

# the synergy of AI & modular design

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?



## SWE-bench

Can Language Models Resolve Real-World GitHub Issues?

ICLR 2024

Carlos E. Jimenez\*, John Yang\*,  
Alexander Wettig, Shunyu Yao, Kexin Pei,  
Ofir Press, Karthik Narasimhan

### a benchmark for realistic coding problems

2,294 issue/pull request pairs from 12 Python repos  
best LLM resolves 65% of issues

arXiv > cs > arXiv:2410.06992

Computer Science > Software Engineering

[Submitted on 9 Oct 2024 (v1), last revised 10 Oct 2024 (this version, v2)]

### SWE-Bench+: Enhanced Coding Benchmark for LLMs

Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, Song Wang

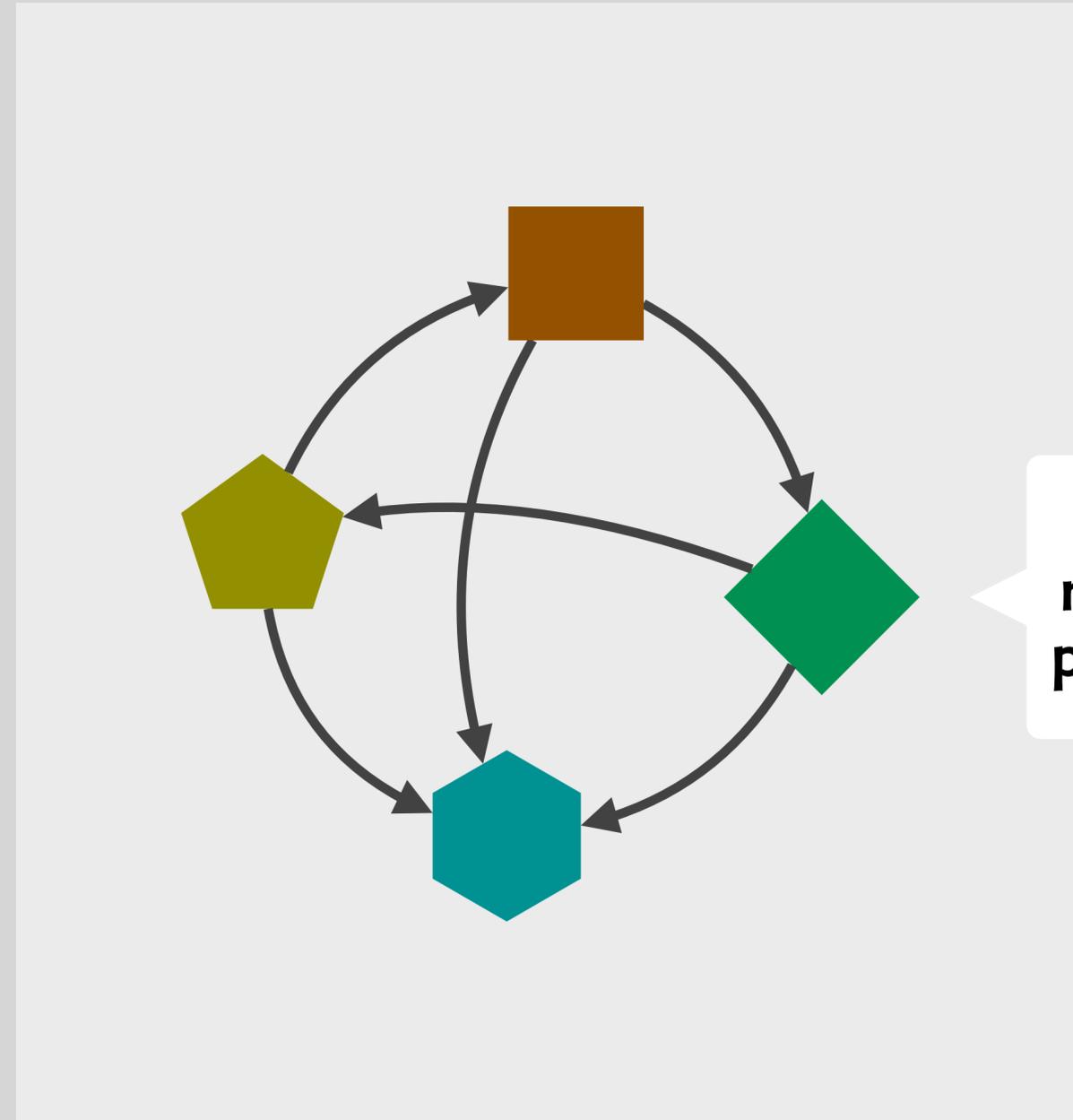
