the synergy of AI & modular design

Daniel Jackson · MIT EECS/CSAIL · AI in Software Engineering Forum · 2025

how good are LLMs at coding?

how good are LLMs at generating code?

single functions

46%

of code is autogenerated
when Copilot enabled
(June 2023)

92%

accuracy of coding
GPT-4 + AgentCoder on MBPP
(Jan 2024)

whole apps

If you ask GPT-4 to write whole apps for you, it usually can't get there. But if you architect the app in your head, you can ask it to build it out one function at a time pretty effectively.

HackerNews, May 2024

The skills of the developer will be to figure out, "How small do I have to go until I can leverage AI to synthesize that code for me?"

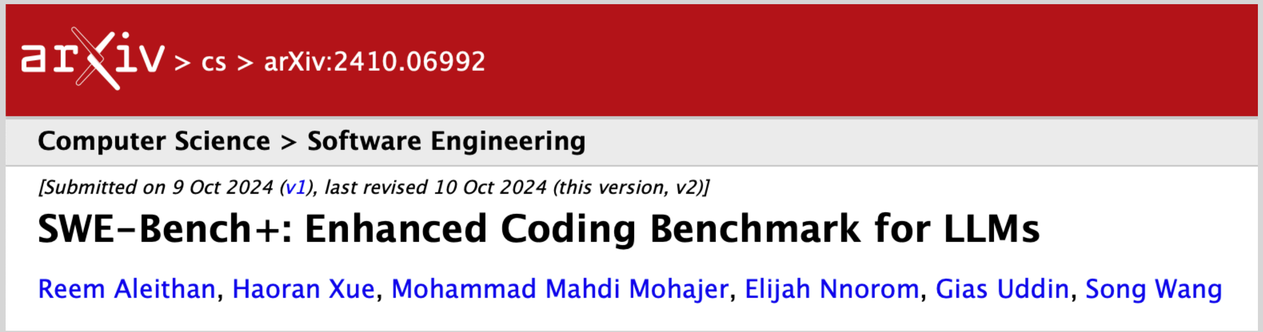
Thomas Dohmke, GitHub CEO, June 2023

what about small code fragments in a more realistic context?

a benchmark for realistic coding problems

2,294 issue/pull request pairs from 12 Python repos

best LLM resolves 65% of issues



follow-up study at York University

33% of good patches "cheated": code appears in issue

31% of patches deemed correct by incomplete tests

94% issues were present before training cutoff

with all this, resolution rate for GPT-4 falls to 0.55%

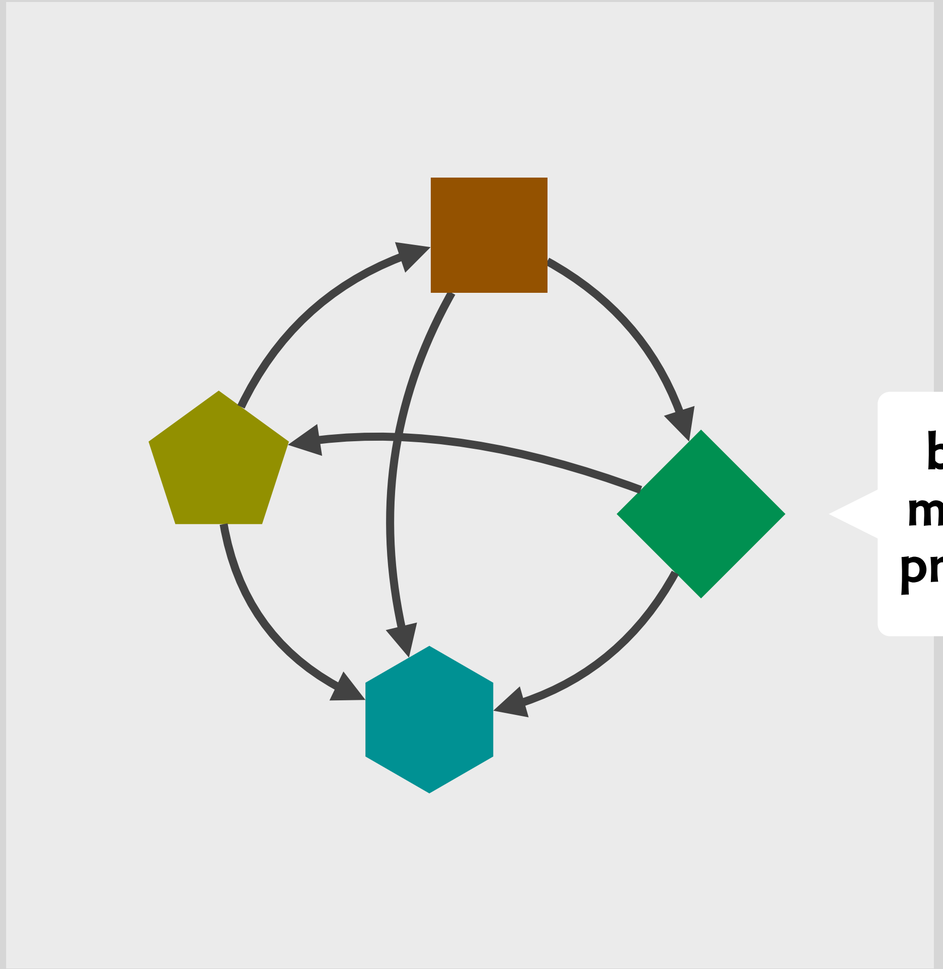
in short: LLM-based coding assistants

often suggest code that doesn't work

and breaks existing functionality

modularity

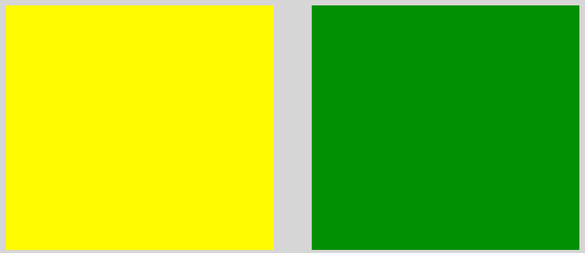
the obvious solution: modularity



build one small module at a time preserving others

what does modularity mean?

