my mid-life crisis
a traditional view
quality = code correctness

my career so far
lightweight formal models
design → code
exploration → certification

but now I’m wondering
do details matter?
are these the wrong details?
are we looking under the lamppost?

rethinking my assumptions
dropbox woes
a sad dropbox tale

Someone accidentally deleted thousands of files in my company Dropbox: how can I quickly undelete them?
Reader Paul Cramblett has a problem with others who just don’t know how to share. He writes:

I maintain a Dropbox folder that I use to share files with a select group of friends. I've tried to explain how Dropbox works to these people but someone invariably drags all the files out of the folder, which means they're no longer available to the rest of us. Is there some way to prevent files from being removed by someone who doesn’t understand the difference between “copy” and “move”? 
Share 'party' with others

- Alyssa P. Hacker
dnj+alice@csail.mit.edu

Allow members to invite others

Share folder  Cancel
alyssa spoils everything

Dropbox

Delete folder?

Are you sure you want to delete party from your Dropbox?

Delete
survey of MIT Dropbox users

correctly predicting behavior

- good knowledge: delete shared folder results in leaving
- average knowledge: delete shared subfolder removes it
- poor knowledge: delete shared subfolder removes it

Kelly Zhang
the software problem
needs

requirements
“everyone inside holds a ticket”
“every ticket has been paid for”

specifications
buy: charge, record, issue ticket
enter: if ticket is valid

charge/record not atomic

forgot staff, press passes, family tickets

visitor gets to stile before database update

code

DATA BASE

TICKET DESK

ENTRY STILE

CARD ISSUER

DATABASE

charge
record
validate
needs

requirements
“everyone inside holds a ticket”
“every ticket has been paid for”

specifications
buy: charge, record, issue ticket
enter: if ticket is valid

charge/record not atomic

visitor gets to stile before database update

must get these right

forgot staff, press passes, family tickets

but these matter more

g_database lost: all issued tickets now invalid

g_forgery: people enter with fake tickets

g_scalping: no tickets available, prices rocket

this is a design!
the software problem, revisited
needs
- purposes
  - concept
    - purposes
  - concepts
- code

Balance budget
Grow attendance
Customer satisfaction

get visitors to pay
- Ticket
- Button
- Queue

limit crowds
purposes
needs

not a design!

purposes

concept

concepts

purposes
code

Balance budget
Grow attendance
Customer satisfaction

get visitors to pay

limit crowds

Ticket

Button

Queue

database lost: all issued tickets now invalid

scalping: no tickets available, prices rocket

forgey: people enter with fake tickets

these are misfits of the ticket concept

this is a design!

this is a design!
Such a list of requirements is potentially endless... But if we think of the requirements from a negative point of view, as potential misfits, there is a simple way of picking a finite set. This is because it is through misfit that the problem originally brings itself to our attention. We take just those relations between form and context which obtrude most strongly, which demand attention most clearly, which seem most likely to go wrong. We cannot do better than this.
so how to prevent misfits?

- avoid premature design
  - eg, conventional requirements
- analyze concept design
- articulate purpose & mechanism
- learn from the past
  - most concepts used before!
concept
parts
the data model
what the concept is about
the purpose
what the concept is for

motivating purpose
get visitors to pay

separate use of resource from granting of use

refined generic purpose
the actions

what the concept does

issue: exchange money for ticket
validate: check that ticket is valid
use: customer uses resource

a joint action

a user action

a system action
the operational principle
how the concept works

an archetypal scenario explains how to use the concept
shows how purpose if fulfilled

“user requests that system issue ticket; user presents ticket for validation; if successful, user can then use resource”

“you insert some money and the machine gives you a ticket; then you insert the ticket into the turnstile, and it lets you in”

generic OP

specific OP
Engineering includes the operational principles of machines... Physics and chemistry, on the other hand, include no knowledge of the operational principles of machines. Hence a complete physical and chemical topography of an object would not tell us whether it is a machine, and if so, how it works, and for what purpose.