### follow-up study at York University

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

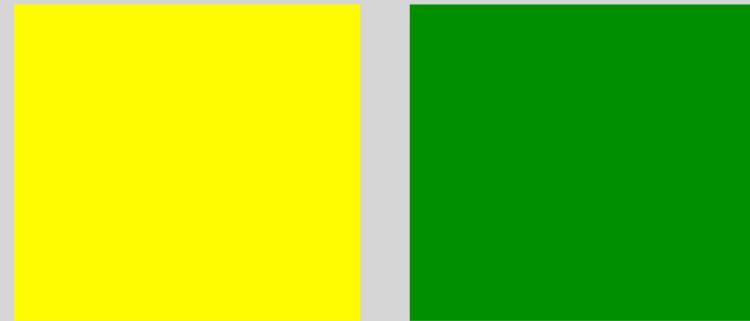


**build one small  
module at a time  
preserving others**

# what does modularity mean?

## coherence

a single module doesn't conflate unrelated functions



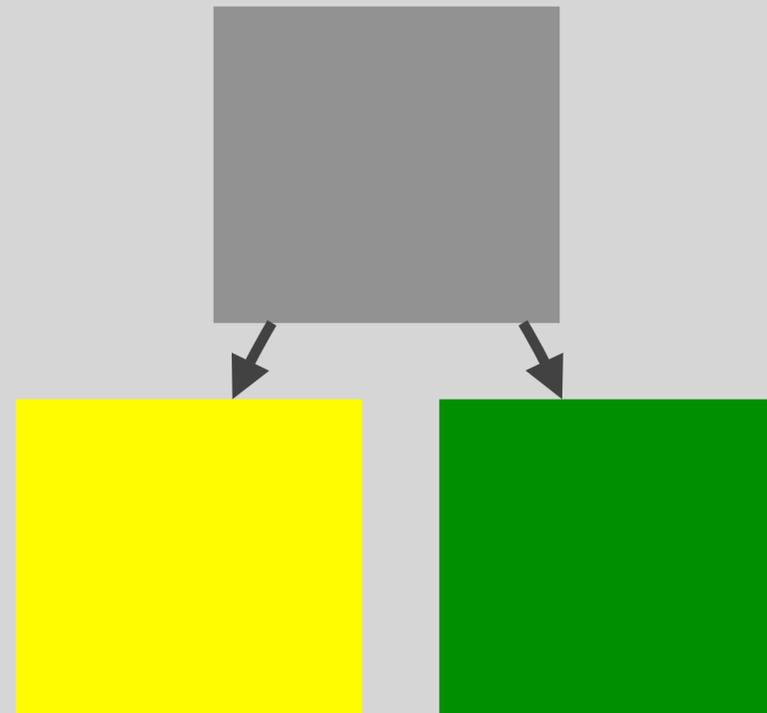
*coherent*



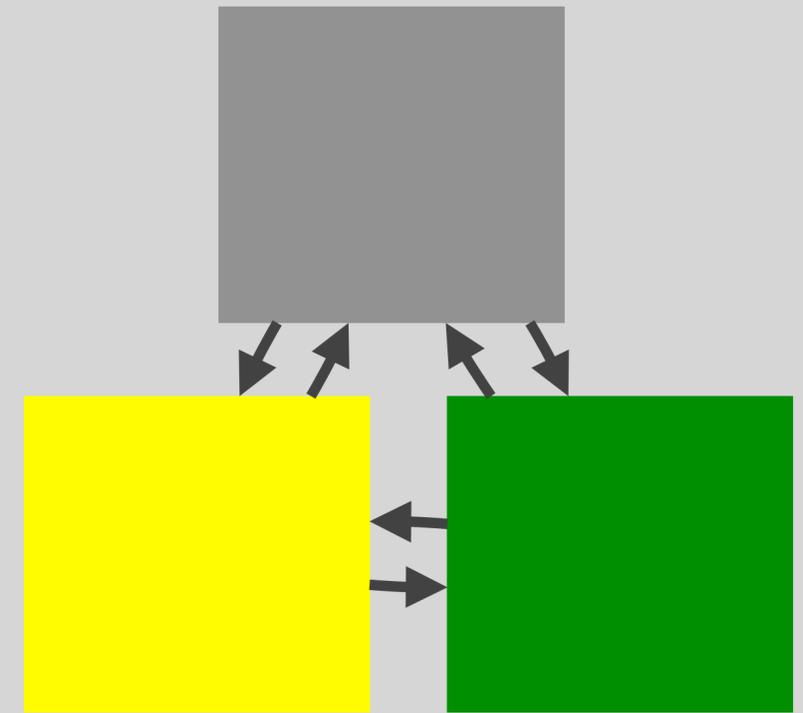
*incoherent*

## independence

one module doesn't rely (unduly) on others

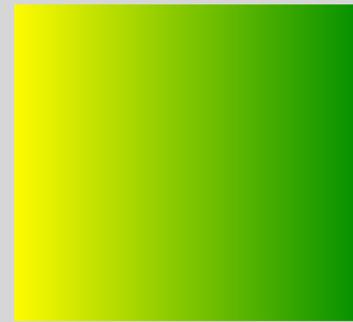


*independent*



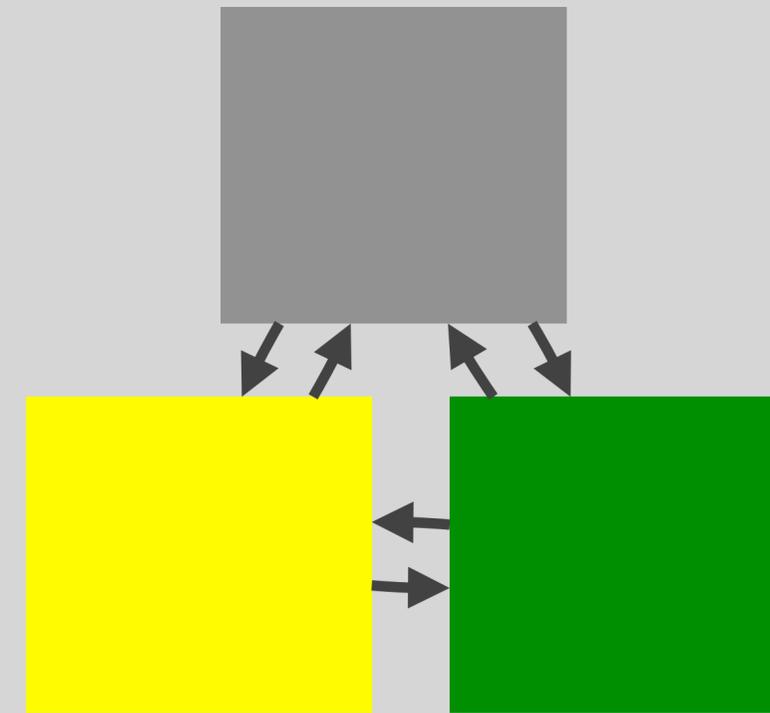
*dependent*

# 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

Y **Hacker News** new | past | comments | ask | show | jobs | submit

login

▲ Jackson structured programming (wikipedia.org)