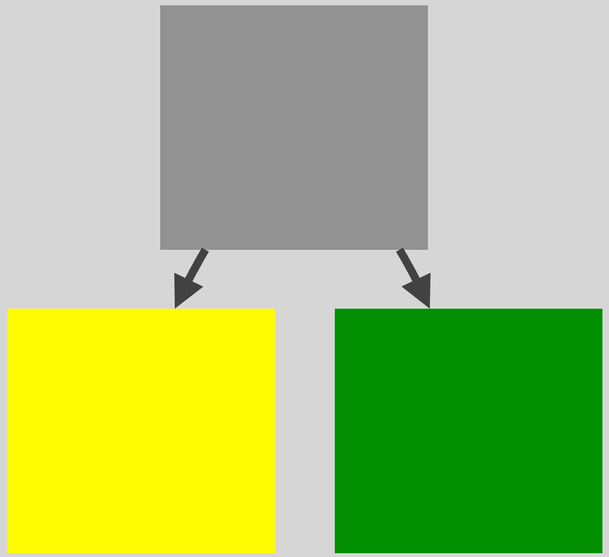
coherence
a single module doesn't
conflate unrelated functions



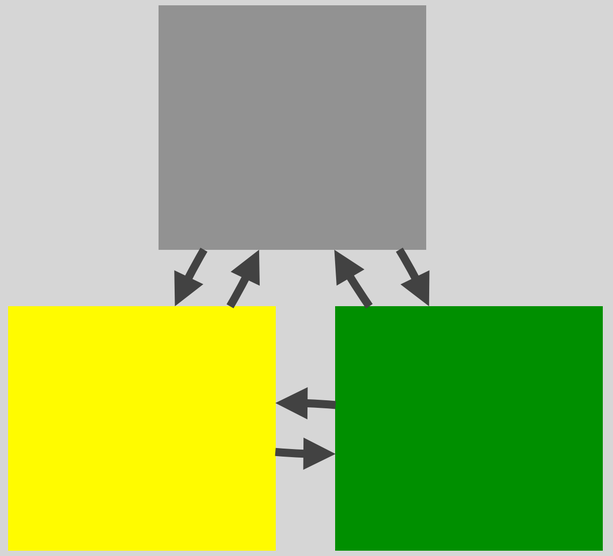
coherent

incoherent

independence
one module doesn't
rely (unduly) on others



independent

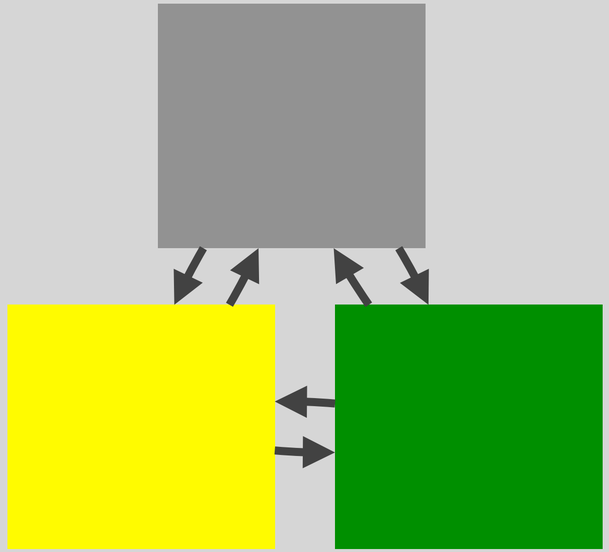


dependent

why modularity matters

incoherence

needless complexity
arising from conflation of functions and purposes



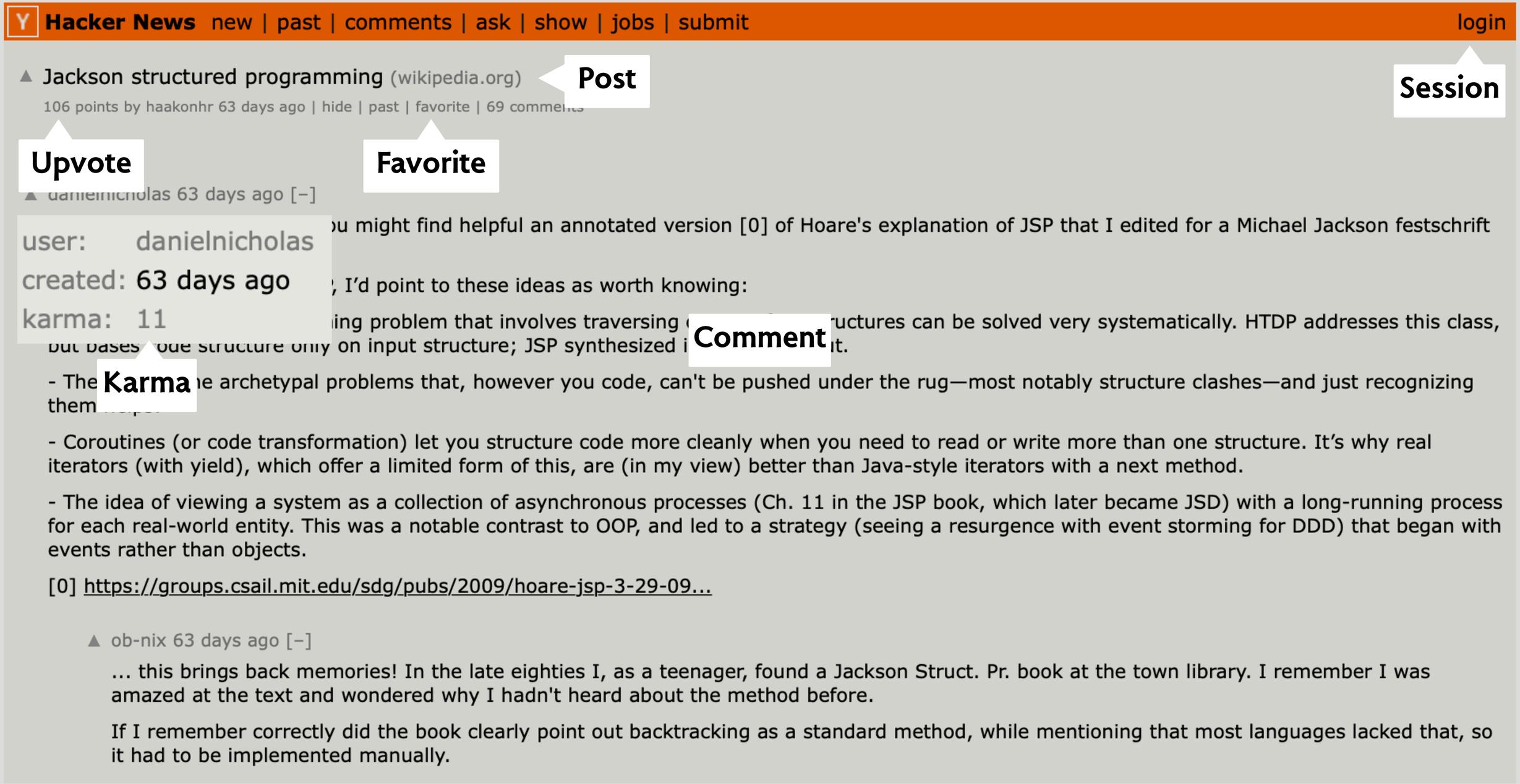
dependence

failure of localization
changes to one module break another; can't reason one at a time

