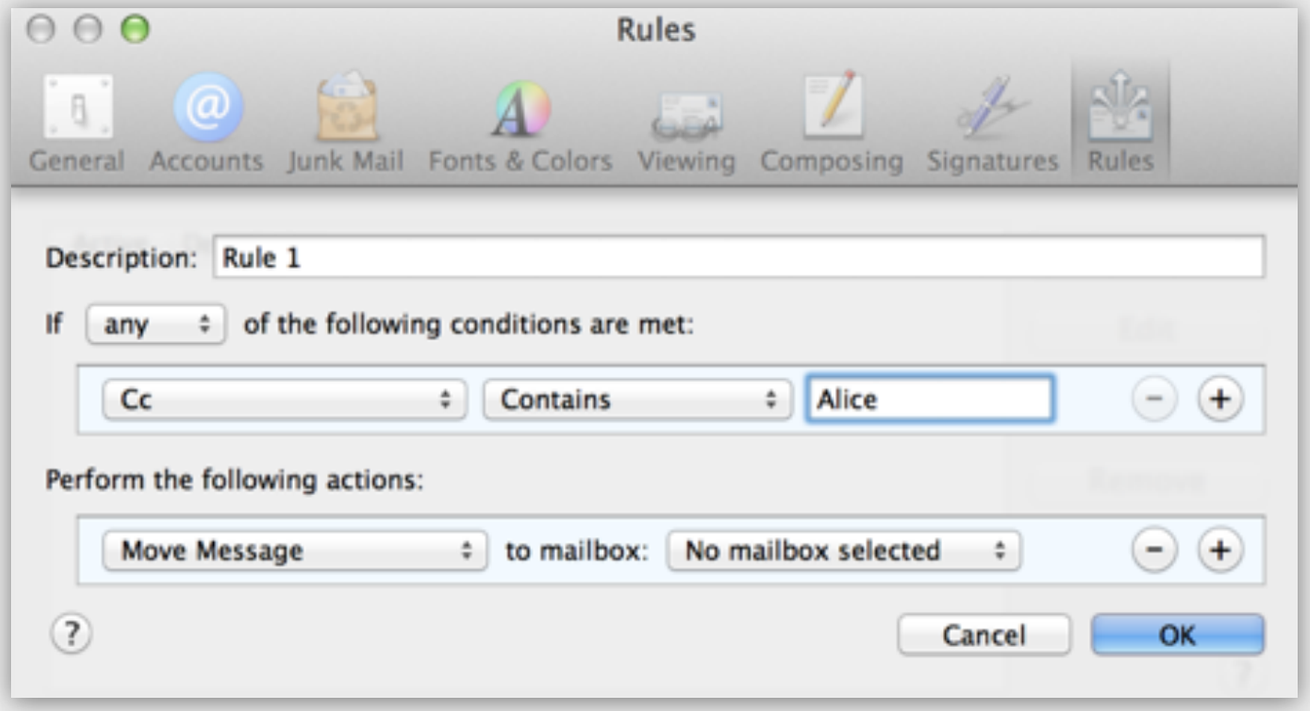
labels & **categories** (Gmail)
› two ways to classify messages

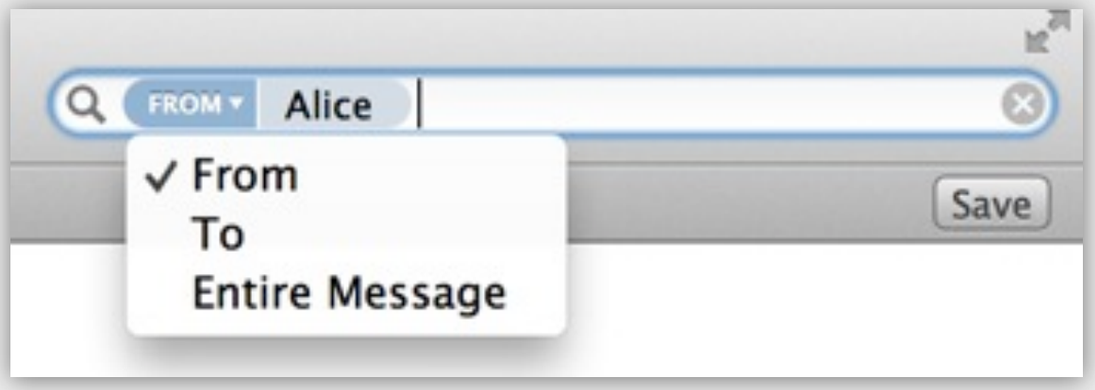
text object & **text box** (Acrobat 10)
› document text too: all different

permissions (AFS)
› coexist with Unix permissions

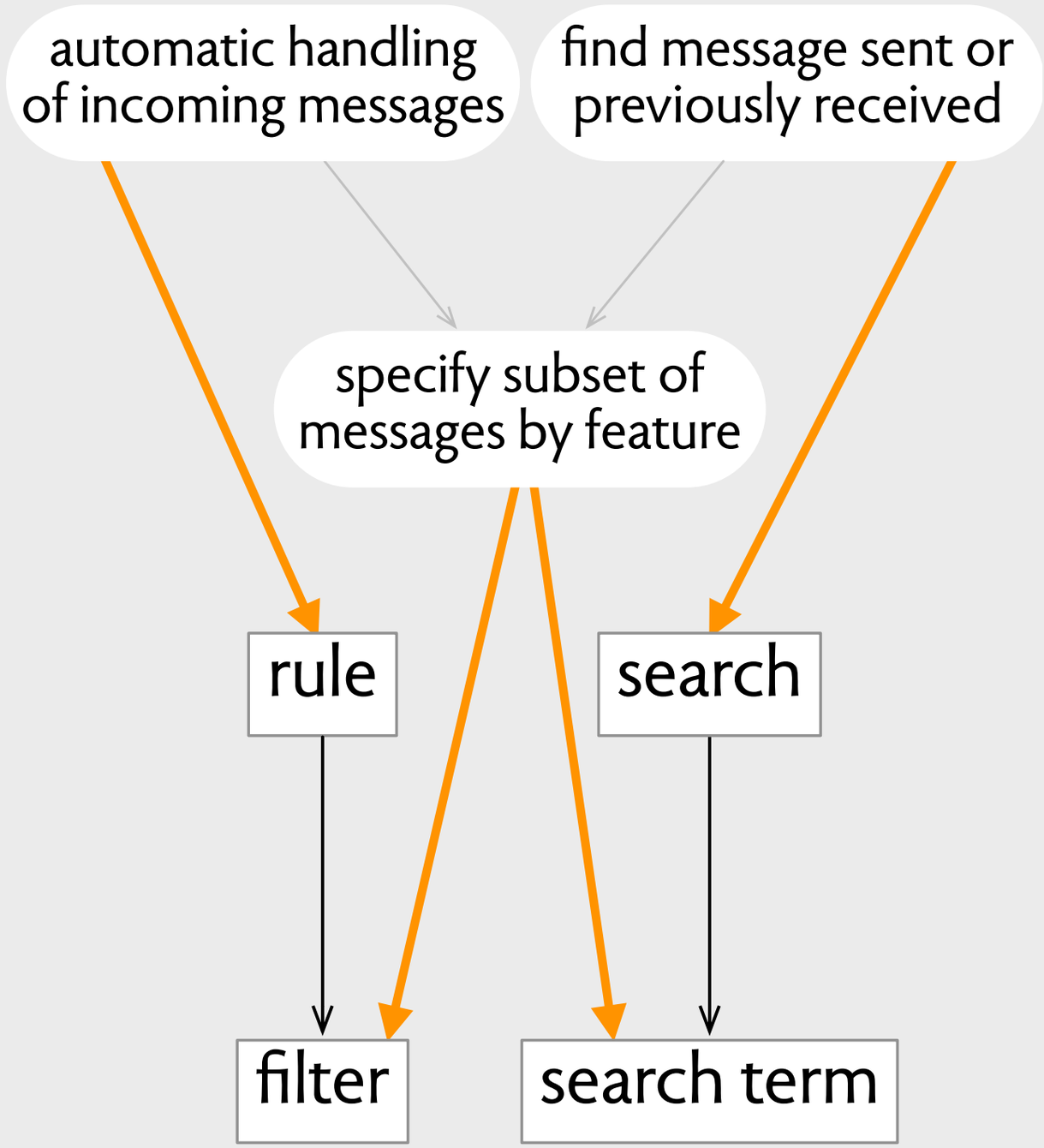
headline, title, caption (IPTC)
› original purposes lost

variant concepts apple mail



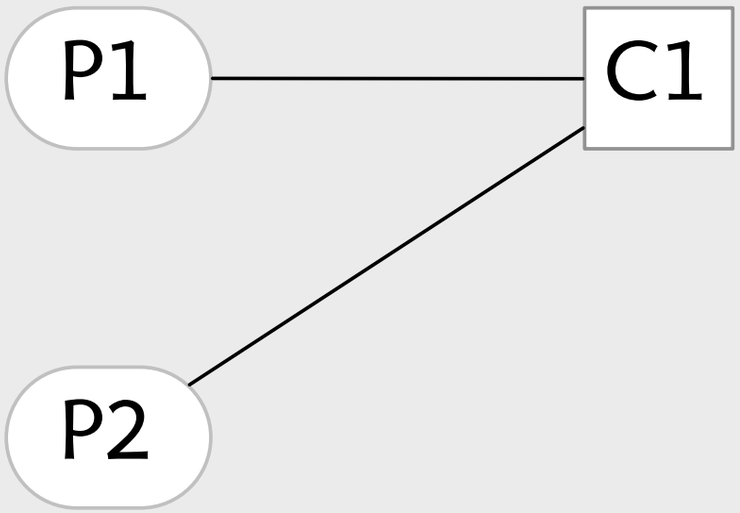


variant concepts for subpurposes



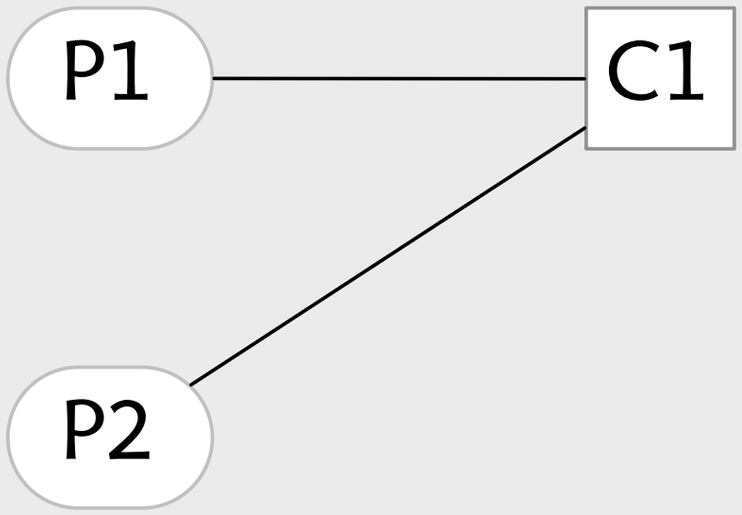
overloaded concepts

No one can serve two masters. Either you will hate the one and love the other, or you will be devoted to the one and despise the other. [Matthew 6:24]



