AI Proof Your Career With Software Architecture

Kelly Morrison

Agenda


• Introduction
• GPT, Copilot, CodeWhisperer
• Example low and high level prompts
• Issues with AI code generation
• Junior developer guidance
• Senior developer guidance
• Learning architecture
• ArchUnit
• Questions

Kelly Morrison, Ph.D. Computer Engineering

Solution Architect - Daugherty Business Solutions

The University of Georgia

TEAM Daugherty Est. 1985

CNN

THE HOME DEPOT

COLORBUS

AUBURN UNIVERSITY

Turner

DIGITAL output MAGAZINE

PATHFIRE Enabling Digital Media

THE CLIMATE CORPORATION

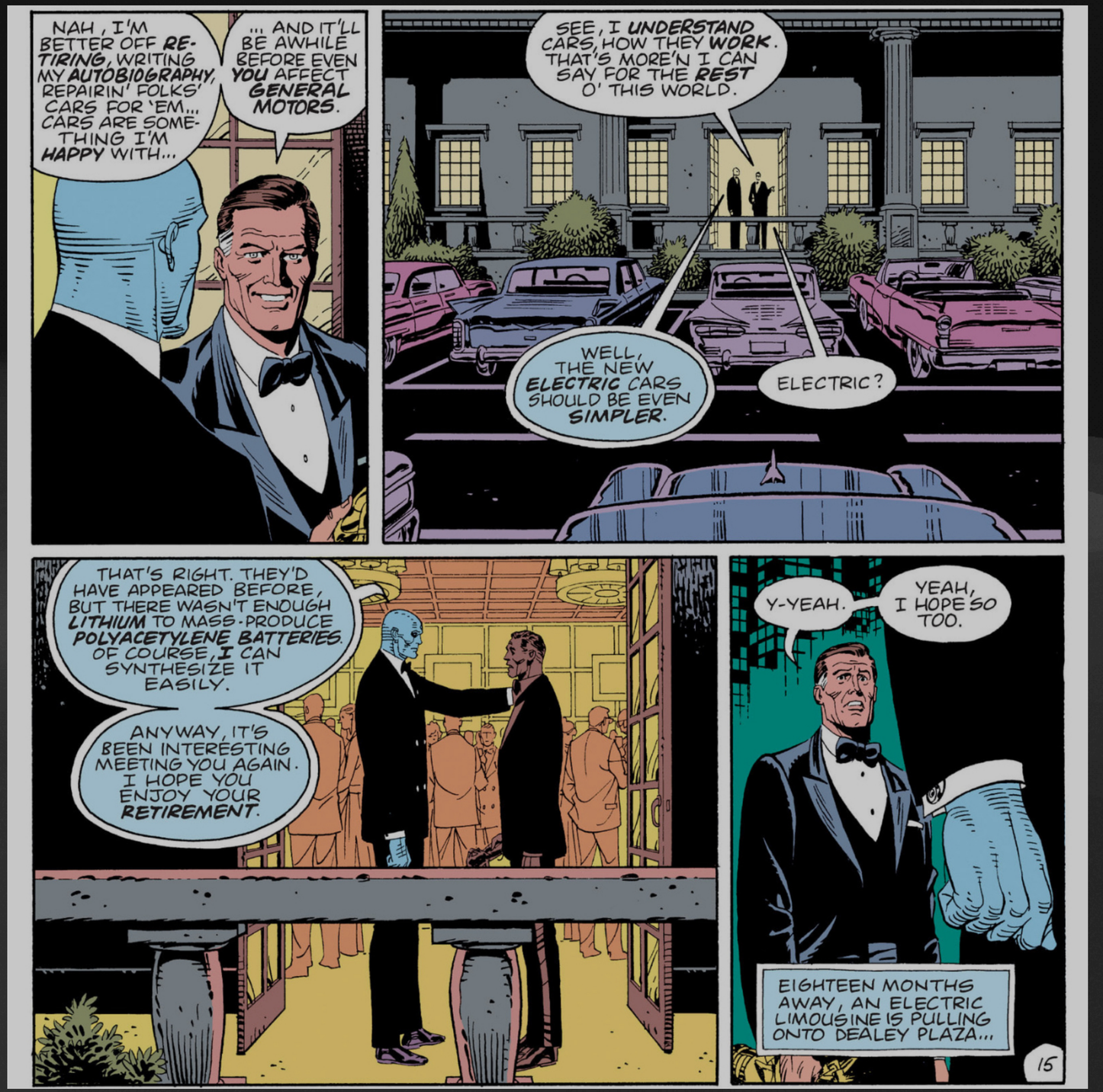
NASA Marshall Space Flight Center

Performance MATTERS


ROCK 'EM
SOCK 'EM
ROBOTS BY MARX

Watchmen

Alan Moore
1986



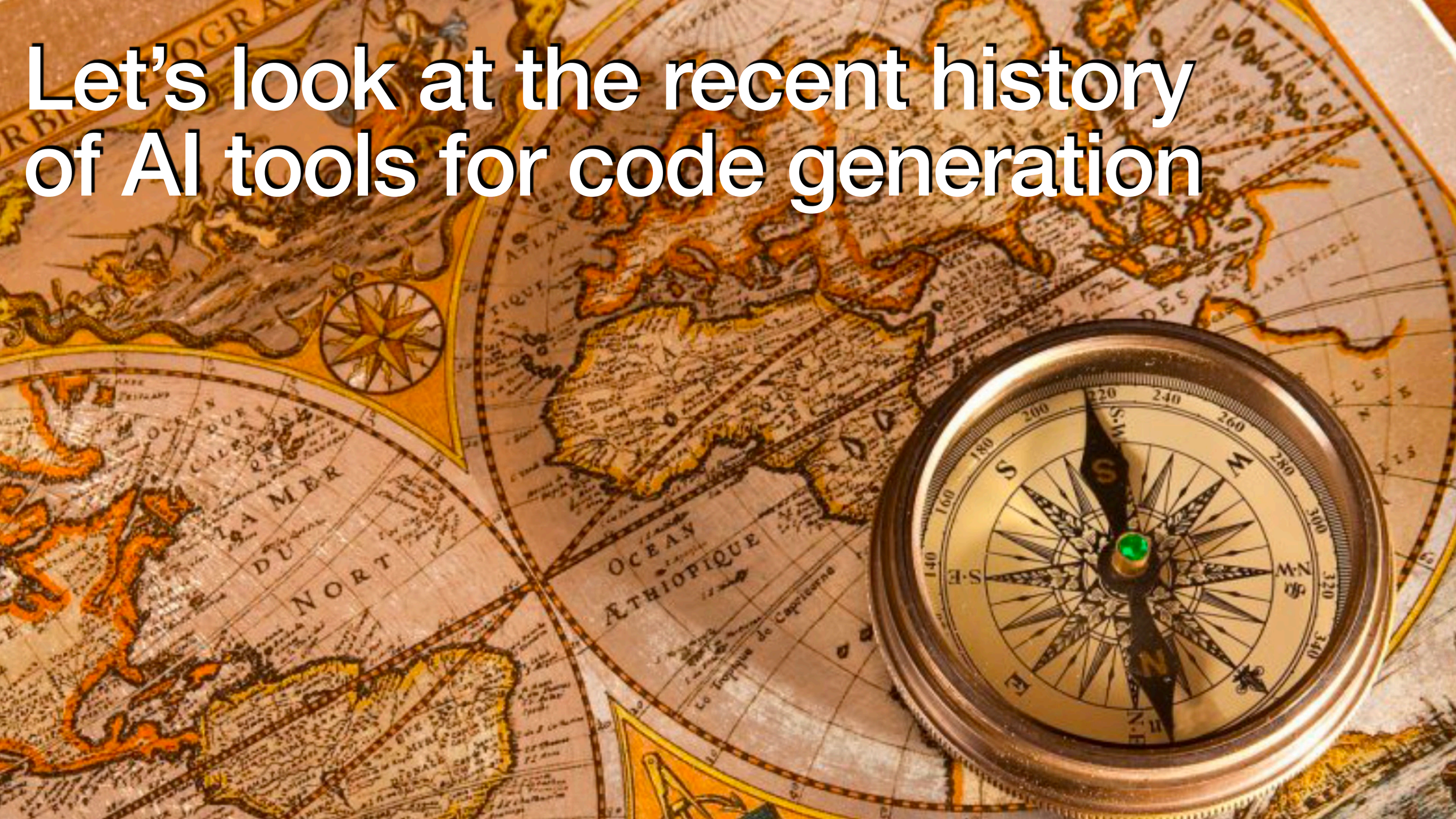
Let’s Take A Quick Survey!

Who is afraid they'll lose their job to AI?