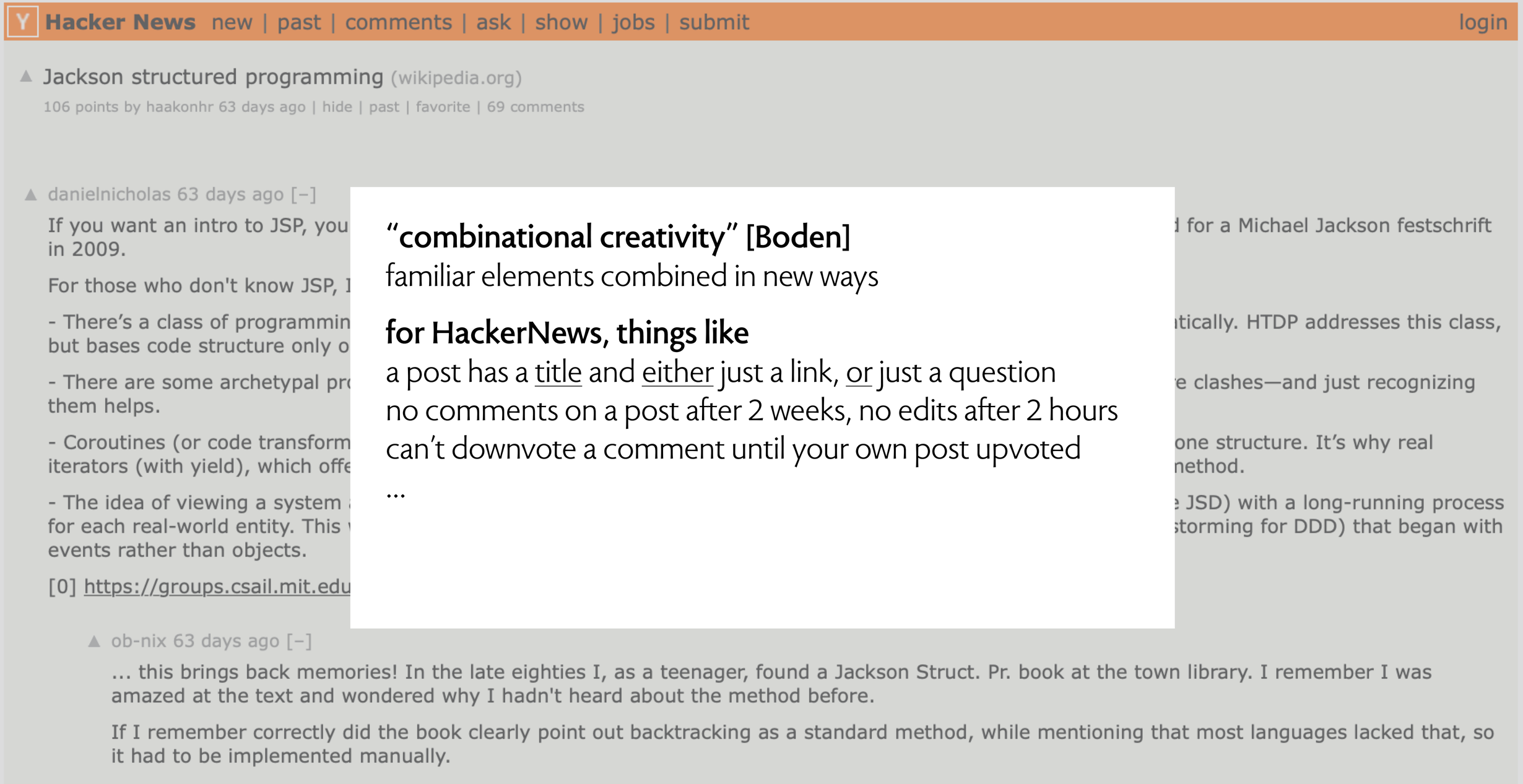
but we tolerated this, because we thought (often wrongly) that these costs outweighed the difficulty of achieving modularity

an example: Hacker News

let's look at an example: hacker news



familiar functions with some creative variation



"combinational creativity" [Boden]

familiar elements combined in new ways

for HackerNews, things like

a post has a title and either just a link, or just a question

no comments on a post after 2 weeks, no edits after 2 hours

can't downvote a comment until your own post upvoted

...

let's code it!

let's build it!

```
class User {
  String name;
  String password;
  User register (n, p) { ... }
  User authenticate (n, p) { ... }
 }
```

```
class Post {
  User author;
  String body;
  Post new (a, b) { ... }
}
```

adding upvoting

```
class User {
  String name;
  String password;
  User register (n, p) { ... }
  User authenticate (n, p) { ... }
}
```

```
class Post {
  User author;
  String body;
  Set [User] ups, downs;
  Post new (a, b) { ... }
  upvote (u) { ... }
  downvote (u) { ... }
}
```

adding karma

```
class User {
  String name;
  String password;
  int karma;
  User register (n, p) { ... }
  User authenticate (n, p) { ... }
  incKarma (i) { ... }
  bool hasKarma (i) { ... }
}
```

```
class Post {
  User author;
  String body;
  Set [User] ups, downs;
  Post new (a, b) { ... }
  upvote (u) { ... }
  downvote (u) {
    if u.hasKarma (10)  ... }
}
```

adding commenting

```
class User {
  String name;
  String password;
  int karma;
  User register (n, p) { … }
  User authenticate (n, p) { … }
  incKarma (i) { … }
  bool hasKarma (i) { … }
}
```

```
class Post {
  User author;
  String body;
  Set [User] ups, downs;
  Seq [Post] comments;
  Post new (a, b) { … }
  upvote (u) { … }
  downvote (u) {
    if u.hasKarma (10)  … }
  addComment (c) { … }
}
```

what's wrong with this code?

```
class User {
  String name;
  String password;
  int karma;
  User register (n, p) { ... }
  User authenticate (n, p) { ... }
  incKarma (i) { ... }
  bool hasKarma (i) { ... }
}
```

```
class Post {
  User author;
  String body;
  Set [User] ups, downs;
  Seq [Post] comments;
  Post new (a, b) { ... }
  upvote (u) { ... }
  downvote (u) {
    if u.hasKarma (10)  ... }
  addComment (c) { ... }
}
```