Nam Suh
independence axiom

overloaded concepts



Pamela Zave:
Secrets of CF

conference review
› feedback vs. selection

call forwarding
› follow-me vs. delegate

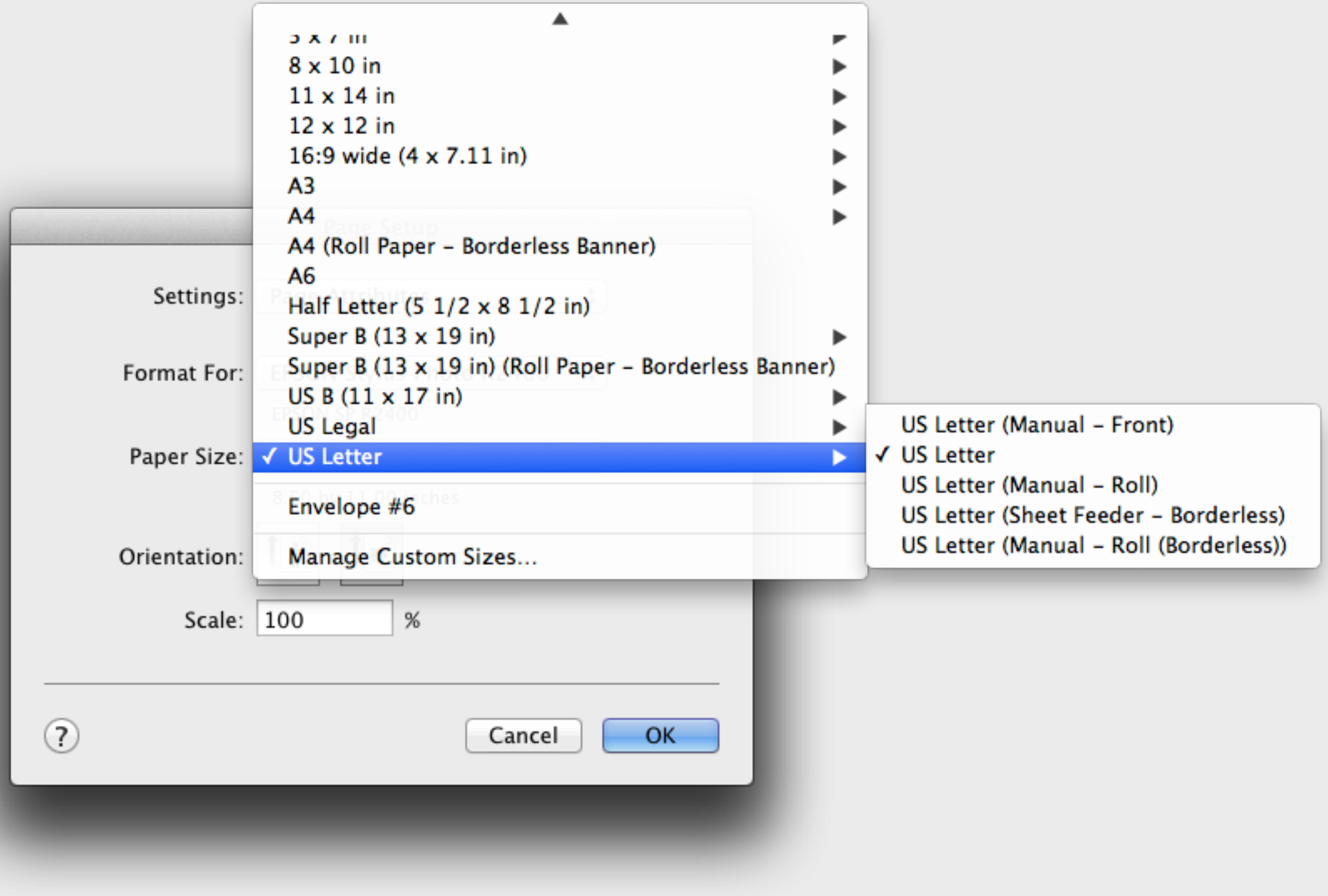
contact (Apple address book)
› shortcut vs. format addressee

friends (Facebook)
› filter posts vs. limit access

signature (Acrobat 9)
› digital vs. physical

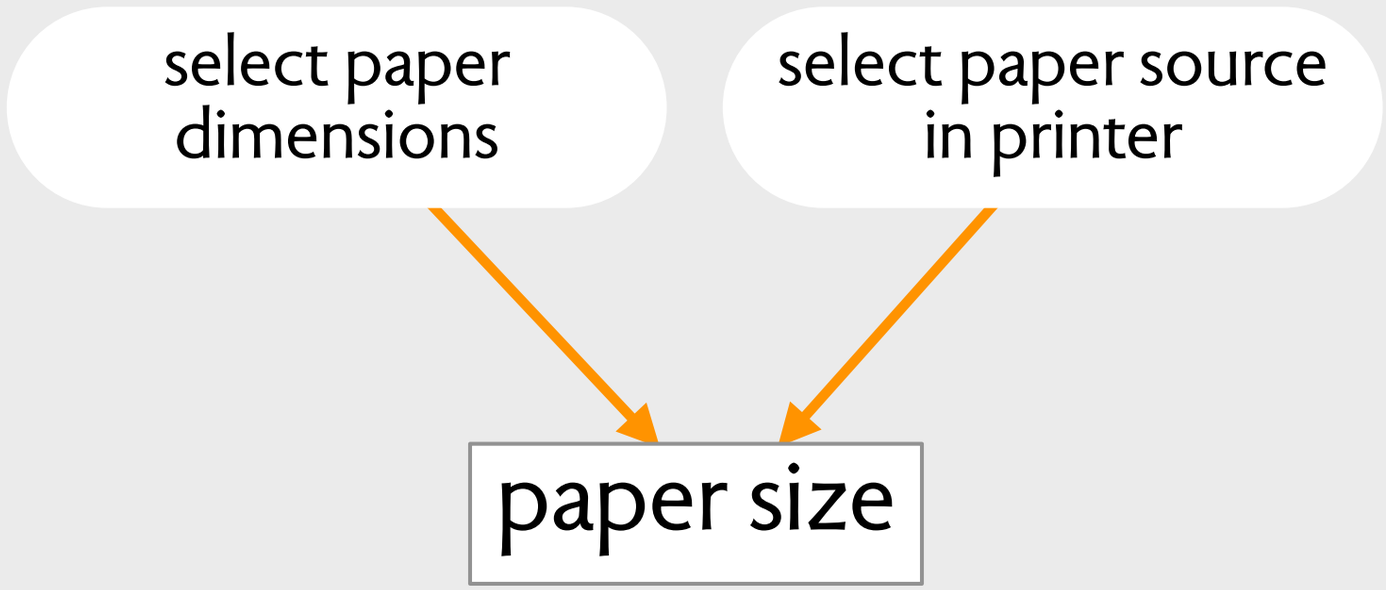
paper size (Epson printer driver)
› dimensions vs. source

overloaded concepts epson driver

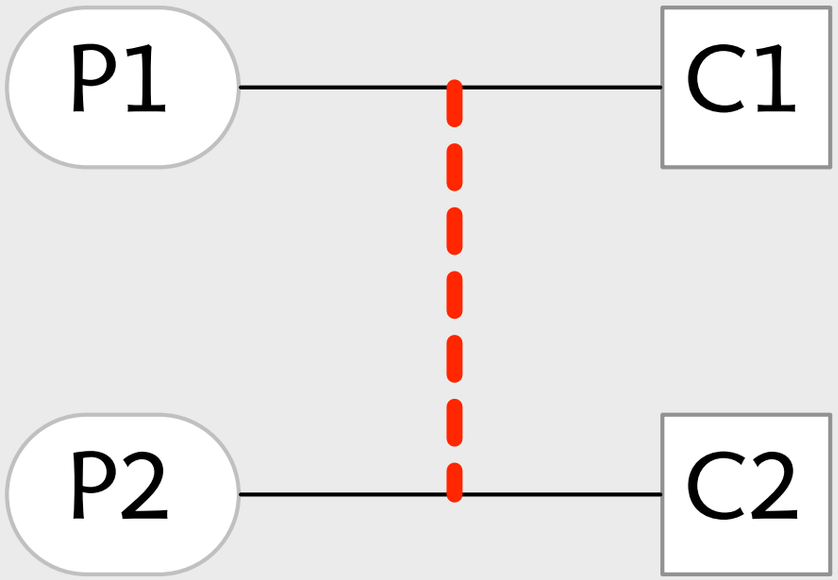


result: can’t create custom size for front loading

overloaded concepts epson driver

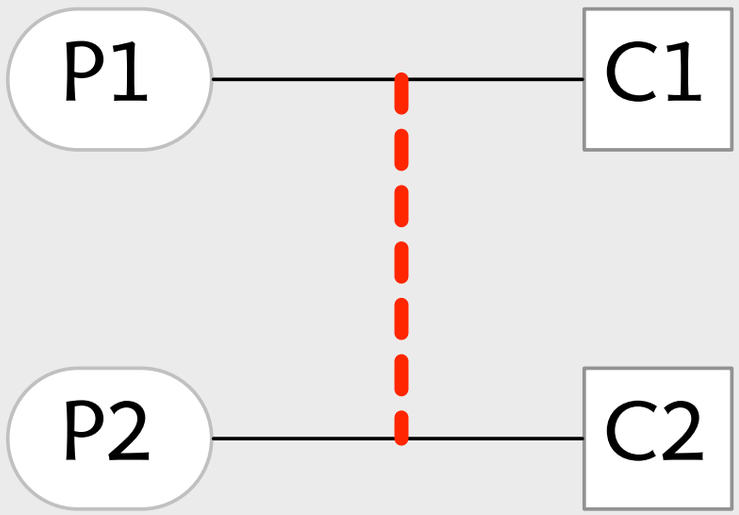


orthogonality



orthogonality is violated when one concept's fitness for purpose is undermined by another concept

non-orthogonal concepts



Shriram Krishnamurthi
BCC example

origin, space, exclusion (CSS)
› 4 position values for 2×3×2 options

conversation & label (Gmail)
› same subject, get same label

listserv & bcc (SMTP)
› modified subject reveals target

title & reply (Tumblr)
› adding ? to title enables replies (!)