Post

Session

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

Upvote

Favorite

▲ danielnicholas 63 days ago [-]

user: danielnicholas

created: 63 days ago

karma: 11

Comment

you might find helpful an annotated version [0] of Hoare's explanation of JSP that I edited for a Michael Jackson festschrift  
, I'd point to these ideas as worth knowing:  
ing problem that involves traversing structures can be solved very systematically. HTDP addresses this class,  
but bases code structure only on input structure; JSP synthesized it.

- The archetypal problems that, however you code, can't be pushed under the rug—most notably structure clashes—and just recognizing them

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators (with yield), which offer a limited form of this, are (in my view) better than Java-style iterators with a next method.

- The idea of viewing a system as a collection of asynchronous processes (Ch. 11 in the JSP book, which later became JSD) with a long-running process for each real-world entity. This was a notable contrast to OOP, and led to a strategy (seeing a resurgence with event storming for DDD) that began with events rather than objects.

[0] <https://groups.csail.mit.edu/sdg/pubs/2009/hoare-jsp-3-29-09...>

▲ ob-nix 63 days ago [-]

... this brings back memories! In the late eighties I, as a teenager, found a Jackson Struct. Pr. book at the town library. I remember I was amazed at the text and wondered why I hadn't heard about the method before.

If I remember correctly did the book clearly point out backtracking as a standard method, while mentioning that most languages lacked that, so it had to be implemented manually.

# familiar functions with some creative variation

Y **Hacker News** new | past | comments | ask | show | jobs | submit

login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [-]

If you want an intro to JSP, you can find one in 2009.

For those who don't know JSP, I

- There's a class of programming languages that are syntactically similar to Java but bases code structure only on classes.

- There are some archetypal programming patterns that knowing them helps.

- Coroutines (or code transformers) and iterators (with yield), which offer a way to write code that is more declarative.

- The idea of viewing a system as a collection of events rather than objects. This is a common pattern for each real-world entity. This is a common pattern for events rather than objects.

[0] <https://groups.csail.mit.edu>

▲ ob-nix 63 days ago [-]

... this brings back memories! In the late eighties I, as a teenager, found a Jackson Struct. Pr. book at the town library. I remember I was amazed at the text and wondered why I hadn't heard about the method before.

If I remember correctly did the book clearly point out backtracking as a standard method, while mentioning that most languages lacked that, so it had to be implemented manually.

**“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

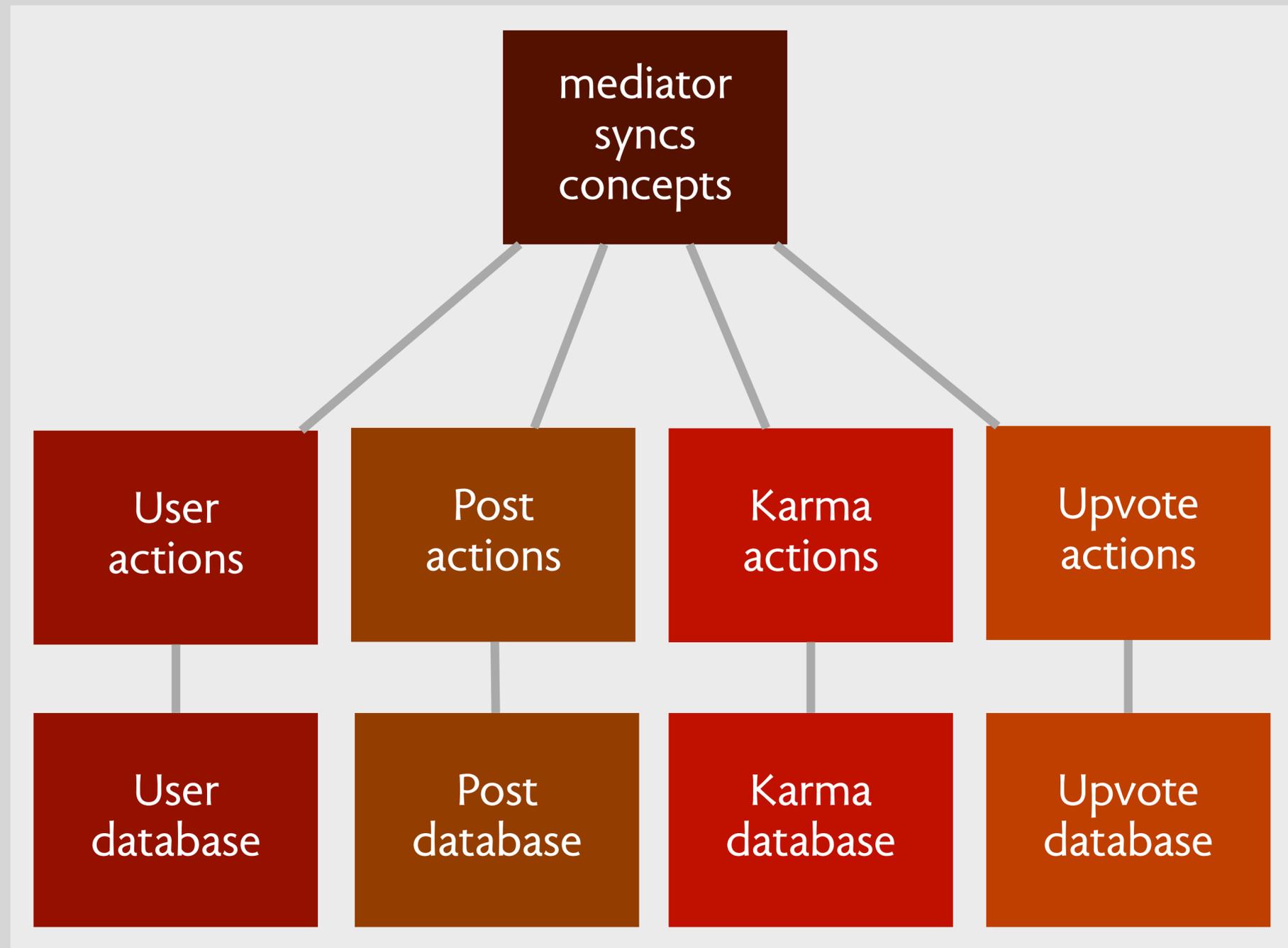
*Post* class calls *User* class to get karma points