User authentication

Posting

Upvoting

Commenting

Karma

incoherence
Post class contains posting, commenting, upvoting, karma

→

classes are not reusable
Post class won't work in an app that doesn't have karma points

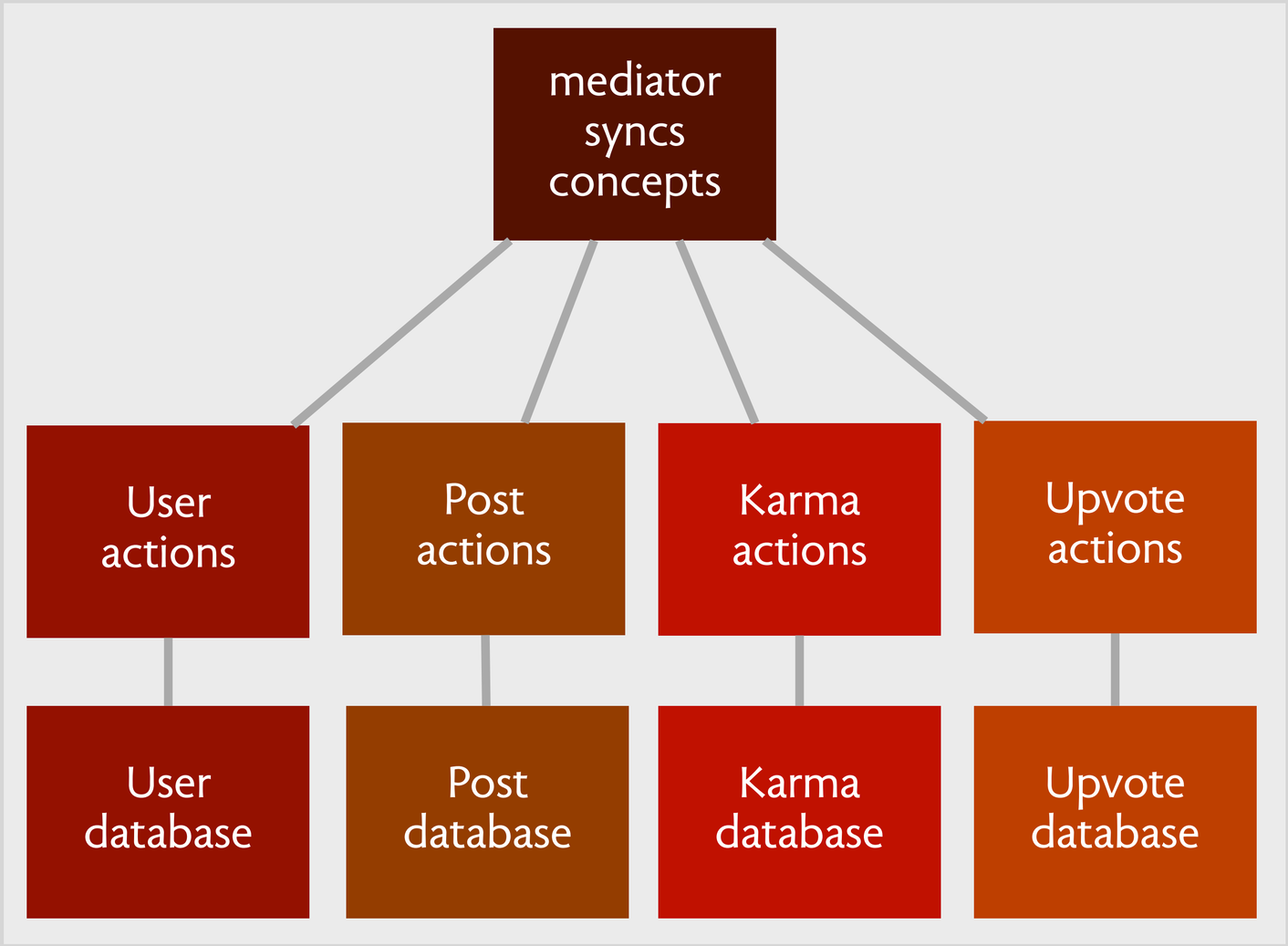
dependence
Post class calls *User* class to get karma points

→

can't be built independently
to build *Post* class, need *User* class to have been built already

a better way

what if we built it like this instead?



concepts: modularizing user-facing functions

```
concept User {
  Map [User, String] name;
  Map [User, String] password;
  User register (n, p) { ... }
  User authenticate (n, p) { ... }
}
```

```
concept Karma [U] {
  Map [U, Int] karma;
  incKarma (u, i) { ... }
  hasKarma (u, i) { ... }
}
```

concerns now cleanly separated

coupling is gone: refs are polymorphic

```
concept Post [U] {
  Map [Post, U] author;
  Map [Post, URL] url;
  Post new (a, u) { ... }
}
```

```
concept Upvote [U, I] {
  Map [U, I] ups, downs;
  upvote (u, i) { ... }
  downvote (u, i) { ... }
}
```

```
concept Comment [U, T] {
  Map [Comment, U] author;
  Map [Comment, T] target;
  Map [Comment, String] body;
  Comment new (a, t, b) { ... }
}
```