Michael Polanyi
# ticket concept

<table>
<thead>
<tr>
<th>name</th>
<th>Ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td>purpose</td>
<td>separate use of resource from granting of use</td>
</tr>
<tr>
<td>sample uses</td>
<td>event sales; software licensing; road pricing; flights</td>
</tr>
<tr>
<td>mechanism</td>
<td>data, actions, operational principle</td>
</tr>
<tr>
<td>variants</td>
<td>transferable, one-time/repeating, open/expiring</td>
</tr>
<tr>
<td>misfits</td>
<td>forgery, scalping, transfers</td>
</tr>
<tr>
<td>related to</td>
<td>Reservation, Coupon, Credit</td>
</tr>
</tbody>
</table>
dropbox revisited
purpose
organize files with localized namespaces

operational principle
if you create a folder with a name N in a folder that has a pathname P, and put objects inside it, you can then access them at the name P/N

misfits
delete is not destroy new name, new file

data model

```
Name
  ↓
Entry
  ↓
Object
  ↓
File
  ↓
Folder
  ↓
name
  ↓
entries
  ↓
links
```

- File
- Folder
- Object
- Entry
- Name
synchronized folder

purpose
keep copies of files consistent across machines

misfits
propagated deletion
propagated temps

operational principle
if you say two drives should be synced, then changes in one are copied to the other

data model
Folder
Drive
sync
root
shared folder

**purpose**
allow users to share files and folders

**operational principle**
if a user who owns a folder shares it with another user, that user can mount the folder in a folder of theirs and then read and write it

**misfits**
can’t share subfolder

**data model**

- **Object**
  - File
  - Folder
  - User
  - Entry
  - Name

- links, mounts
- owns, shares
- entries

name
how to evaluate a concept
the fundamental principle

in a well-designed system
each concept is motivated by one purpose
the ideal mapping

purposes

P1

concepts

C1

P2

C2
4 bad smells

unfulfilled purpose

unmotivated concept

overloaded concept

redundant concepts
happy concepts
trash

**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 (and makes space on disk)

**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

**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)
selection

slides in Keynote
photos in Adobe Lightroom
messages in Apple Mail
objects in OS X Finder
thumbnails in Preview
notes in Evernote
subtlety selection scope

subset of selection in scope

subset of selection out of scope
subtlety active element

Adobe Lightroom: brightest thumbnail is the “active photo”
Photoshop: outline shown with “marching ants”

Photoshop: selection shown in Quick Mask mode
subtlety folder selection

Google Drive: selecting folder = selecting children
CrashPlan: selecting folder = selecting all future children
# selection: concept parts

<table>
<thead>
<tr>
<th>name</th>
<th><strong>Selection</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>purpose</strong></td>
<td>apply operation in aggregate to many elements</td>
</tr>
<tr>
<td><strong>sample uses</strong></td>
<td>text formatting in word processors (eg Word); styling in CSS; color themes in Powerpoint.</td>
</tr>
<tr>
<td><strong>mechanism</strong></td>
<td><em>data, actions, operational principle</em></td>
</tr>
<tr>
<td><strong>variants</strong></td>
<td>scope, active element, continuous, hierarchy</td>
</tr>
<tr>
<td><strong>misfits</strong></td>
<td>accidental deletion?</td>
</tr>
<tr>
<td><strong>related to</strong></td>
<td>Group, Folder</td>
</tr>
</tbody>
</table>
sad

concepts
### Gmail's Categories

<table>
<thead>
<tr>
<th>Primary</th>
<th>Social</th>
<th>Promotions</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Google" /></td>
<td><img src="#" alt="New sign-in from Chrome on Mac" /></td>
<td><img src="#" alt="100+ new" /></td>
</tr>
<tr>
<td><img src="#" alt="Keith Muhammad at DeMont." /></td>
<td><img src="#" alt="DeMontrond Auto Group" /></td>
<td><img src="#" alt="August 26" /></td>
</tr>
<tr>
<td><img src="#" alt="AT&amp;T High Speed Internet." /></td>
<td><img src="#" alt="AT&amp;T High Speed Internet Service Activation" /></td>
<td><img src="#" alt="10:37 am" /></td>
</tr>
<tr>
<td><img src="#" alt="Keith Muhammad at DeMont." /></td>
<td><img src="#" alt="DeMontrond Auto Group" /></td>
<td><img src="#" alt="August 26" /></td>
</tr>
<tr>
<td><img src="#" alt="betterbatonrougejobs.com" /></td>
<td><img src="#" alt="Job Update -- 2015-08-26" /></td>
<td><img src="#" alt="August 26" /></td>
</tr>
</tbody>
</table>
Choose which message categories to show as inbox tabs. Other messages will appear in the Primary tab.
Like many Gmail users, I greeted the news of the introduction of tabs to the interface with a degree of anticipation -- now it was just a matter of waiting for the feature to roll out so I could try it for myself. Earlier today I was randomly signed out of my Gmail account, and after signing back in and checking the settings menus, I could see that tabs were now available to me. Excitement was short-lived, however; it quickly became apparent that this new feature is a disaster.
Using labels