group & selection (many old apps)
› can't select object in a group

group & connector (Keynote 5.3)
› can't select box if connected

non-orthogonality fuji x100s

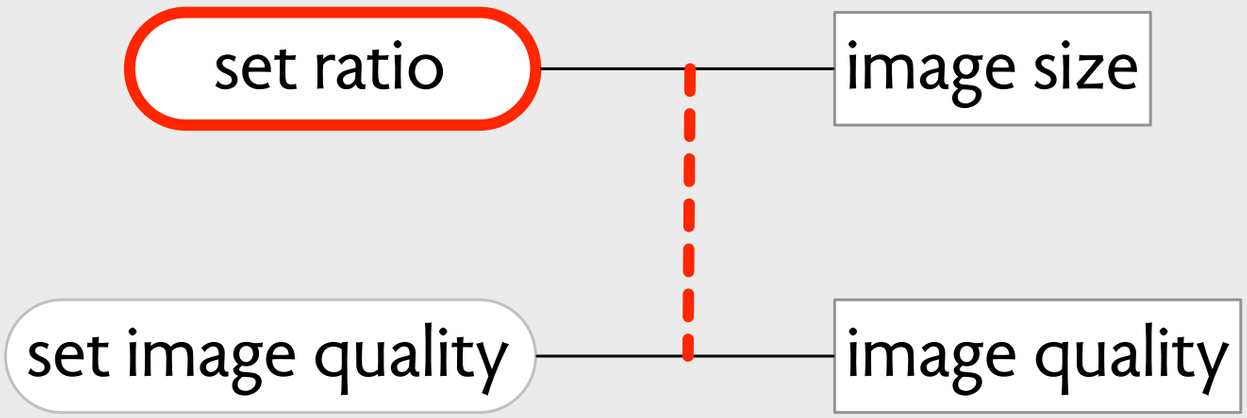
image quality setting

aspect ratio

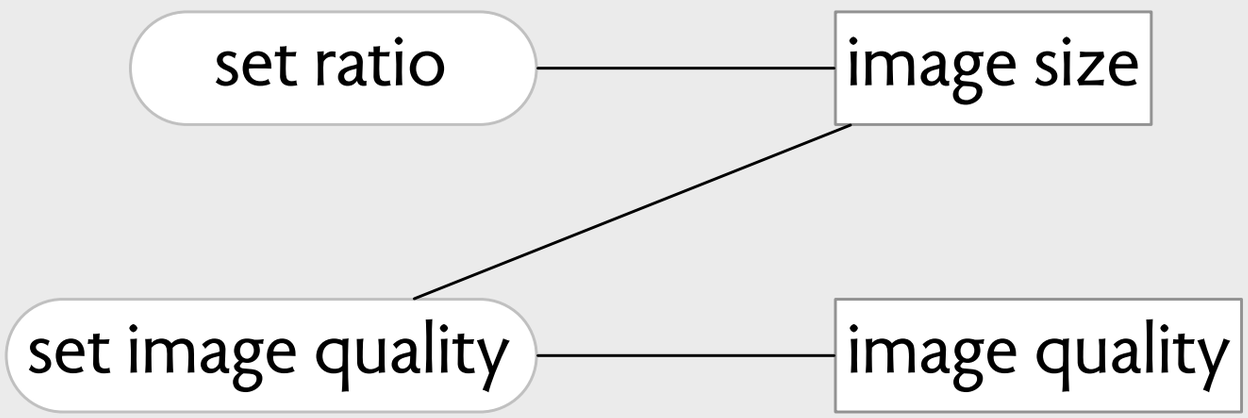
“image size” setting

non-standard ratio + RAW?

image quality undermines image size

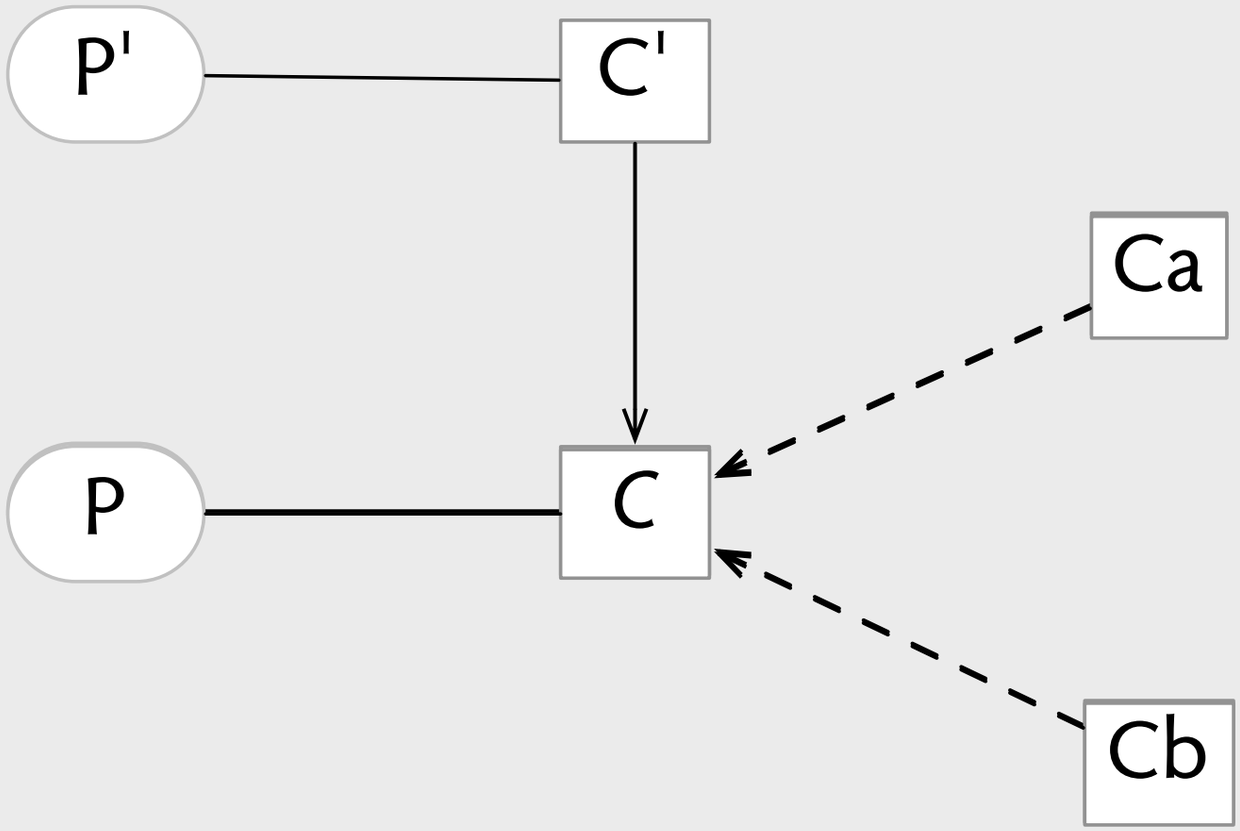


an orthogonality violation



overloaded concept too

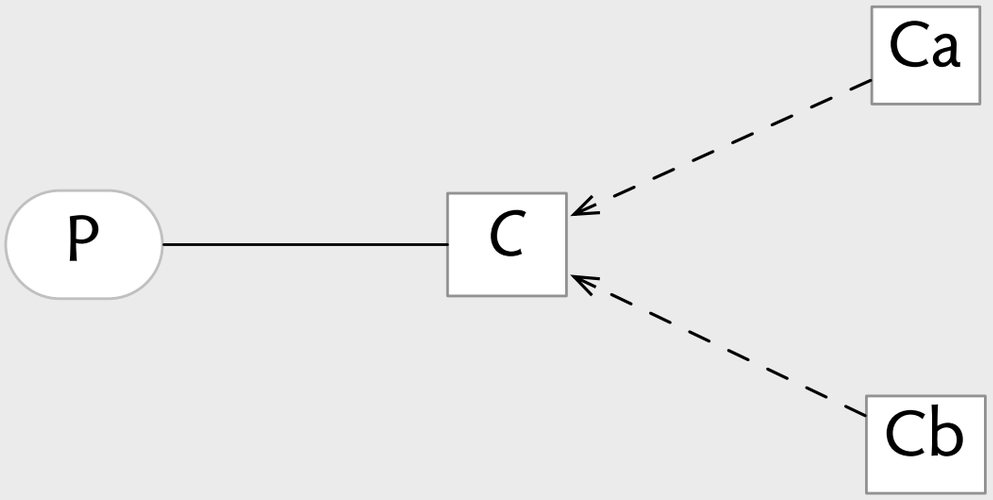
uniformity



uniformity is violated when
instantiations differ with respect to
fulfillment of purpose