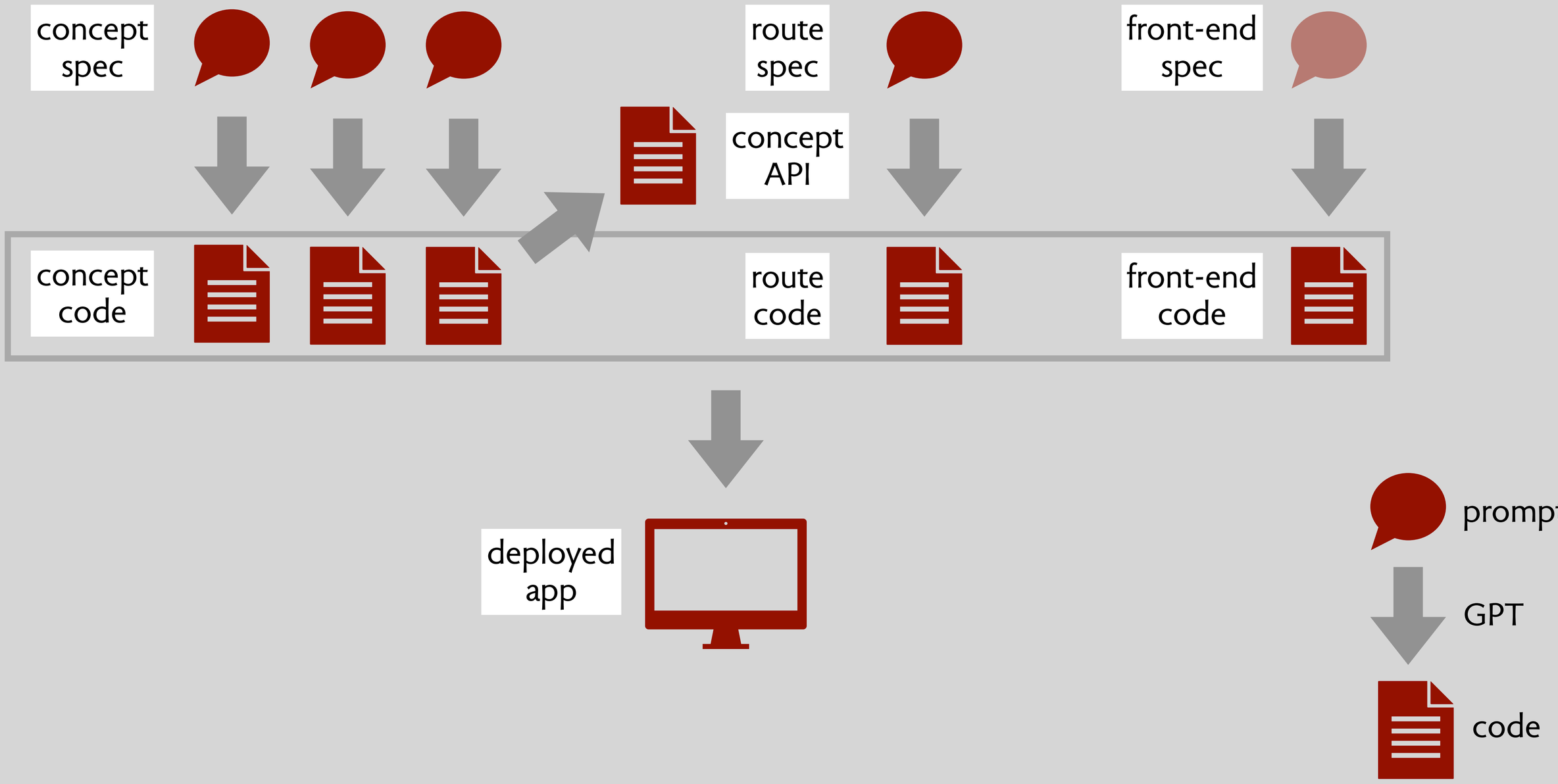
syncs hold cross concept functionality

```
when
  Web.request (downvote, u, i)
  Karma.hasKarma (u, 10)
then
  Upvote.downvote (u, i)
```

Kodless, an LLM code generator

joint work with Abutalib Namazov

exploiting concept modularity to generate code



creating a concept

Create a new concept

Bulk create

Concept name

Karma

Description

Each karma should have user (generic) and points (number). Actions should be: increase(user, x), decrease(user, x), get(user), isAllowed(user, threshold) (throws error if user doesn't have at least threshold)

Create

brief & application independent

first kodless generates the concept code

Source code:

```
1  import { ObjectId } from "mongodb";
2  import DocCollection, { BaseDoc } from "../framework/doc";
3  import { NotAllowedError } from "../framework/errors";
4
5  interface KarmaDoc extends BaseDoc {
6    user: ObjectId;
7    points: number;
8  }
9
10 export default class KarmaConcept {
11   public readonly karmas: DocCollection<KarmaDoc>;
12
13   constructor(collectionName: string) {
14     this.karmas = new DocCollection<KarmaDoc>(collectionName);
15   }
16
17   async increase(user: ObjectId, x: number) {
18     const karma = await this.karmas.readOne({ user });
19     if (karma) {
20       await this.karmas.collection.updateOne({ user }, { $inc: { points: x } });
21     } else {
22       await this.karmas.createOne({ user, points: x });
23     }
```