**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?



no dependencies between concepts!

# 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) { ... }  
}
```

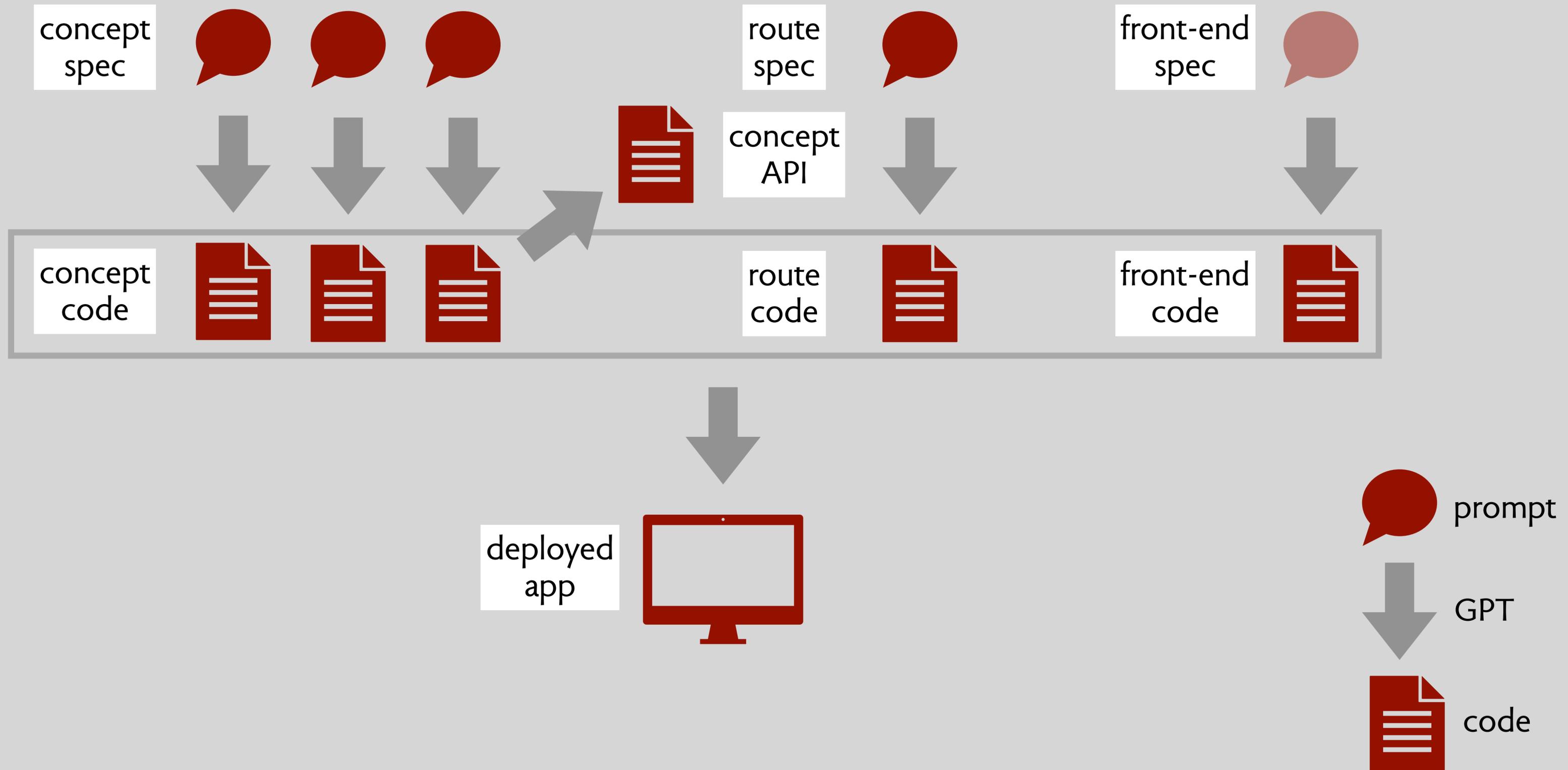
syncs hold  
cross concept  
functionality

```
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     }
24   }
25 }
```

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; cre  
  ates a new karma record if not exists  
  async decrease: (user: ObjectId, x: number) => void; // decreases the karma points for a user; thr  
  ows NotAllowedError if resulting in negative karma  
  async get: (user: ObjectId) => number; // returns the karma p user; returns 0 if the us  
  er 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

 Create a new route

Bulk create

Route description

POST /register – register and login. New users start with 1 karma.

Create

**no explicit mention  
of concepts or their  
actions**

# kodless generates the sync (aka route)

Delete