Who has
already lost a job to AI
or knows someone who has?


Let me reassure you!

Let’s look at the recent history of AI tools for code generation



GPT - Generative Pre-training Transformer

- An LLM (Large Language Model)
  - Modern LLMs became popular in 2017
  - GPT was created by OpenAI in 2018
  - Parameters are variables used to infer new content
  - GPT-2 had 2 billion parameters
  - GPT-3 had 175 billion parameters
  - GPT-4 has 1 trillion parameters

GitHub Copilot

- Released in 2021
- Collaboration between OpenAI and Microsoft/GitHub
- Uses Codex, a specialized version of GPT-3 for creating code
- Trained on billions of lines of code from GitHub
- Can learn from a local code base

Amazon CodeWhisperer

- Introduced in 2022
- Trained on billions of lines of code
- Supports code generation for 15 languages
- Specialized for AWS cloud development
- Looks at code *and comments* and then offers help

How Good Is AI, Really?

A ChatGPT example - Generating a web service

show me how to write a Java 17 Spring Boot REST API for storing and retrieving baseball statistics in a MongoDB database. Include JUnit 5 test cases that check for the 5 most common REST errors.

The result

3. Define the MongoDB Configuration:

Create a `MongoDBConfig` class to configure the connection to your MongoDB database.

java — Copy code

```java
import org.springframework.context.annotation.Configuration;
import org.springframework.data.mongodb.config.AbstractMongoClientConfigurat

@Configuration
public class MongoDBConfig extends AbstractMongoClientConfiguration {

    @Override
    protected String getDatabaseName() {
        return "baseball_stats_db";
    }

    @Override
    public MongoClient mongoClient() {
        return MongoClients.create("mongodb://localhost:27017");
    }
}
```

4. Create a Baseball Stat Entity:

Define the model for baseball statistics, e.g., `BaseballStat.java`.

java — Copy code

```java
import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "baseball_stats")
@Data
public class BaseballStat {
    @Id
    private String id;
    private String playerName;
    private int homeRuns;
    private int runsBattedIn;
    // Add more fields as needed
}
```

5. Create a REST Controller:

Create a REST controller to handle CRUD operations for baseball statistics.

java Copy coc

```java
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/baseball-stats")
public class BaseballStatController {

    // Implement your CRUD operations here
}
```

6. Implement REST Endpoints:

Implement the endpoints for creating, reading, updating, and deleting baseball statistics in the controller. For example:

java Copy code

```java
@PostMapping
public ResponseEntity<BaseballStat> createStat(@RequestBody BaseballStat sta
    // Implement logic to create a new stat
}

@GetMapping("/{id}")
public ResponseEntity<BaseballStat> getStat(@PathVariable String id) {
    // Implement logic to retrieve a stat by ID
}

@PutMapping("/{id}")
public ResponseEntity<BaseballStat> updateStat(@PathVariable String id, @Req
    // Implement logic to update a stat
}

@DeleteMapping("/{id}")
public ResponseEntity<Void> deleteStat(@PathVariable String id) {
    // Implement logic to delete a stat
}
```