database schema etc

typical coding details

then it generates an API spec from the code

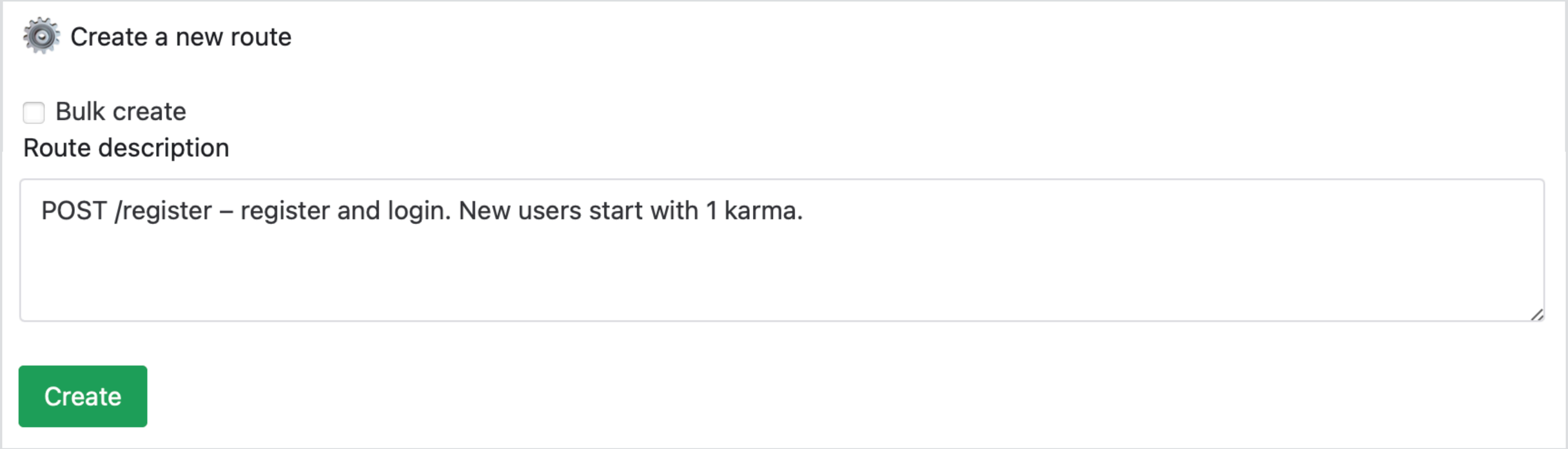
Generated spec:

```
KarmaDoc = {
  user: ObjectId;
  points: number;
}

KarmaConcept = {
  constructor: (collectionName: string) => KarmaConcept; // instantiates a new KarmaConcept with the given collection name
  async increase: (user: ObjectId, x: number) => void; // increases the karma points for a user; creates a new karma record if not exists
  async decrease: (user: ObjectId, x: number) => void; // decreases the karma points for a user; throws NotAllowedError if resulting in negative karma
  async get: (user: ObjectId) => number; // returns the karma p  ser; returns 0 if the user has no karma record
  async isAllowed: (user: ObjectId, threshold: number) => void; // throws NotAllowedError if user's karma points do not meet the threshold
}
```

spec is more detailed than prompt

creating a route



no explicit mention of concepts or their actions

kodless generates the sync (aka route)

Delete

```
@Router.post("/register")
  async createUser(session: WebSessionDoc, username: string, password: string) {
    WebSession.assertLoggedOut(session);
    const { msg, user } = await User.create(username, password);
    await Karma.increase(user._id, 1);
    WebSession.start(session, user._id);
    return { msg };
  }
```

Kodless uses concept APIs to select actions

hacked news: home page

Hacked News new | past | comments | ask | show | jobs | submit login

1. ▲ Show HN: Kodless -- make websites with AI (github.com)
   1 point by dnj a day ago | hide | discuss
2. ▲ Monolith -- CLI tool for saving web pages as a single HTML file (crates.io)
   1 point by rust 9 days ago | hide | discuss
3. ▲ Show HN: Kodless -- make websites with AI (github.com)
   3 points by kodless 9 days ago | hide | 2 comments
4. ▲ Software = concepts (essenceofsoftware.com)
   6 points by dnj 9 days ago | hide | 2 comments
5. ▲ Jobs HN: I am hiring a wine expert
   1 point by recruiter 10 days ago | hide | discuss
6. ▲ Show HN: Fuiz -- free, open-source and privacy-friendly alternative to Kahoot (fuiz.us)
   3 points by best_dev 10 days ago | hide | 7 comments
7. ▲ Ask HN: How many bugs do you have per line of code?
   1 point by big_asker 10 days ago | hide | discuss
8. ▲ Github -- Use this website to share your code (github.com)
   2 points by barish2 10 days ago | hide | discuss

Guidelines | FAQ | Lists | API | Security | Legal | Apply to YC | Contact

Search:

hacked news: thread

Hacked News new | threads | past | comments | ask | show | jobs | submit kodless (3) | logout

Software = concepts (essenceofsoftware.com)
6 points by dnj 9 days ago | unvote | hide | favorite | 2 comments

add comment

* kodless 9 days ago | next [–]

dnj, you should check out my platform Kodless -- it helps you generate software with concepts without writing any code.

reply

dnj 9 days ago | next [–] unvote

That's great to see! Have you seen GPT-powered concept tutor? https://essenceofsoftware.com/studies/larger/tutor/

reply

Guidelines | FAQ | Lists | API | Security | Legal | Apply to YC | Contact

Search:

hacked news: user profile

Hacked News new | threads | past | comments | ask | show | jobs | submit kodless (3) | logout

user: kodless
created: 2024-03-25T19:14:36.316Z
karma: 3
about:

submissions
threads
favorites
hidden

change top bar color

Guidelines | FAQ | Lists | API | Security | Legal | Apply to YC | Contact

Search:

hacked news: posts by date

Hacked News new | threads | past | comments | ask | show | jobs | submit kodless (3) | logout

previous day | next day | 2024-03-25

1. Software = concepts (essenceofsoftware.com)
   6 points by dnj 9 days ago | unvote | hide | 2 comments
2. * Show HN: Kodless -- make websites with AI (github.com)
   3 points by kodless 9 days ago| hide | 2 comments
3. Show HN: Fuiz -- free, open-source and privacy-friendly alternative to Kahoot (fuiz.us)
   3 points by best_dev 10 days ago | unvote | hide | 7 comments
4. Github -- Use this website to share your code (github.com)
   2 points by barish2 10 days ago | unvote | hide | discuss
5. ▲ Ask HN: How many bugs do you have per line of code?
   1 point by big_asker 10 days ago| hide | discuss

Guidelines | FAQ | Lists | API | Security | Legal | Apply to YC | Contact

Search:

successes & challenges

coverage of all major functionality
flagging, hiding, karma, etc
nested comments, complicated ranking rules
many application-specific details (eg, karma)