... directly or indirectly

non-uniform concepts



deposits by check (banking)
› funds arrive before clearing

primitive type (Java)
› not like a class type

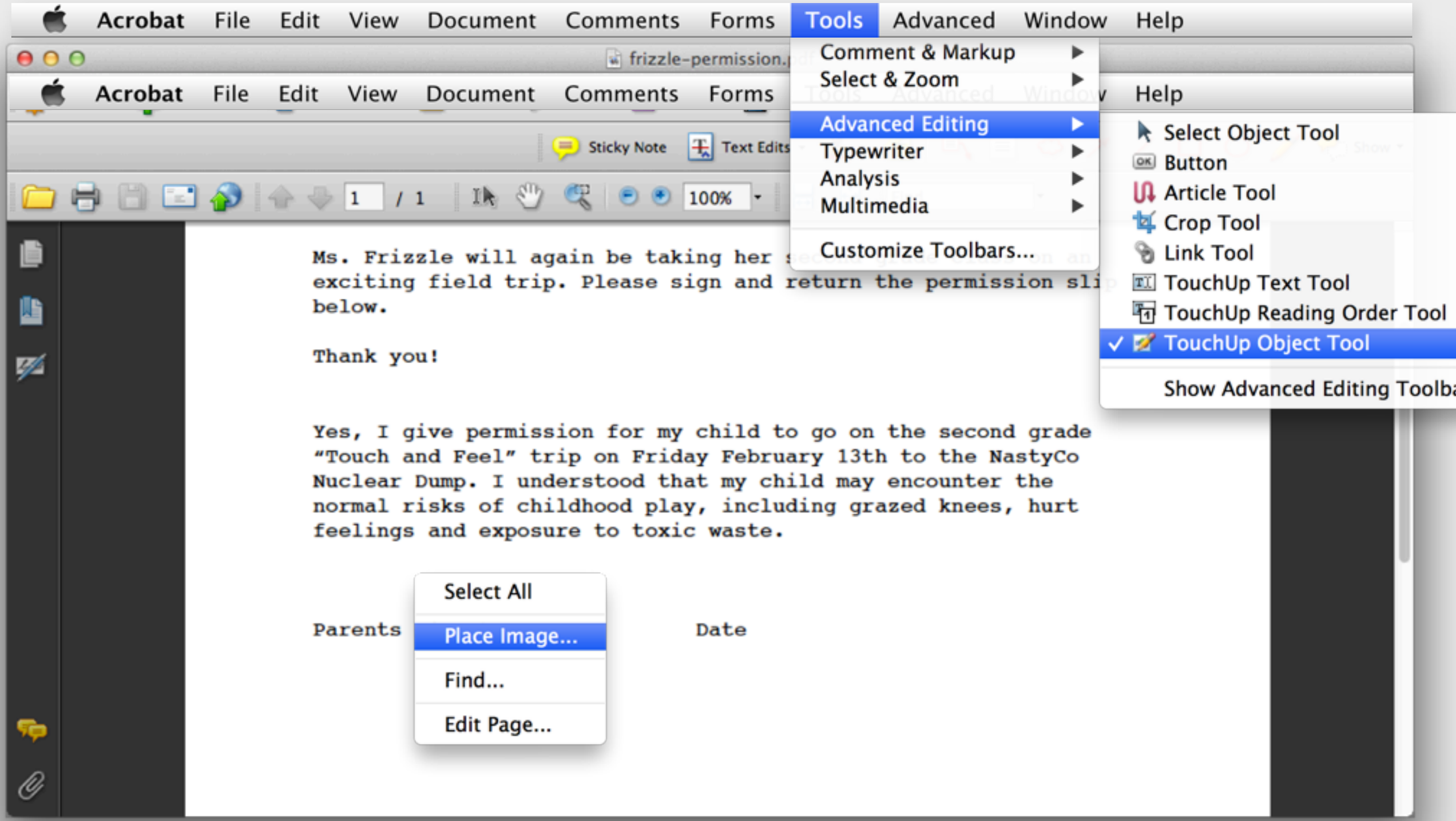
direct flight (Official Airline Guides)
› 1 flight number, but >1 stop

alerts (Apple iCal)
› can't select email alerts for default

custom settings (Fuji x100s)
› only some settings selectable

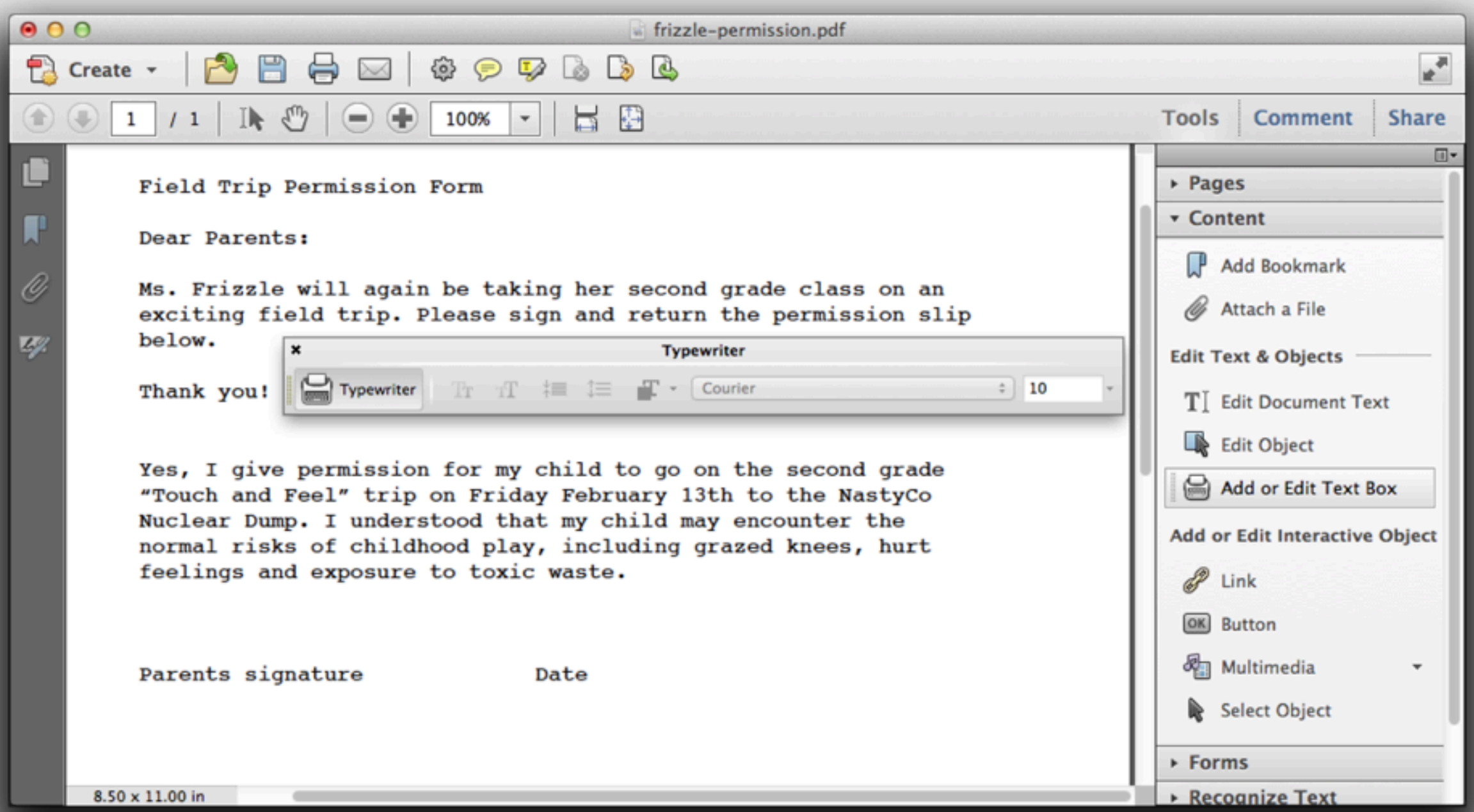
all's well that ends well

acrobat (version 09)



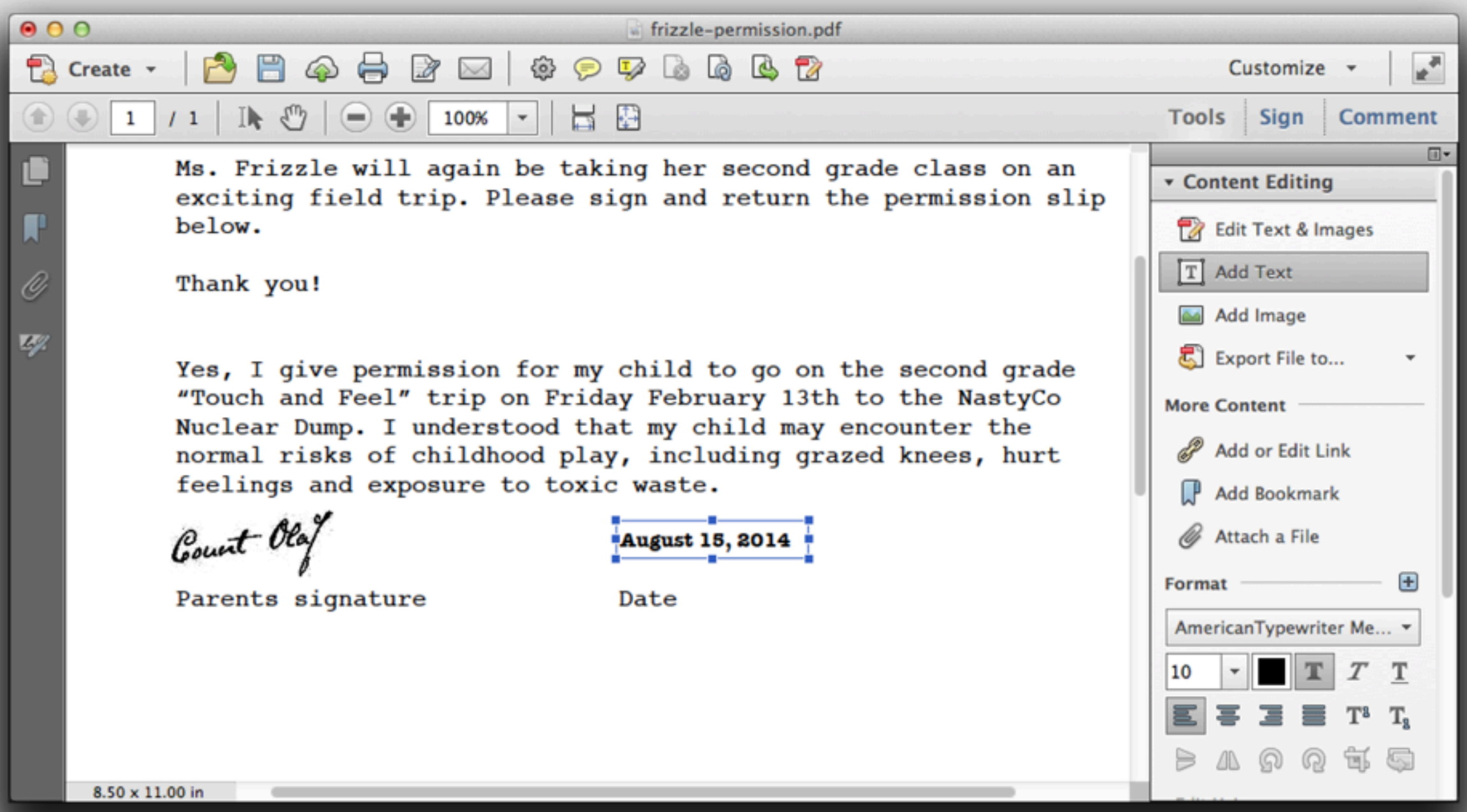
hard to discern any compelling concepts

acrobat (version 10)



user interface has been reworked but still *text*, *text box*, *object*

acrobat (version 11)



conceptual reworking: now just *text*

(mis)applying classic analysis

classic formal design analysis

descriptions

concept $\overset{?}{\Rightarrow}$ purpose ?