Labels help you organize your messages into categories – work, family, to do, read later, jokes, recipes, any category you want. Labels do all the work that folders do, but with an added bonus: you can add more than one to a message.
what you can’t do

associate tabs with labels
feature available only for categories

use tabs outside inbox
tabs disappear when you filter on a label
redundant concepts

classify messages

label
category

category in Gmail
a redundant concept
camera settings
my camera fuji x100s
image quality setting
aspect ratio
image size setting
non-standard ratio + raw?
what you can’t do

non-standard aspect ratio + raw
even though raw images get nice nondestructive crop!
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]

3 forms of overloading:
- **piggybacking** new purpose hacked onto old concept
- **false convergence** two purposes looked the same
- **emergent purpose** users found second purpose for concept
piggybacking fuji camera
new purpose hacked onto old concept

image size
aspect ratio piggybacked
on JPEG dimensions
piggybacking epson driver

result: can’t create custom size for front loading
also, page size presets in Lightroom hold feed setting
false convergence
two purposes looked the same

Performance Review Form 2016

Performance Summary

Evaluate and discuss the employee’s performance. Base your evaluation on the position requirements, MIT Core Competencies, achievement of the goals established during the past year, and your assessment of the employee’s accomplishments.

Goal Setting

State and discuss the expectations and goals for the upcoming review period. Give examples of how these goals can be met (e.g., training). How will you support the employee to accomplish these goals? Reference here

evaluation & goal setting
incompatible purposes
false convergence

two purposes looked the same

filter incoming posts
control access to my posts

distinct purposes

2011: Facebook added subscribe/follow
emergent purpose
users find second purpose for concept

To: Daniel Jackson <dnj@mit.edu>
Re: Catch me if you can in real life!
initial purpose: summarize content

To: csail-related@lists.csail.mit.edu
Re: [csail-related] turn off the lights?
emergent purpose: show sender
if you bcc a list, subject reveals to-address
thanks to Shriram Krishnamurthi

To: Daniel Jackson <dnj@mit.edu>
your trip reservation
emergent purpose: group by conversation
can’t label reservations from Expedia by trip
thanks to Eunsuk Kang
fonts & styles
what's a font?
what you can’t do

define a style that italicizes
  Arno Regular to Arno Italic
  Futura Book to Futura Book Oblique
  Magma Light to Magma Light Italic
Adobe InDesign: An unfulfilled purpose

allow typeface independent styling

subfamily
introducing a concept

Keynote ’09: has subfamilies

Keynote 6: gone again!
rethinking Git
Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, **convenient staging areas**, and **multiple workflows**.

Learn Git in your browser for free with **Try Git**.
NAME

git-rebase - Forward-port local commits to the updated upstream head

SYNOPSIS

```
git rebase [-i | --interactive] [options] [--exec <cmd>] [--onto <newbase>] [<upstream> [<branch>]]
git rebase [-i | --interactive] [options] [--exec <cmd>] [--onto <newbase>] --root [<branch>]
git rebase --continue | --skip | --abort | --edit-todo
```

NAME

git-rebase - Forward-port local commits to the updated upstream head

SYNOPSIS

```
git rebase [-i | --interactive] [options] [--exec <cmd>] [--onto <newbase>] [<upstream> [<branch>]]
git rebase [-i | --interactive] [options] [--exec <cmd>] [--onto <newbase>] --root [<branch>]
git rebase --continue | --skip | --abort | --edit-todo
```

one by one, in order. Note that any commits in HEAD which introduce the same textual changes as a commit in HEAD. <upstream> are omitted (i.e., a patch already accepted upstream with a different commit message or timestamp will be skipped).