```
1 @Router.post("/register")
2   async createUser(session: WebSessionDoc, username: string, password: string) {
3     WebSession.assertLoggedOut(session);
4     const { msg, user } = await User.create(username, password);
5     await Karma.increase(user._id, 1);
6     WebSession.start(session, user._id);
7     return { msg };
8   }
```

**Kodless  
uses concept APIs  
to select actions**

# hacked news: home page

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](#)

# hacked news: thread

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](#)

Search:

# hacked news: user profile

user: kodless

created: 2024-03-25T19:14:36.316Z

karma: 3

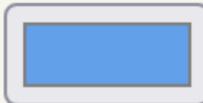
about:

[submissions](#)

[threads](#)

[favorites](#)

[hidden](#)



change top bar color

Search:

# hacked news: posts by date

[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**

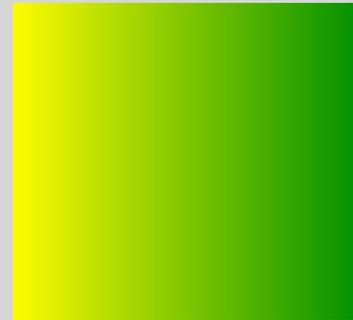
especially packaging objects for the endpoint

## **need concept actions for queries**

eg, `get_comments_by_target`

conclusions

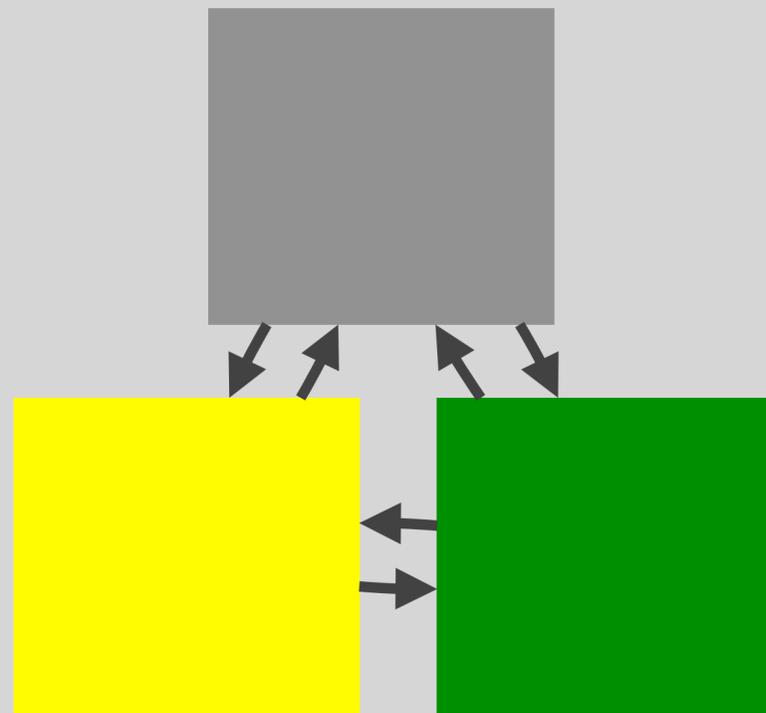
# why modularity matters for LLM code generation



incoherence

**needless complexity**  
arising from conflation of  
functions and purposes

**modules are app-specific**  
novel combinations  
of functions



dependence

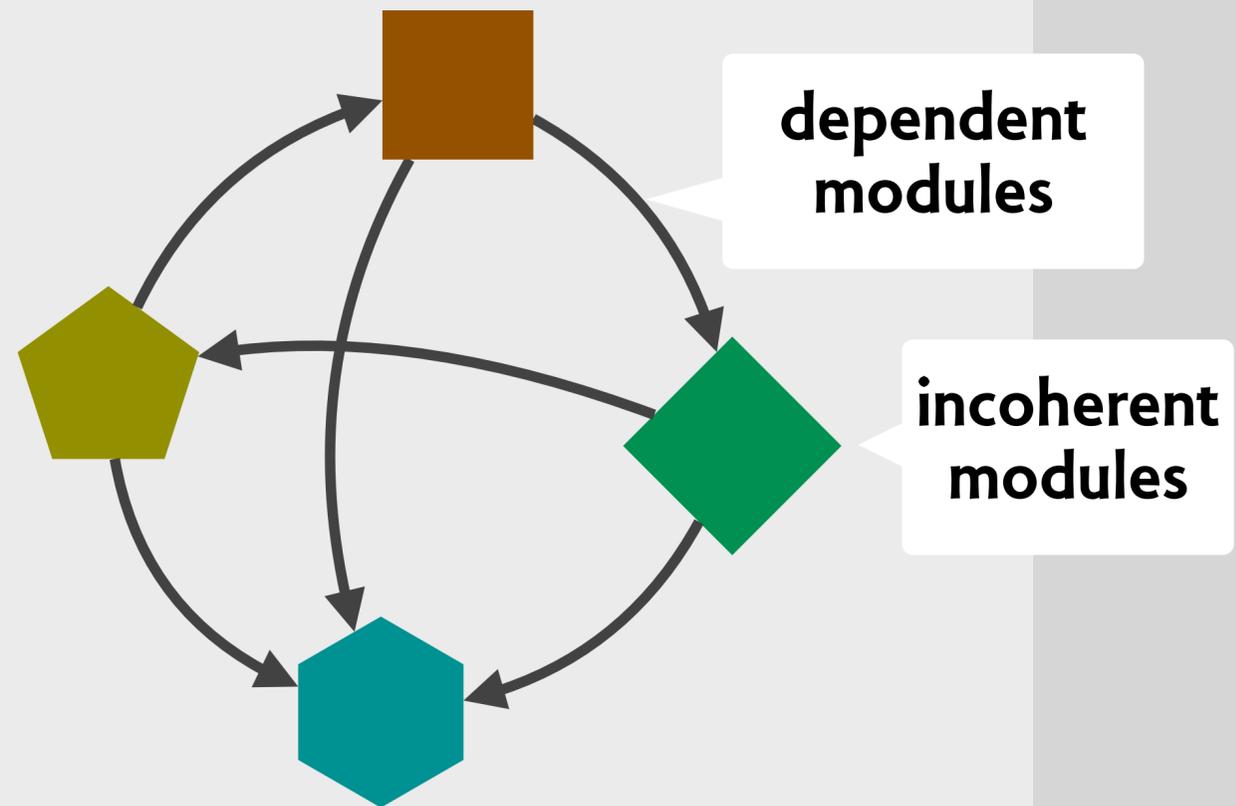
**failure of localization**  
changes to one module  
break another; can't  
reason one at a time

**context isn't bounded**  
need whole codebase  