behaviors

defined $\subseteq$ expected


John Guttag & Jim Horning

Formal Specification as a Design Tool (1980)

misfit #1: purposes ⊈ goals

style: 'consistent formatting'
doesn't just mean do it once!
means that it's easy to maintain

layer: 'non destructive edits'
doesn't just mean edit can be undone
means that edits can be turned on/off, replayed

request: 'take user to chosen floor'
doesn't mean don't stop on the way
doesn't mean 'eventually'

misfit #2: errors ⊈ counterexamples

Christopher Alexander
misfits

changeable
wall display of prints

and: strong enough,
no damage to prints...

fulfills?

motivates

adhesive corners

not strong enough

corkboard tiles

damages prints

magnetic paint

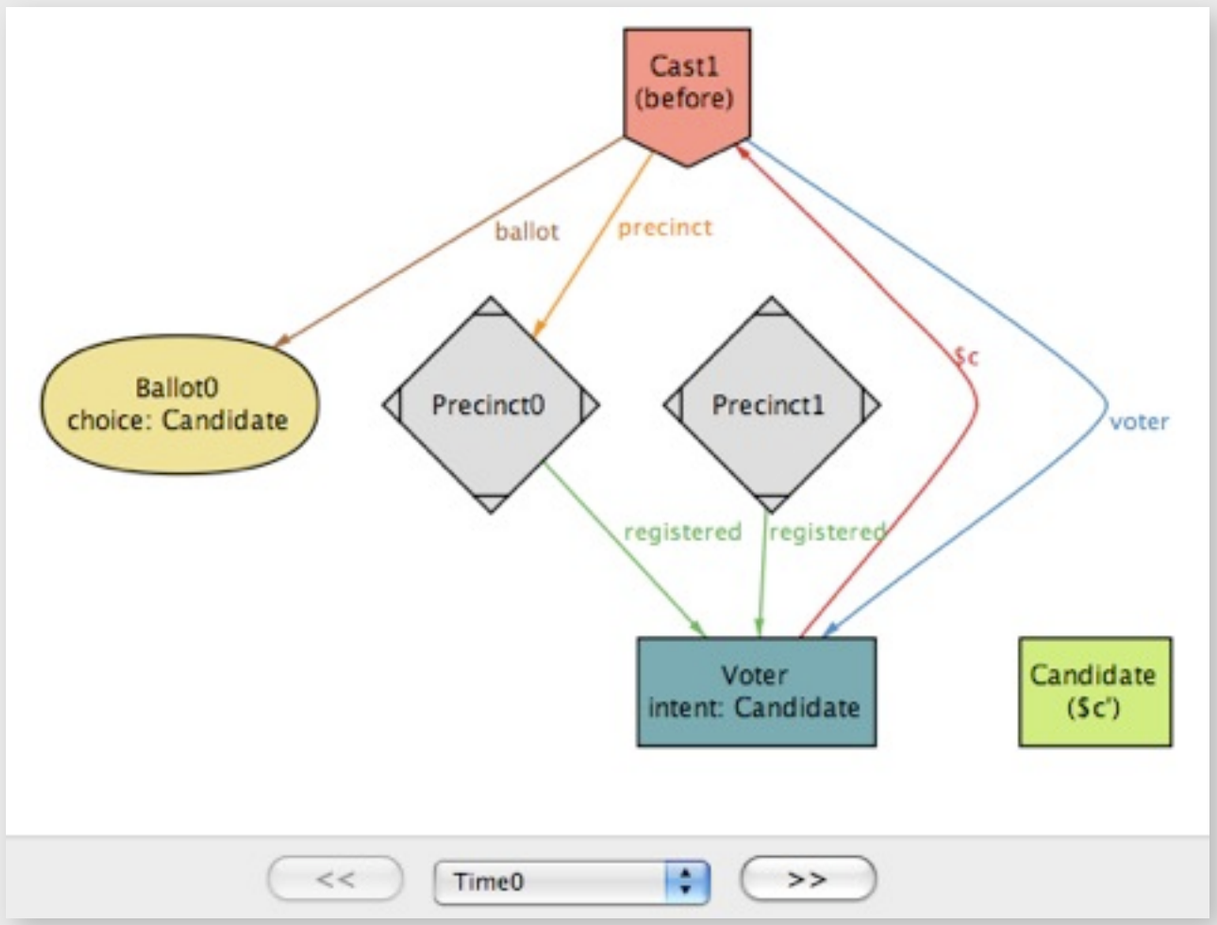
shields wifi signal

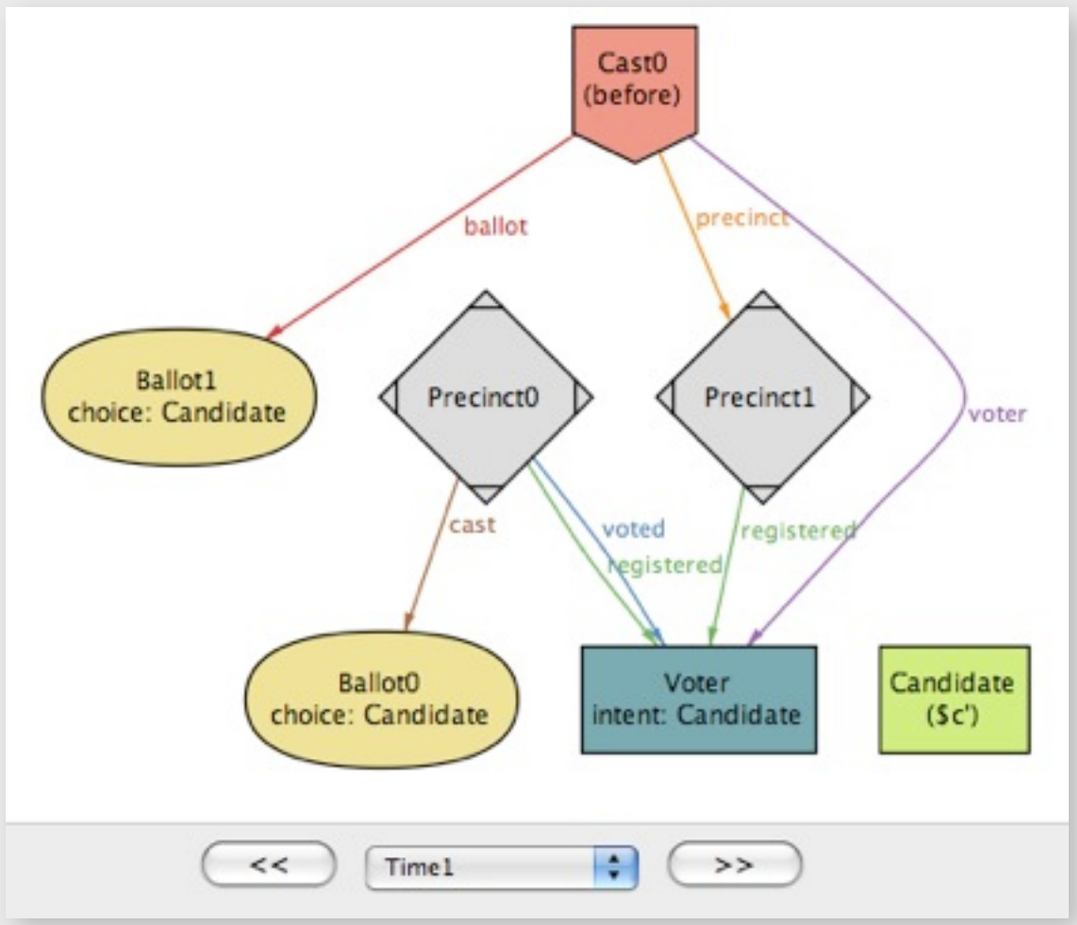
misfit #2: errors ⊈ counterexamples

projection: 'visualize time-varying relation as cartoon'