It is possible that a merge failure will prevent this process from being completely automatic. You will have to resolve any such merge failure and run `git rebase --continue`. Another option is to bypass the commit that caused the merge failure with `git rebase --skip`. To check out the original <branch> and remove the .git/rebase-apply working files, use the command `git rebase --abort` instead.

Assume the following history exists and the current branch is "topic":

```bash
  HEAD -> upstream/master
  upstream/master -> branch
```

After applying changes to branch, switch to upstream/master:

```
git rebase upstream/master
```
NAME

git-wave-stash — wave all staged stashes next to various cherry-picked non-applied applied trees

SYNOPSIS

git-wave-stash --predict-whistle-tree --dodge-pack

OPTIONS

--predict-whistle-tree
  the subtree should not be flashed by a requested pack

--dodge-pack
  fast-import the histories of a few files that are parsed

SEE ALSO

git-gouge-head(1), git-strip-history(1), git-recommend-pack(1), git-tilt-branch(1)
NAME

git-distinguish-tree — distinguish a few non-cleaned remote trees inside various rev-listed upstreams

SYNOPSIS

git-distinguish-tree [ --distinguish-grope-history | --relieve-ref | --delineate-log ]

--distinguish-grope-history
import the bases of a few files that are archived

--relieve-ref
use ref to checkout origins/stages/ to an exported ref

--delineate-log
save the histories of a few stages that are failed

SEE ALSO

git-engineer-submodule(1), git-lecture-archive(1)
NAME

git-control-stash — control some non-bundled staged stashes over any shown submodules

SYNOPSIS

git-control-stash [ --steer-stash | --scout-area | --collide-index-origin ]

DESCRIPTION

SEE ALSO

git-page-path(1), git-pocket-stash(1), git-race-head(1)
This is Git. It tracks collaborative work on projects through a beautiful distributed graph theory tree model.

Cool. How do we use it?

No idea. Just memorize these shell commands and type them to sync up. If you get errors, save your work elsewhere, delete the project, and download a fresh copy.
conceptual problems in Git