or complete interfaces

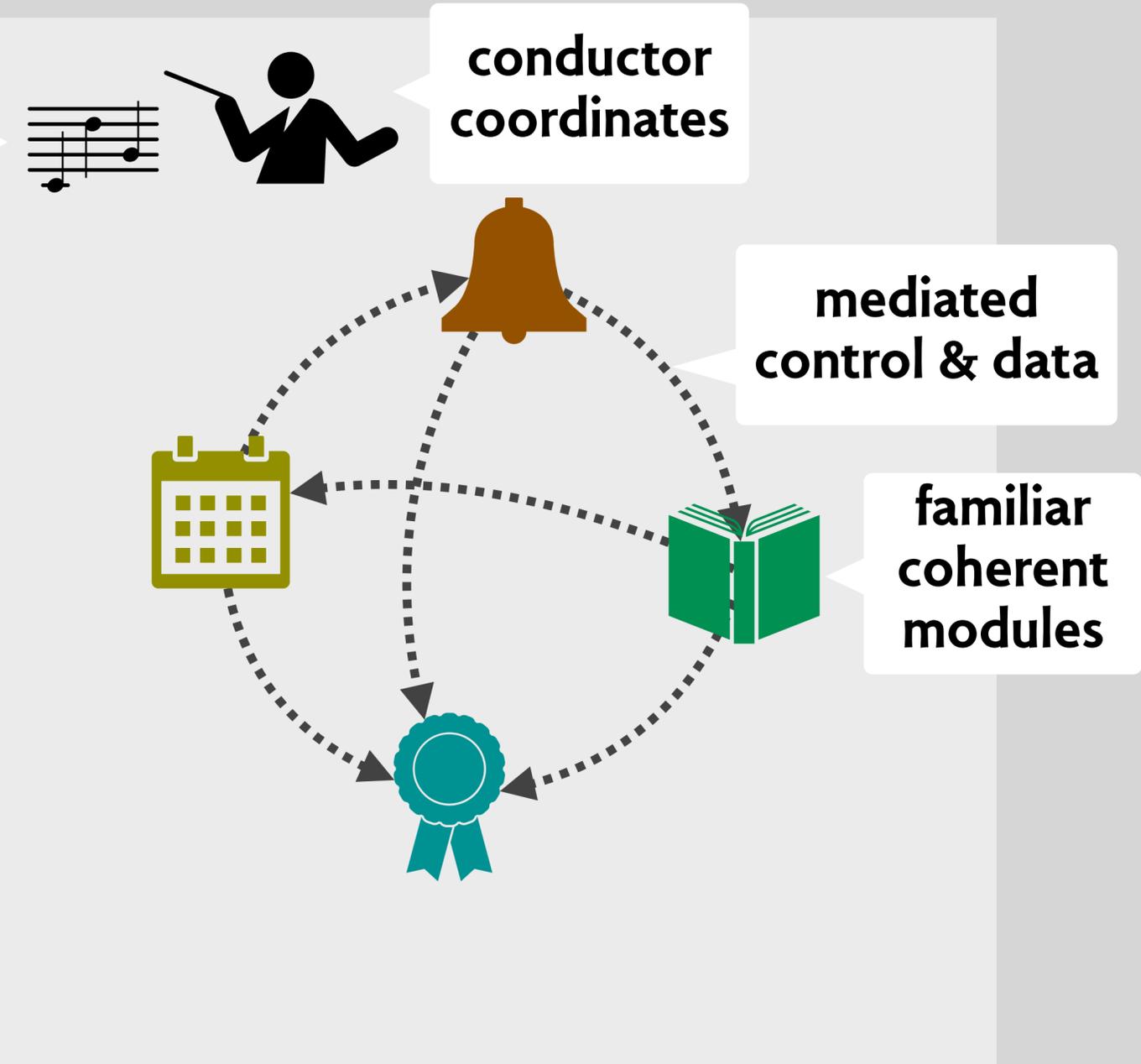
but we tolerated this, because we thought (often wrongly) that these costs outweighed the difficulty of achieving modularity

**modularity is the very key to code generation with LLMs**

# summary: a new kind of modularity



app-specific details



# 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**

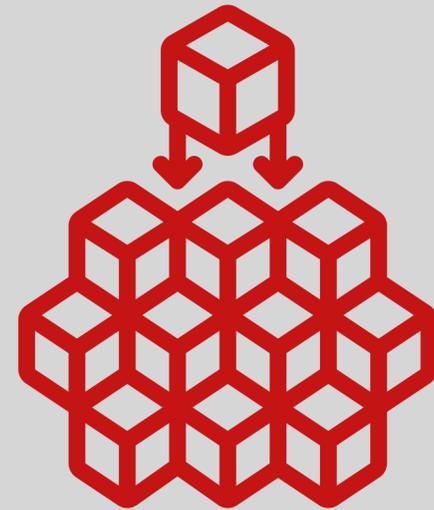
can be reused without regeneration  
can be audited for security & correctness

# the benefits concepts bring

## initial motivations



**better UX**  
clarity & power



**modularity**  
in design & code



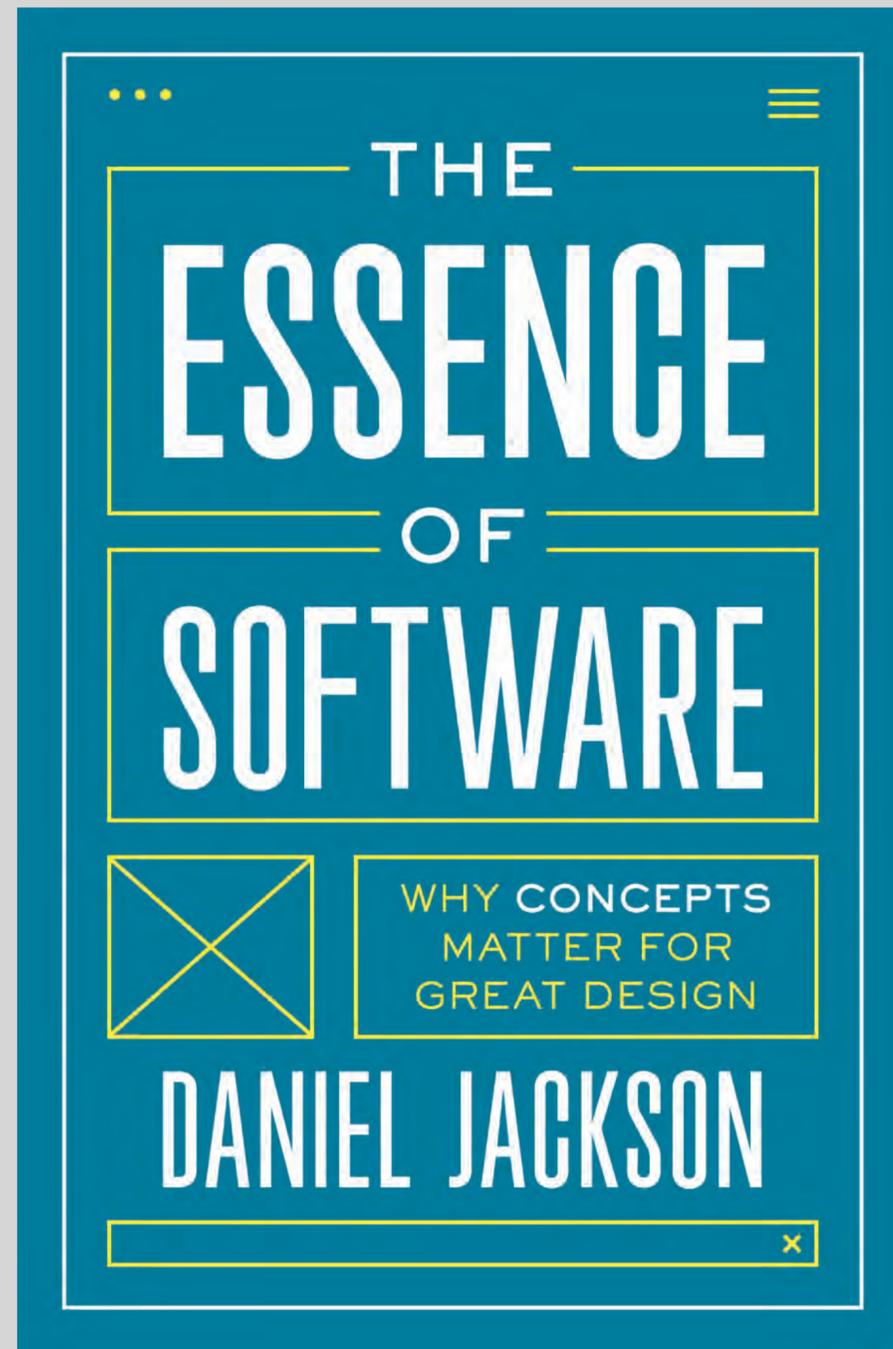
**a design language**  
bridging roles too