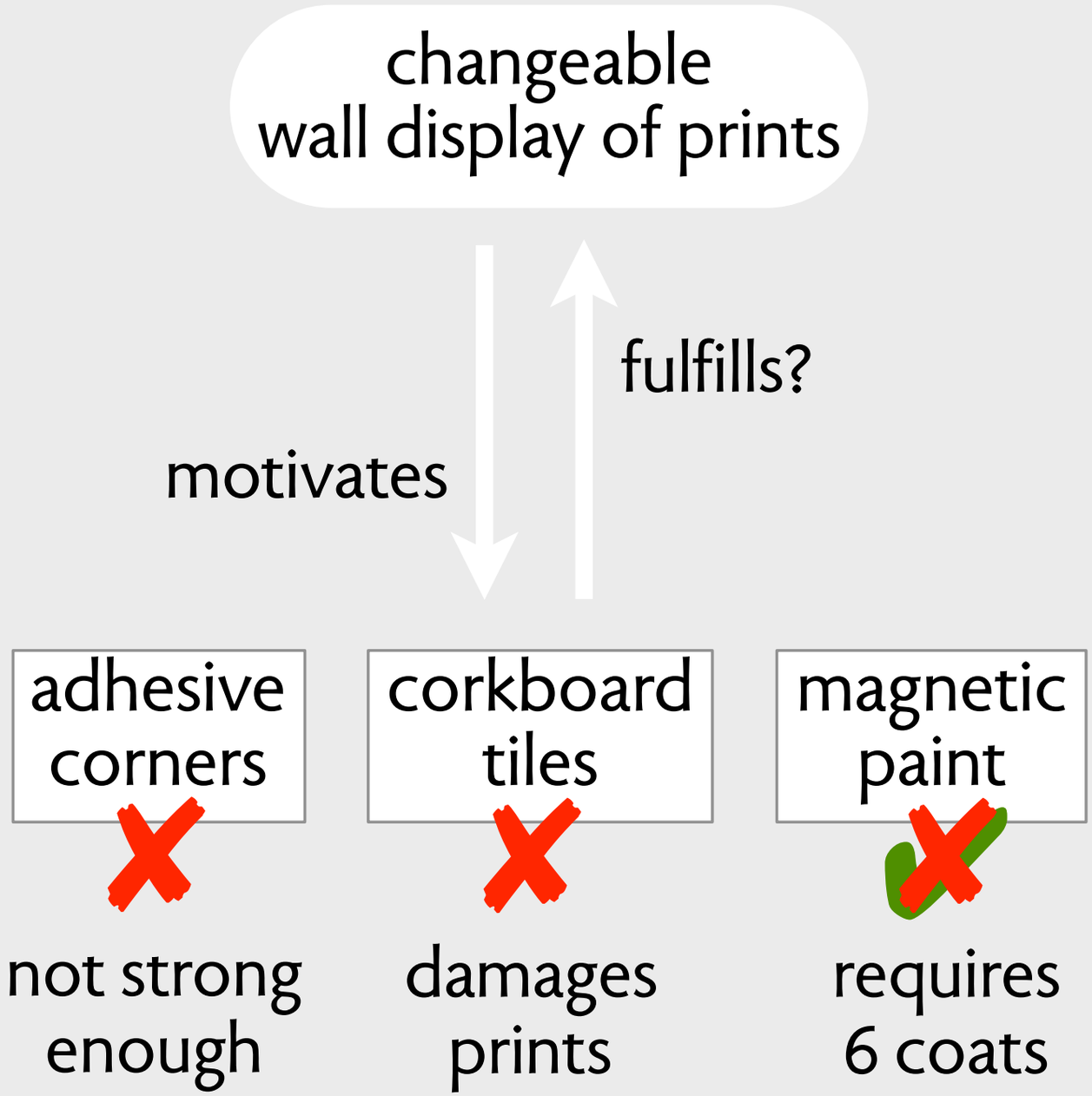
a key concept in the Alloy visualizer

bad flaw: independent layout of each cartoon frame



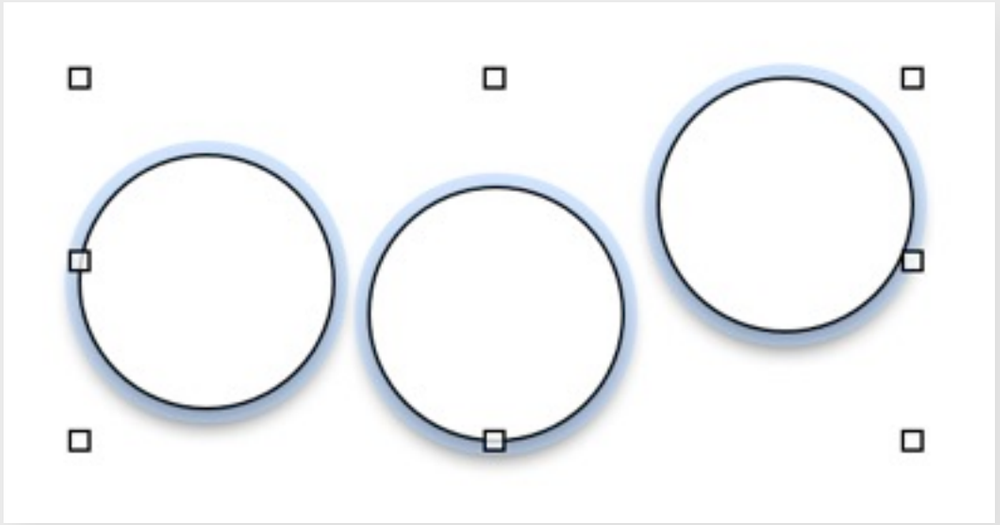


misfit #3: user behavior isn't fixed

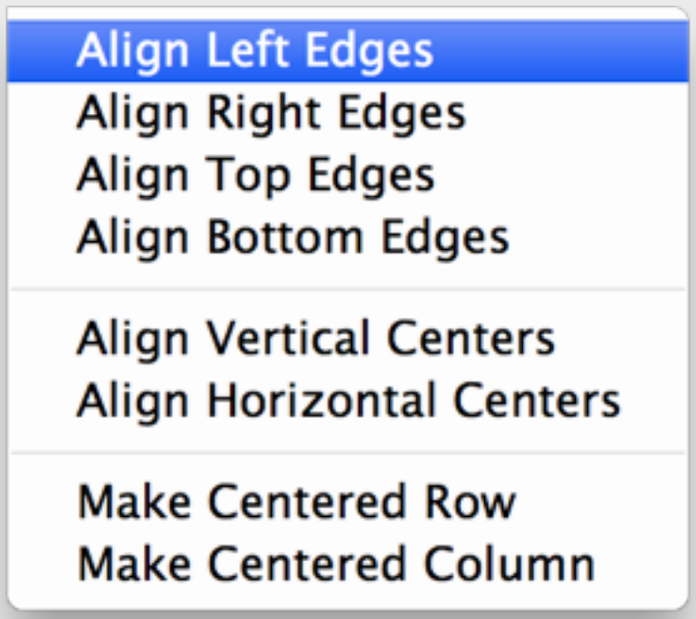


misfit #3: user behavior isn't fixed

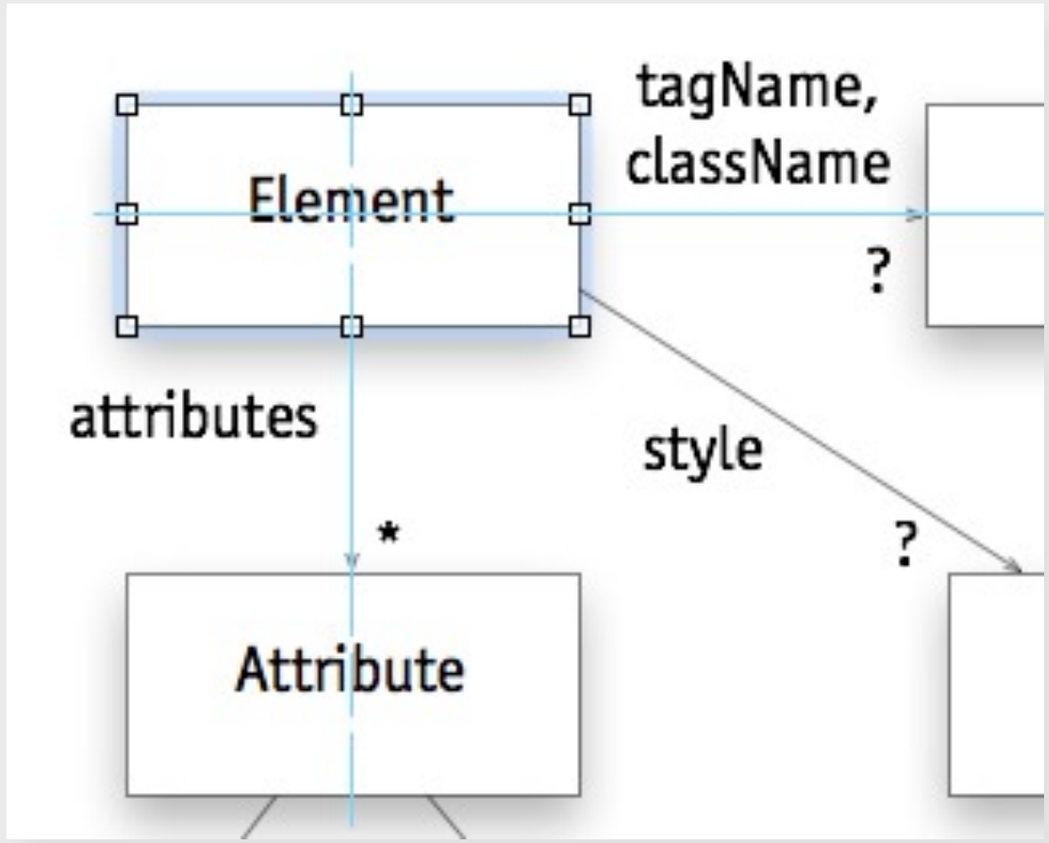
purpose: help align objects



a design: autoalign



a better design: snap align



the root of the problem

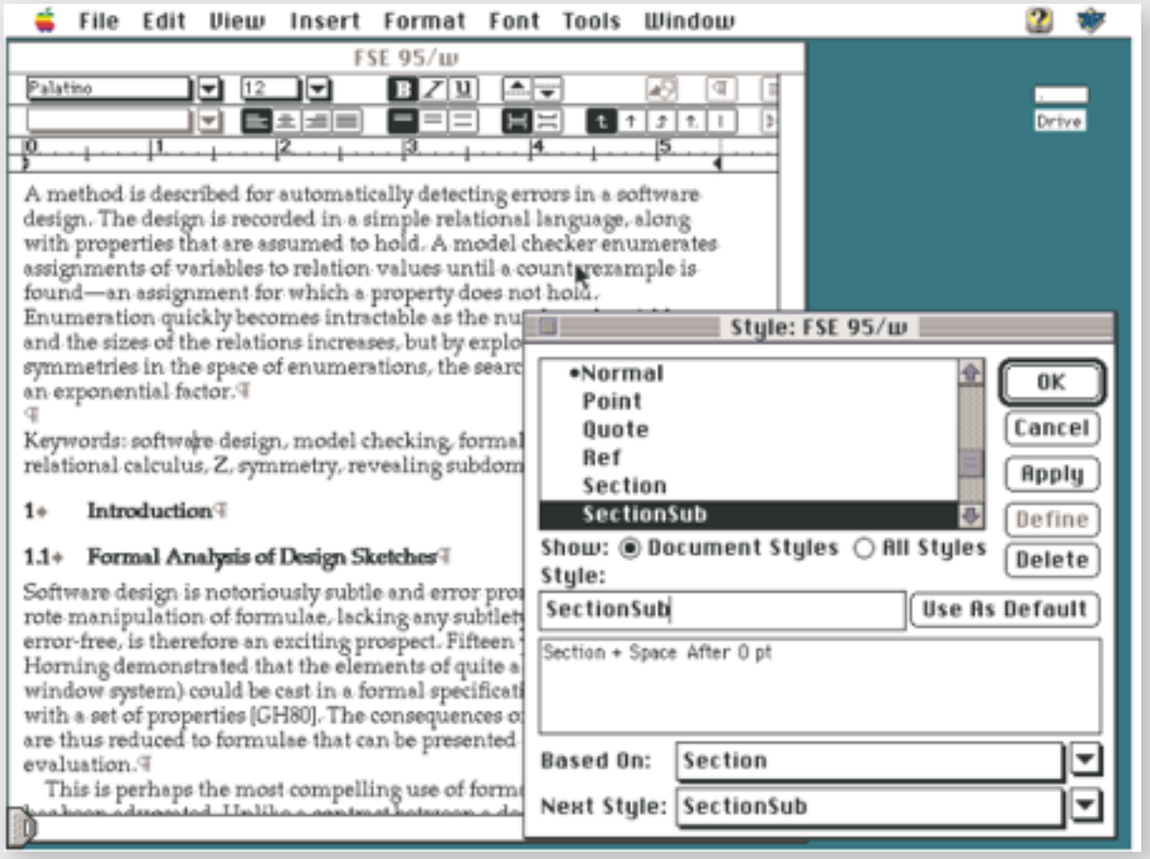
The rôle of a formal functional specification is simply to act as a logical firewall between two completely different concerns... The **pleasantness problem** concerns the question whether a system... would satisfy our needs... The correctness problem concerns the question whether a given design meets such-and-such a formal functional specification. The logical firewall ... isolates computing science's well-carved niche from the pleasantness problem to which science has little to contribute. Please note that I did not say that the one problem is more important than the other; after all, no chain is stronger than its weakest link.