unmotivated concept

P1
C1
C2

overloaded concept

P1
C1
P2

example: stash
pseudo purpose: overcome misfits in branching concept

example: commit
P1. group logically related changes
P2. save files to protect against loss
<table>
<thead>
<tr>
<th>Misfit</th>
<th>Question</th>
<th>Upvotes</th>
<th>Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saving Changes</td>
<td>Q1 Using Git and Dropbox together effectively?</td>
<td>927</td>
<td>215523</td>
</tr>
<tr>
<td></td>
<td>Q2 Backup a Local Git Repository</td>
<td>122</td>
<td>78674</td>
</tr>
<tr>
<td></td>
<td>Q3 Fully backup a git repo?</td>
<td>54</td>
<td>37502</td>
</tr>
<tr>
<td></td>
<td>Q4 Is it possible to push a git stash to a remote repository?</td>
<td>105</td>
<td>30820</td>
</tr>
<tr>
<td></td>
<td>Q5 Git fatal: Reference has invalid format: refs/heads/master</td>
<td>90</td>
<td>25717</td>
</tr>
<tr>
<td></td>
<td>Q6 Is “git push --mirror” sufficient for backing up my repository?</td>
<td>34</td>
<td>18415</td>
</tr>
<tr>
<td></td>
<td>Q7 How to back up private branches in git</td>
<td>33</td>
<td>10580</td>
</tr>
<tr>
<td>Switching Branches</td>
<td>Q8 The following untracked working tree files would be overwritten by checkout</td>
<td>365</td>
<td>378331</td>
</tr>
<tr>
<td></td>
<td>Q9 Git: Switch branch and ignore any changes without committing</td>
<td>148</td>
<td>129120</td>
</tr>
<tr>
<td></td>
<td>Q10 Why git keeps showing my changes when I switch branches (modified, added, deleted files) no matter if I run git add or not?</td>
<td>47</td>
<td>10524</td>
</tr>
<tr>
<td>Detached Head</td>
<td>Q11 Git: How can I reconcile detached HEAD with master/origin?</td>
<td>784</td>
<td>397694</td>
</tr>
<tr>
<td></td>
<td>Q12 Fix a Git detached head?</td>
<td>490</td>
<td>397985</td>
</tr>
<tr>
<td></td>
<td>Q13 Checkout GIT tag</td>
<td>125</td>
<td>98328</td>
</tr>
<tr>
<td></td>
<td>Q14 git push says everything up-to-date even though I have local changes</td>
<td>113</td>
<td>79203</td>
</tr>
<tr>
<td></td>
<td>Q15 Why did my Git repo enter a detached HEAD state?</td>
<td>202</td>
<td>78856</td>
</tr>
<tr>
<td></td>
<td>Q16 Why did git set us on (no branch)?</td>
<td>65</td>
<td>41866</td>
</tr>
<tr>
<td></td>
<td>Q17 gitx How do I get my 'Detached HEAD' commits back into master</td>
<td>136</td>
<td>42794</td>
</tr>
<tr>
<td>File Rename</td>
<td>Q18 Handling file renames in git</td>
<td>315</td>
<td>242864</td>
</tr>
<tr>
<td></td>
<td>Q19 Is it possible to move/rename files in git and maintain their history?</td>
<td>367</td>
<td>153701</td>
</tr>
<tr>
<td></td>
<td>Q20 Why might git log not show history for a moved file, and what can I do about it?</td>
<td>34</td>
<td>17099</td>
</tr>
<tr>
<td></td>
<td>Q21 How to REALLY show logs of renamed files with git?</td>
<td>60</td>
<td>12923</td>
</tr>
<tr>
<td>File Tracking</td>
<td>Q22 Why does git commit not save my changes?</td>
<td>177</td>
<td>142189</td>
</tr>
<tr>
<td></td>
<td>Q23 Git commit all files using single command</td>
<td>165</td>
<td>141815</td>
</tr>
<tr>
<td></td>
<td>Q24 Ignore files that have already been committed to a Git repository</td>
<td>1588</td>
<td>387112</td>
</tr>
<tr>
<td></td>
<td>Q25 Stop tracking and ignore changes to a file in Git</td>
<td>975</td>
<td>353136</td>
</tr>
<tr>
<td></td>
<td>Q26 Making git &quot;forget&quot; about a file that was tracked but is now in .gitignore</td>
<td>1458</td>
<td>286435</td>
</tr>
<tr>
<td></td>
<td>Q27 git ignore files only locally</td>
<td>562</td>
<td>120700</td>
</tr>
<tr>
<td></td>
<td>Q28 Untrack files from git</td>
<td>218</td>
<td>140663</td>
</tr>
<tr>
<td></td>
<td>Q29 Git: How to remove file from index without deleting files from any repository</td>
<td>110</td>
<td>61498</td>
</tr>
<tr>
<td></td>
<td>Q30 Ignore modified (but not committed) files in git</td>
<td>135</td>
<td>38293</td>
</tr>
<tr>
<td></td>
<td>Q31 Ignoring an already checked-in directory's contents?</td>
<td>169</td>
<td>49692</td>
</tr>
<tr>
<td>Untracking File</td>
<td>Q32 Apply git .gitignore rules to an existing repository [duplicate]</td>
<td>40</td>
<td>28286</td>
</tr>
<tr>
<td></td>
<td>Q33 undo git update-index --assume-unchanged &lt;file&gt;</td>
<td>165</td>
<td>37262</td>
</tr>
<tr>
<td></td>
<td>Q34 using gitignore to ignore (but not delete) files</td>
<td>55</td>
<td>23381</td>
</tr>
<tr>
<td></td>
<td>Q35 How do you make Git ignore files without using .gitignore?</td>
<td>58</td>
<td>23709</td>
</tr>
<tr>
<td></td>
<td>Q36 Can I get a list of files marked --assume-unchanged?</td>
<td>191</td>
<td>20184</td>
</tr>
<tr>
<td></td>
<td>Q37 Keep file in a Git repo, but don’t track changes</td>
<td>74</td>
<td>15572</td>
</tr>
<tr>
<td></td>
<td>Q38 Committing Machine Specific Configuration Files</td>
<td>58</td>
<td>5934</td>
</tr>
<tr>
<td>Empty Directory</td>
<td>Q39 How can I add an empty directory to a Git repository?</td>
<td>2383</td>
<td>432218</td>
</tr>
<tr>
<td></td>
<td>Q40 What are the differences between .gitignore and .gitkeep?</td>
<td>841</td>
<td>121484</td>
</tr>
<tr>
<td></td>
<td>Q41 How to .gitignore all files/folder in a folder, but not the folder itself? [duplicate]</td>
<td>227</td>
<td>80119</td>
</tr>
</tbody>
</table>
Gitless: a version control system

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).
example: branch

**concept:** branch

**purpose:** support independent line of development

**operational principle:** when you switch branches, your working directory is synchronized with the new branch, and you can make and commit changes which will be invisible on other branches; when you’re done, you can merge the branch into the master branch...