minimal prompting required
< 100 lines of prompts

no code editing needed
some prompt revisions to tweak behavior

backend code only
frontend hard because of overlapping concepts

some syncs are ugly
especially packaging objects for the endpoint

need concept actions for queries
eg, get_comments_by_target

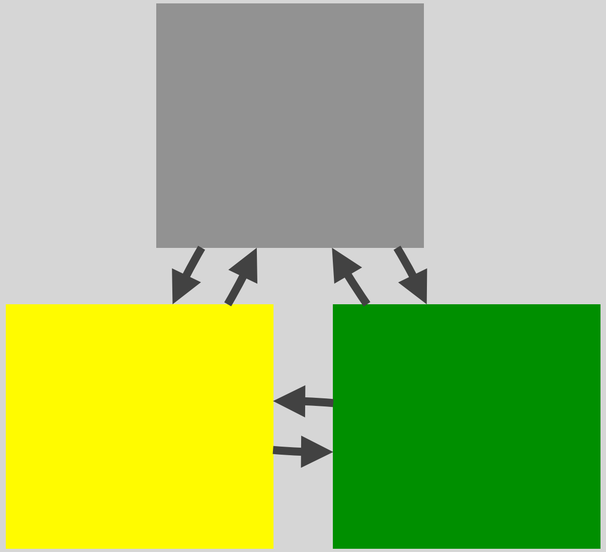
conclusions

why modularity matters **for LLM code generation**

incoherence

needless complexity
arising from conflation of functions and purposes

modules are app-specific
novel combinations of functions



dependence

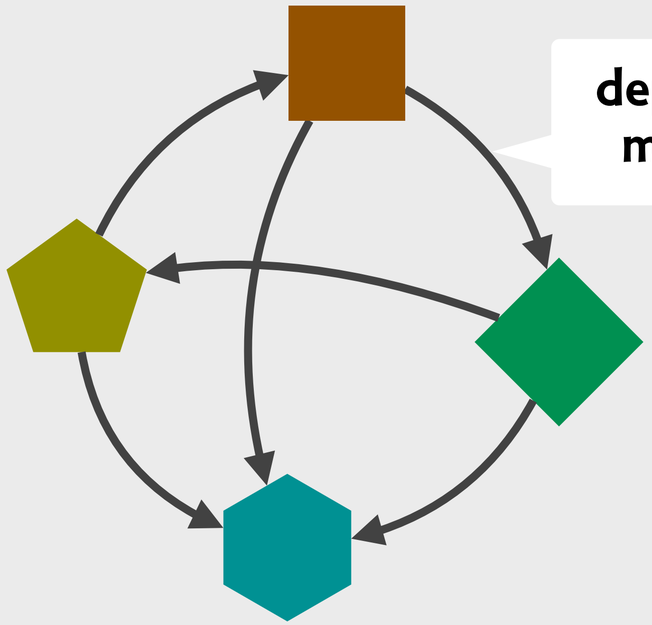
failure of localization
changes to one module break another; can't reason one at a time

context isn't bounded
need whole codebase or complete interfaces

but we tolerated this, because we thought (often wrongly) that these costs outweighed the difficulty of achieving modularity

modularity is the very key to code generation with LLMs

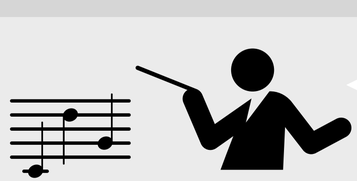
summary: a new kind of modularity



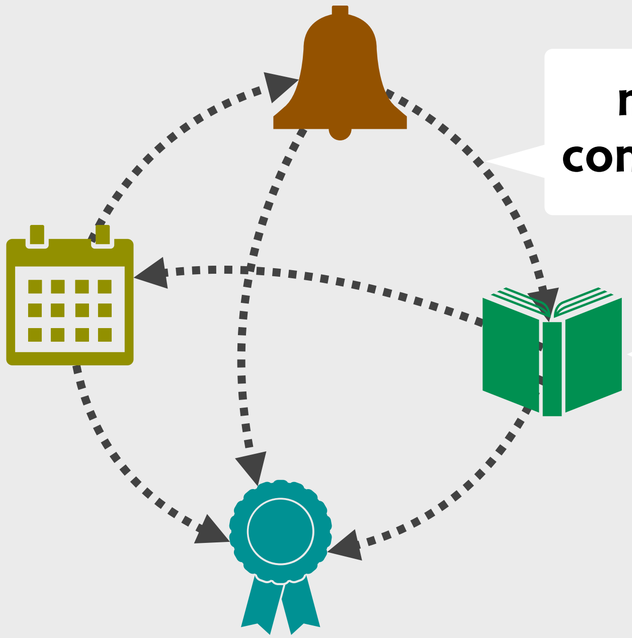
dependent modules

incoherent modules

app-specific details



conductor coordinates



mediated control & data

familiar coherent modules

icons by Luis Prado & Zach Bogart, Noun Project

what we're working on now

synchronizations
fully declarative (order independent)
richly expressive and incremental
in an LLM-familiar style (eg, Datalog)
factor out presentation, persistence & more!

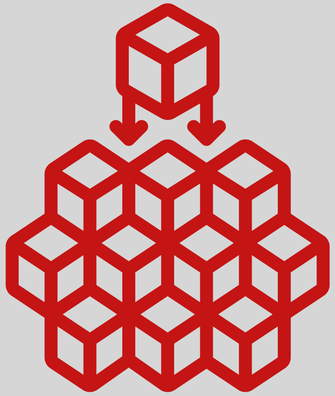
front-end generation
using concepts to structure

a library of concepts
can be reused without regeneration
can be audited for security & correctness