Edsger Dijkstra
EWD952

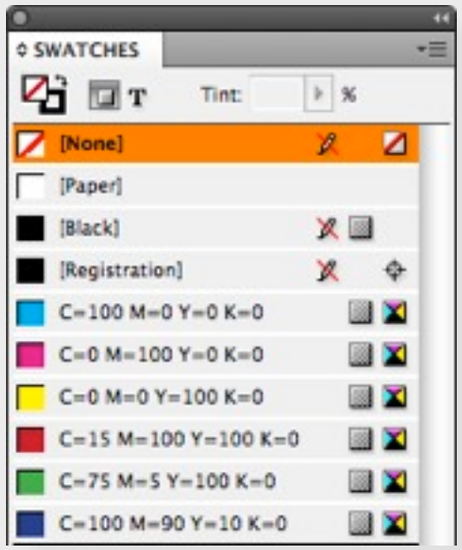
what were we analyzing?

Elements of Style:
Analyzing a Software Design Feature
with a Counterexample Detector

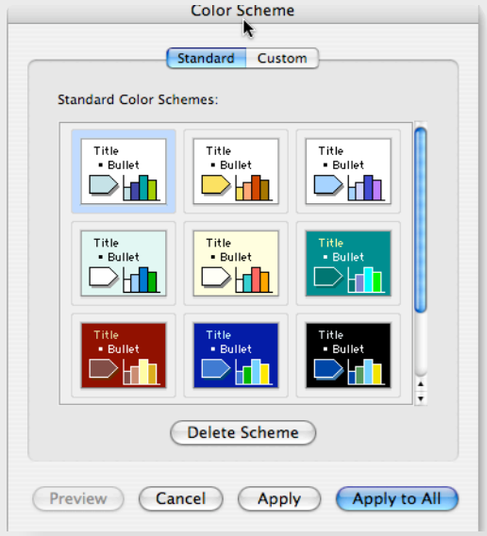
Jackson & Damon, ISSTA'96



Microsoft Word



Indesign swatches



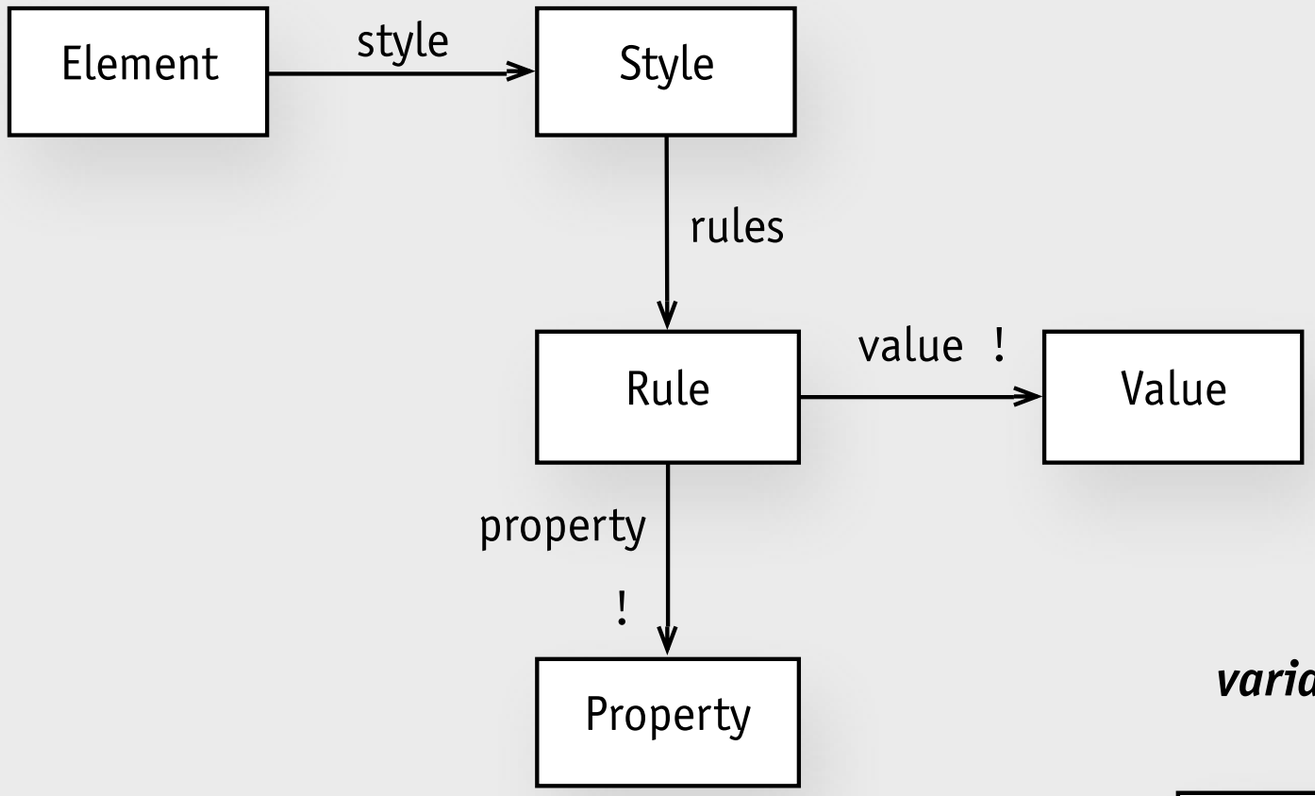
Powerpoint schemes

a style concept idiom

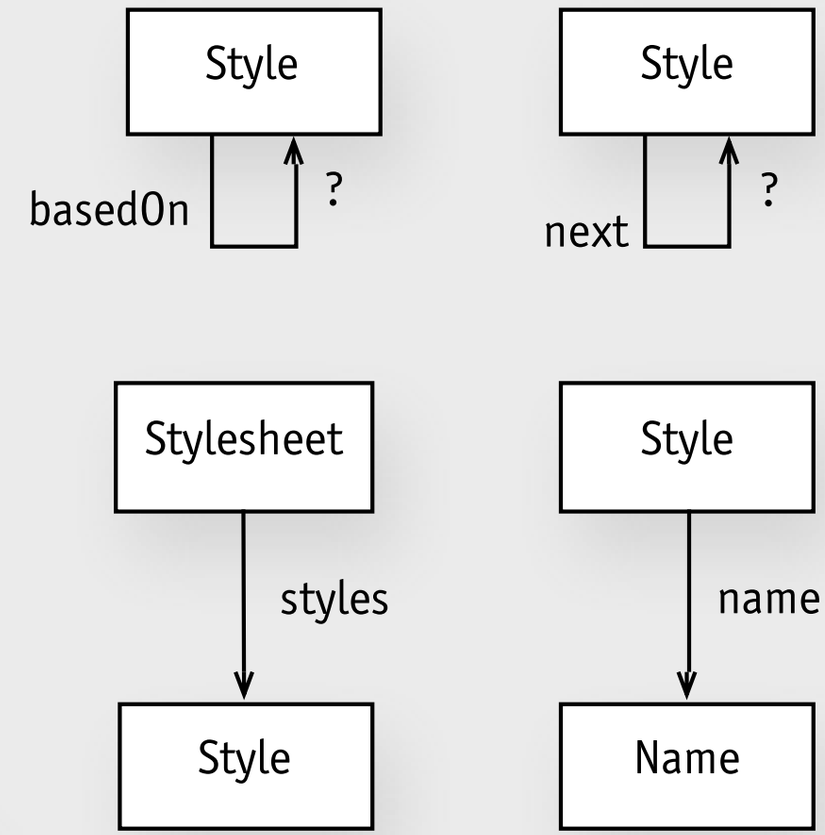
concept name	style
purpose	achieve consistent formatting of a set of elements
known instances	para and char styles in Word, Indesign, Pages, Keynote 6; Indesign swatches; Powerpoint themes; classes in CSS
related to	style buffer, stencil, master
challenges	as-is inheritance; partially defined styles; precedence when overlapping

style idiom object model

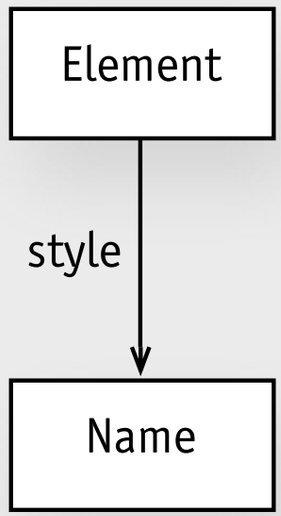
base



add ons



variant



consistency	achieve consistency
style	achieve consistent structure
master	use archetypal objects for consistency
stencil	reformat another object like a previous one to save effort
style buffer	allow setting of many properties at once
preset	

organization	organize collection of items into a hierarchy
folder	organize collection of resources by simple path names
REST	group items so they can be treated as a single item
group	allow easy inclusion/exclusion of sets of items
layer	place items in stacking order for precedence
stack	apply action in aggregate to many items at once
selection	add labels to items so they can be found later
label	address one or more items with a shorthand name
alias	allow filtering of set of objects by their features
filter	describe an object with properties that have values
property	sort and search for items using associated data
metadata	