**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**

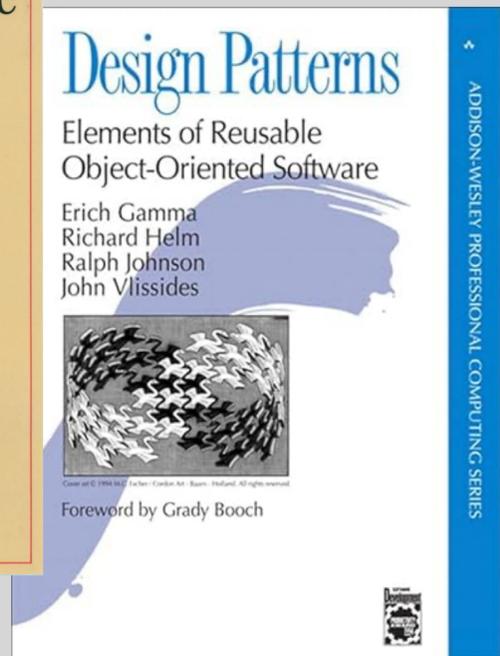
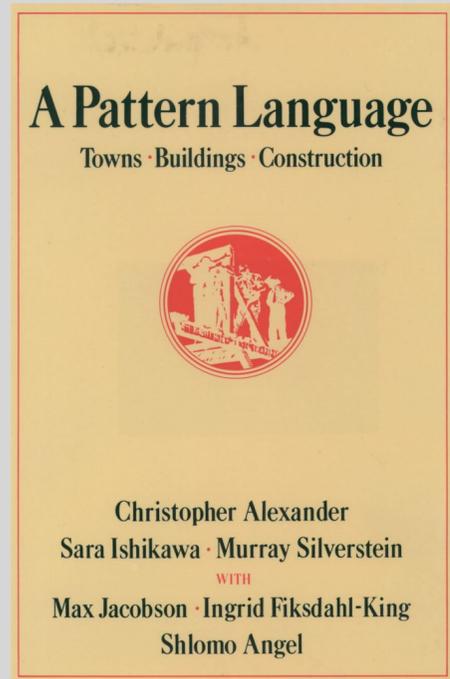
all agree conceptual models are essential  
but little research on how to make them modular

### **with concepts**

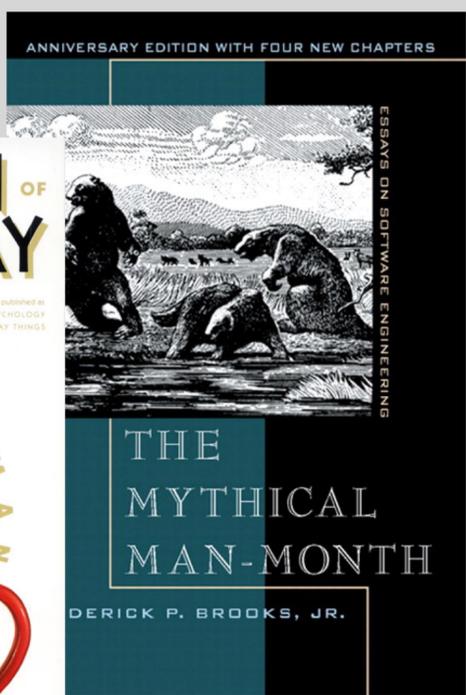
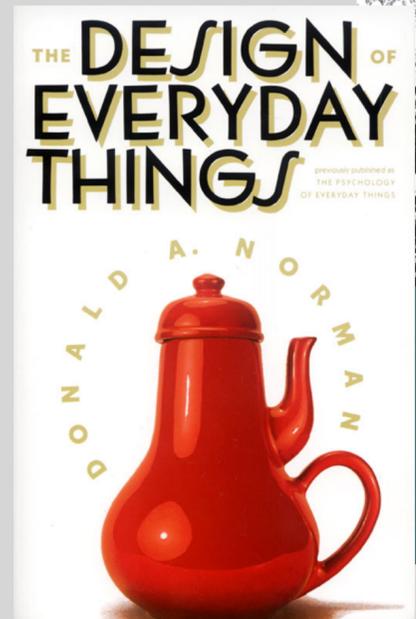
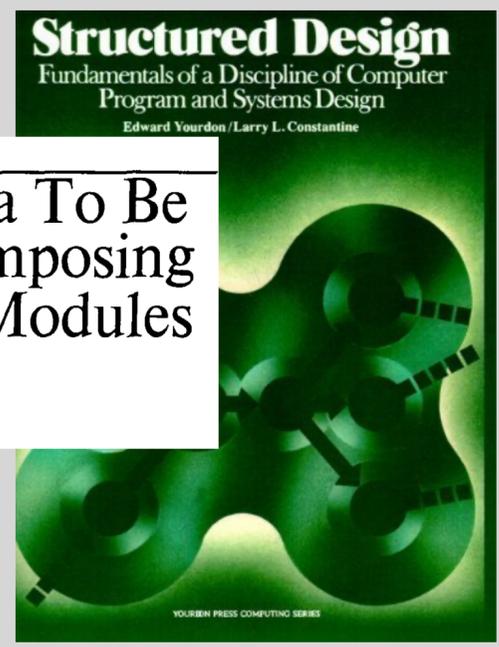
explain complex designs & diagnose flaws  
bring alignment & strategic clarity to large-scale devs  
provide a strong basis for LLM code generation

backup slides

# where the ideas of concept design came from



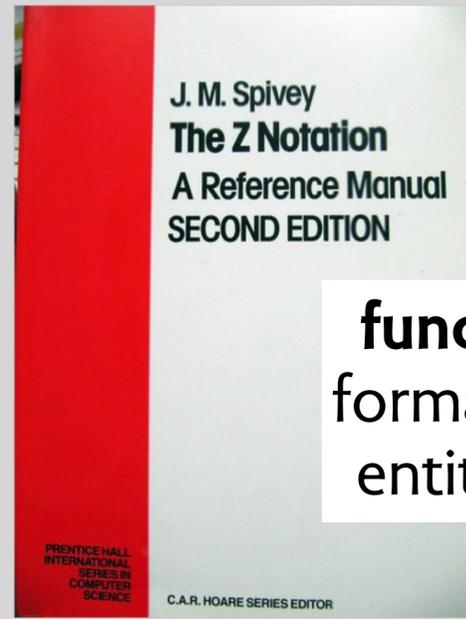
**On the Criteria To Be Used in Decomposing Systems into Modules**  
D.L. Parnas  
Carnegie-Mellon University



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)