the benefits concepts bring

initial motivations

better UX
clarity & power



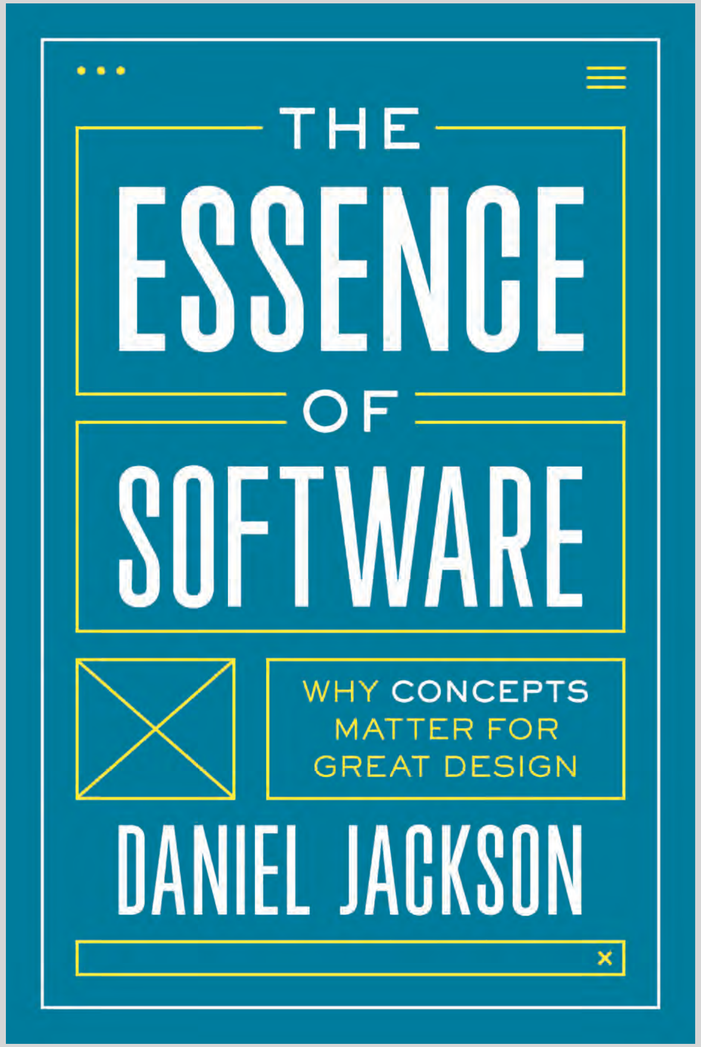
modularity
in design & code

a design language
bridging roles too

a place for design
concept-specific issues

what may matter as much or more

perspective: where this all came from



https://essenceofsoftware.com

origins of this project

analysis of >100 apps over 10 years
trying to pinpoint causes of UX success & failure
seeking a way to analyze & structure function

what I found

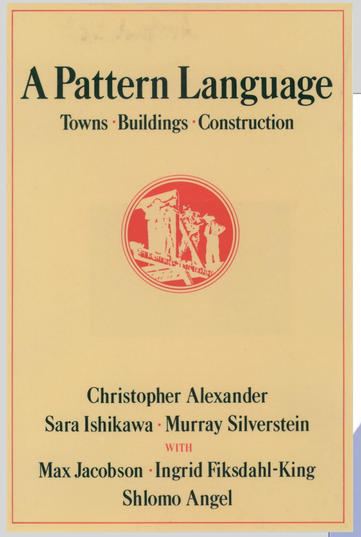
all agree conceptual models are essential
but little research on how to make them modular

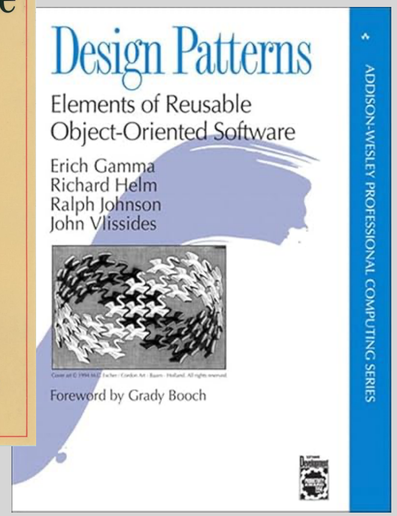
with concepts

explain complex designs & diagnose flaws
bring alignment & strategic clarity to large-scale devs
provide a strong basis for LLM code generation

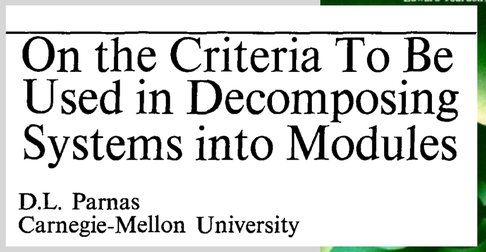
backup slides

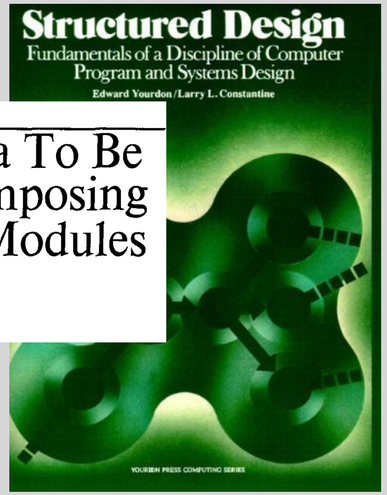
where the ideas of concept design came from



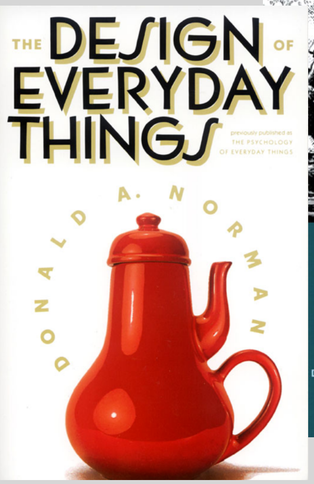


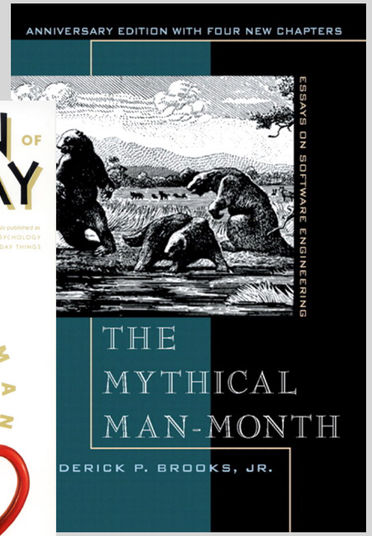
Christopher Alexander's **patterns**
popularized in software by GoF
source of DDD's ubiquitous language?



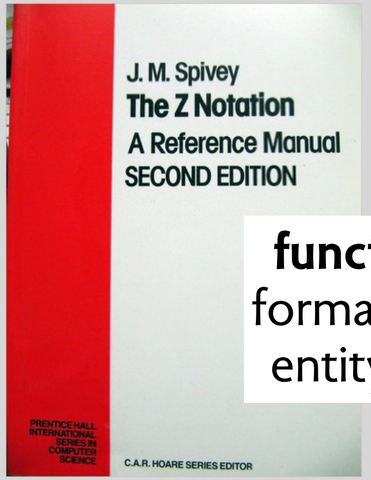


modularity & encapsulation
Parnas: dependencies & design secrets
Yourdon/Constantine: coupling & cohesion





conceptual models
user-centered computing at PARC
Brooks's essence & accident



function as actions on states
formal methods (eg, Z, VDM, B)
entity-relationship data model

a sample concept: upvote

concept Upvote

purpose rank items by popularity

state
by: Vote -> **one** User
for: Vote -> **one** Item
Upvote, Downvote: **set** Vote
rank: Item -> **one** Int

actions
upvote (u: User, i: Item)
downvote (u: User, i: Item)
unvote (u: User, i: Item)