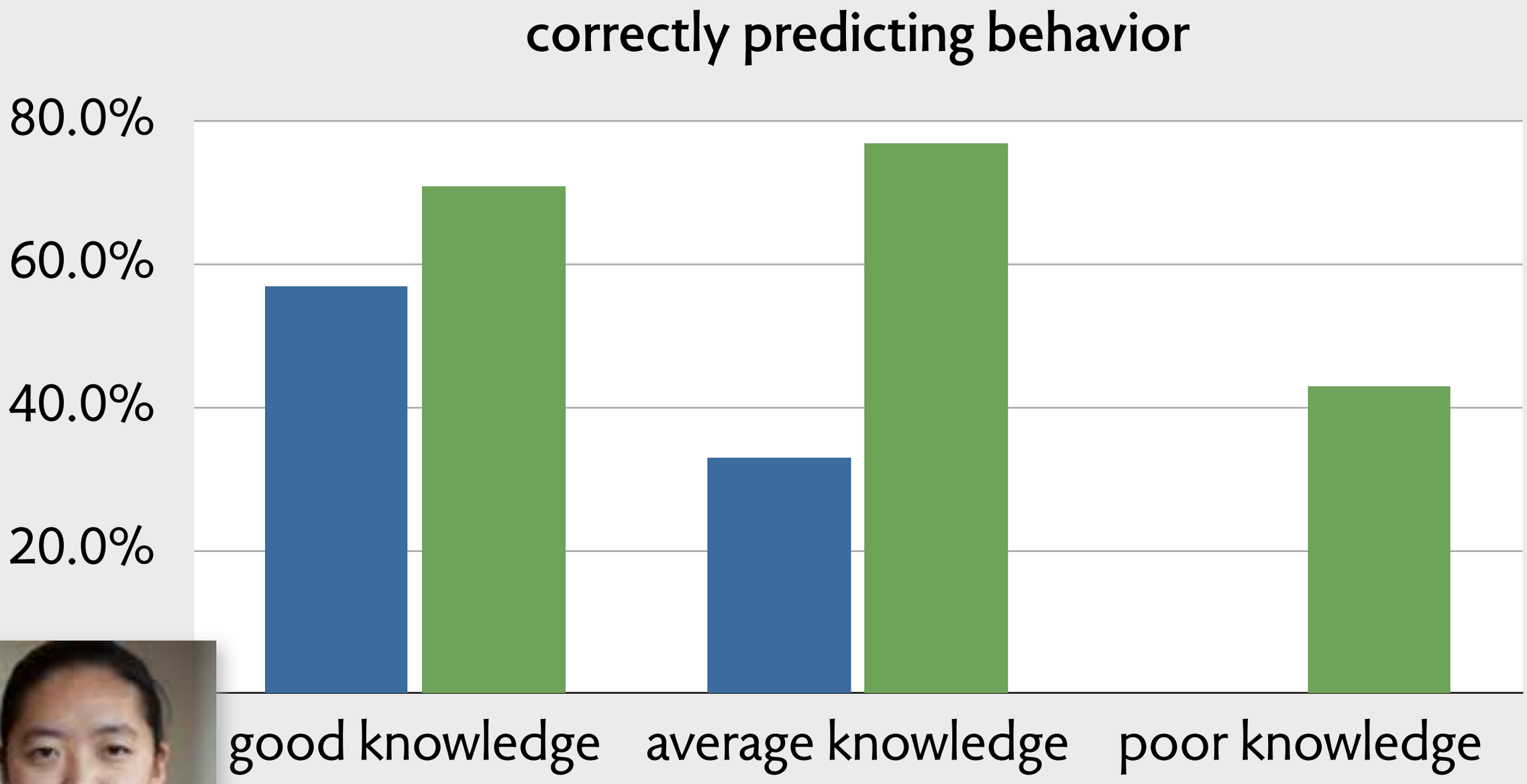
navigation	keep past actions for audit, undo, visibility
history	provide temporary storage area for quick modification
buffer	provide shortcut entry into traversable document
cursor	

access	control access to a resource in an easy way
access token	allocate resources efficiently and prevent conflicts
reservation	authenticate user with 'out of bound' channel unique to user
OOBA	

communication	communicate in discrete packets between endpoints
	share a short communication by broadcast
	know when something happens

design case studies

small survey of MIT dropbox users



Kelly Zhang

gitless: a reworking of git

Gitless: a version control system

Fork me on GitHub

About Gitless

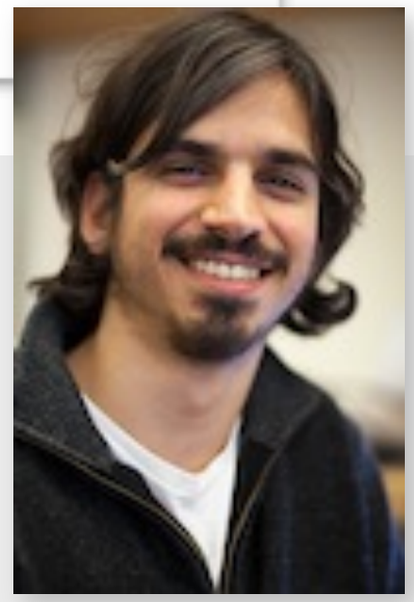
Gitless is an experimental version control system built on top of Git. Many people complain that Git is hard to use. We think the problem lies deeper than the user interface, in the concepts underlying Git. Gitless is an experiment to see what happens if you put a simple veneer on an app that changes the underlying concepts. Because Gitless is implemented on top of Git (could be considered what Git pros call a "porcelain" of Git), you can always fall back on Git. And of course your coworkers you share a repo with need never know that you're not a Git aficionado.

Check out the documentation to get started. If you are a novice user that never used any version control system the documentation should be enough to get you started. If you are a Git pro looking to see what's different from your beloved Git you'll be able to spot the differences by glancing through the Gitless vs. Git section.

Download

- Mac OS X Binary (.tar.gz)
- Linux Binary (.tar.gz)
- Source Code (.tar.gz)

For installation instructions see the readme file. After installation, you should be able to execute the `gl` command. The current Gitless version is 0.7 which was released on 4/2015 (release notes).



Santiago
Perez De Rosso

a sample git misfit

sample misfit

you're working in a branch and want to switch to another
but there are uncommitted changes
you can't switch without overwriting work in the new branch
you don't want to commit unfinished work
best option is to stash, but even that doesn't work if conflicts

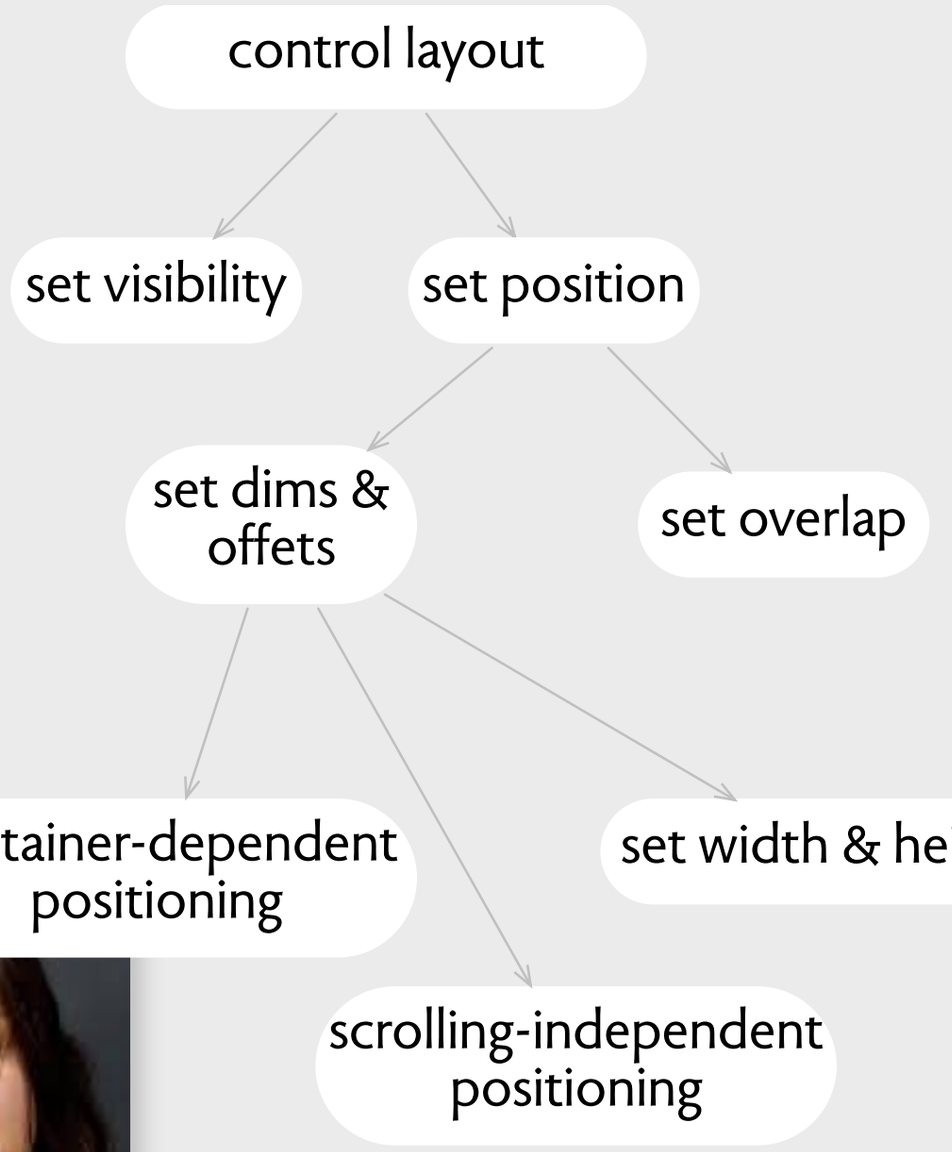
diagnosis

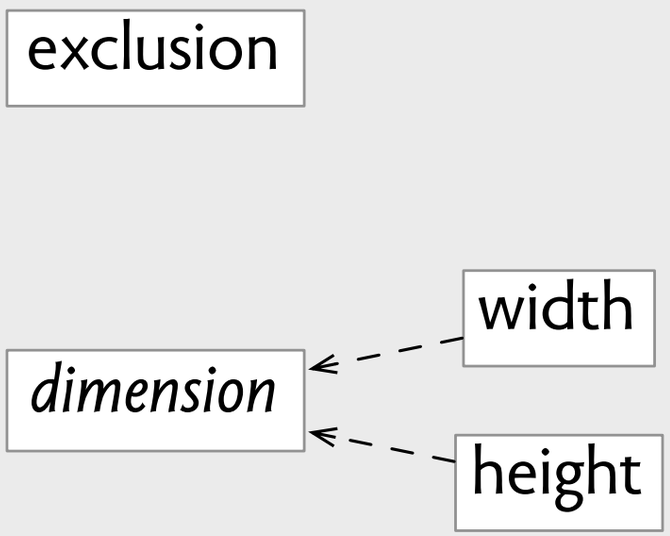
staging area, working directory and branch are coupled
and stash is an unmotivated concept

gitless: new concept of branch

when you switch branches, the working directory changes with it
and all uncommitted changes in the old branch are saved

why is css so damn hard?





Lea Verou

hypothesis: driving design by use cases: gets hopelessly coupled
a lesson for agile development?

code analysis opportunities

are defects clustered around fragile concepts?
predict where next defect will be?
focus refactoring efforts?

can concept coupling be inferred?
map modules to concepts & look for interactions

find needless dependences?
compare code & concept dependences
identify dependences due to implementation

conclusions

summary

concepts
a better way to design software

dependence graph
shape an MVP
explore radical redesigns

operational principle
lightweight test of a concept design

concept-purpose mapping
identify concepts to kill
investigate overloaded concepts for misfits

join the great concept hunt

researcher seeking...

nifty concepts

troubled concepts

get a reward!

an ack & a concept T shirt

for every example I use in my book

dnj@mit.edu