**misfit:** can’t switch branches with uncommitted changes; can stash, but only if no conflicts...

**fix:** give branch its own working directory; when you switch branches, the working directory changes too, and the working directory associated with the previous branch is preserved
user study

experiment design
within-subjects
two hour long sessions
six tasks per session
observation + surveys

tasks
commit staged modified file
create and switch to branch
switch with changes that conflict
switch leaving changes behind
switch in the middle of merge
undo commit

subjects
11 = 3 industry + 3 research + 5 student
Git: 4 novices, 3 regular, 4 experts
Gitless: none used before
results of a user study

<table>
<thead>
<tr>
<th>I enjoyed using Gitless</th>
<th>9%</th>
<th>14%</th>
<th>14%</th>
<th>27%</th>
<th>27%</th>
<th>9%</th>
<th>64%</th>
</tr>
</thead>
<tbody>
<tr>
<td>I found Gitless to be easier to learn than Git</td>
<td>0%</td>
<td>18%</td>
<td>18%</td>
<td>27%</td>
<td>18%</td>
<td>18%</td>
<td>64%</td>
</tr>
<tr>
<td>I found Gitless to be easier to use than Git</td>
<td>18%</td>
<td>9%</td>
<td>5%</td>
<td>5%</td>
<td>9%</td>
<td>27%</td>
<td>73%</td>
</tr>
<tr>
<td>I would continue using Gitless if I could</td>
<td>36%</td>
<td>18%</td>
<td>5%</td>
<td>5%</td>
<td>9%</td>
<td>27%</td>
<td>55%</td>
</tr>
</tbody>
</table>

Task completion time (minutes)

- Git
- Gitless
git novices

**Bar Chart**

- **Satisfaction**
  - Gitless: 5.09
  - Git: 2.75

- **Efficiency**
  - Gitless: 4.91
  - Git: 3.09

- **Difficulty**
  - Gitless: 4.25
  - Git: 5.5

- **Frustration**
  - Gitless: 3.09
  - Git: 2.91

- **Confusion**
  - Gitless: 4.25
  - Git: 3.73

Legend:
- **Gitless**
- **Git**
conclusion
concepts: the key inventions

Microsoft Word
Paragraph Format Style

Twitter
Tweet Hashtag Following

Photoshop
PixelMap Layer/Mask Adjustment
concepts: define app classes

text editor
line
buffer

word processor
paragraph
format
style

desktop publishing app
stylesheet
text flow
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?
## Concept Catalog (so far)

<table>
<thead>
<tr>
<th>Instantiate</th>
<th>Organize</th>
<th>Relate</th>
<th>Resource</th>
<th>Save</th>
<th>Communicate</th>
<th>Personalize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stylesheet</td>
<td>Selection</td>
<td>Friend</td>
<td>Access Token</td>
<td>History</td>
<td>Message</td>
<td>Account</td>
</tr>
<tr>
<td>Master</td>
<td>Folder</td>
<td>Clique</td>
<td>Notification</td>
<td>Buffer</td>
<td>Posting</td>
<td>Karma</td>
</tr>
<tr>
<td>Stencil</td>
<td>Group</td>
<td>Invitation</td>
<td>Reservation</td>
<td>Cursor</td>
<td>Posting</td>
<td>OoBa</td>
</tr>
<tr>
<td>Style Buffer</td>
<td>Label</td>
<td>Rest</td>
<td>Sync</td>
<td>Export</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td>Cart</td>
<td>Subscription</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack</td>
<td>Alias</td>
<td>Purchase Order</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alias</td>
<td>Preset</td>
<td>RMA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cursor</td>
<td>Filter</td>
<td>Coupon</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td>Property</td>
<td>Catalog</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Metadata</td>
<td>Ticket</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
a common view of software design

UI design
soft & human
about presentation

programming
hard & technical
about content
a better view of software design

conceptual design:
essential concepts & behavior

representation design:
organization & performance