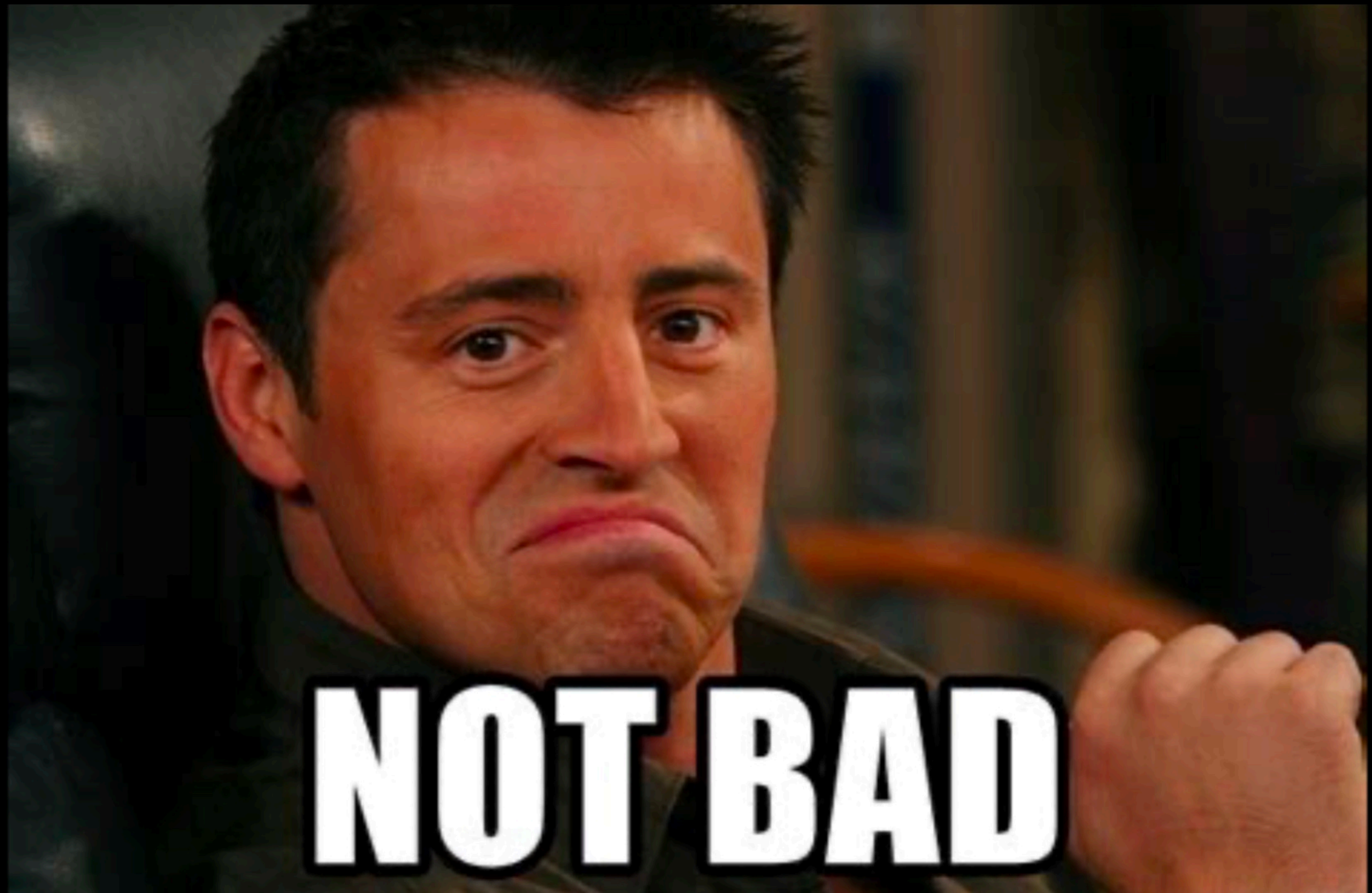
Regene

java

Copy code

```java
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.when;

@SpringBootTest
@AutoConfigureMockMvc
class BaseballStatControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private BaseballStatService baseballStatService;

    @Test
    void createStat_ValidStat_ReturnsCreatedStatus() throws Exception {
        BaseballStat stat = new BaseballStat("John Doe", 20, 50);

        when(baseballStatService.createStat(stat)).thenReturn(stat);

        mockMvc.perform(post("/api/baseball-stats")
                .contentType(MediaType.APPLICATION_JSON)
                .content(asJsonString(stat)))
                .andExpect(status().isCreated());
    }
```

Regene


NOT BAD

Now let’s try something more complicated…

show me how to write an entire enterprise software system for selling over ten thousand small craft supplies, with OpenID security, unit and integration tests, a deployment pipeline, management APIs, backup and rollover processes, an SLA of less than 100ms while supporting 100K concurrent users, providing localization for North American, South American, and European markets, handling PII (personally identifiable information), conform to Sarbanes-Oxley standards, and provide documentation?

What did ChatGPT do…?

It punted.

Developing an entire enterprise software system with such extensive requirements is a complex and time-consuming process that typically involves a team of experienced software engineers, architects, and other professionals. I can provide you with a high-level overview of the steps and components involved in creating such a system. Keep in mind that this is a broad outline, and the actual implementation details may vary based on specific technology choices and business requirements. This overview assumes you have experience with software development and relevant technologies.

1. Define Requirements and Plan:

- Gather detailed requirements from stakeholders.
- Plan the project, including resource allocation and timeline.

2. Choose Technology Stack:

- Select appropriate technologies for backend, frontend, and database.
- Ensure they can meet the performance, security, and compliance requirements.

3. Design the System:

- Create a detailed architectural design for your enterprise system.
- Define the database schema, API structure, and security model.

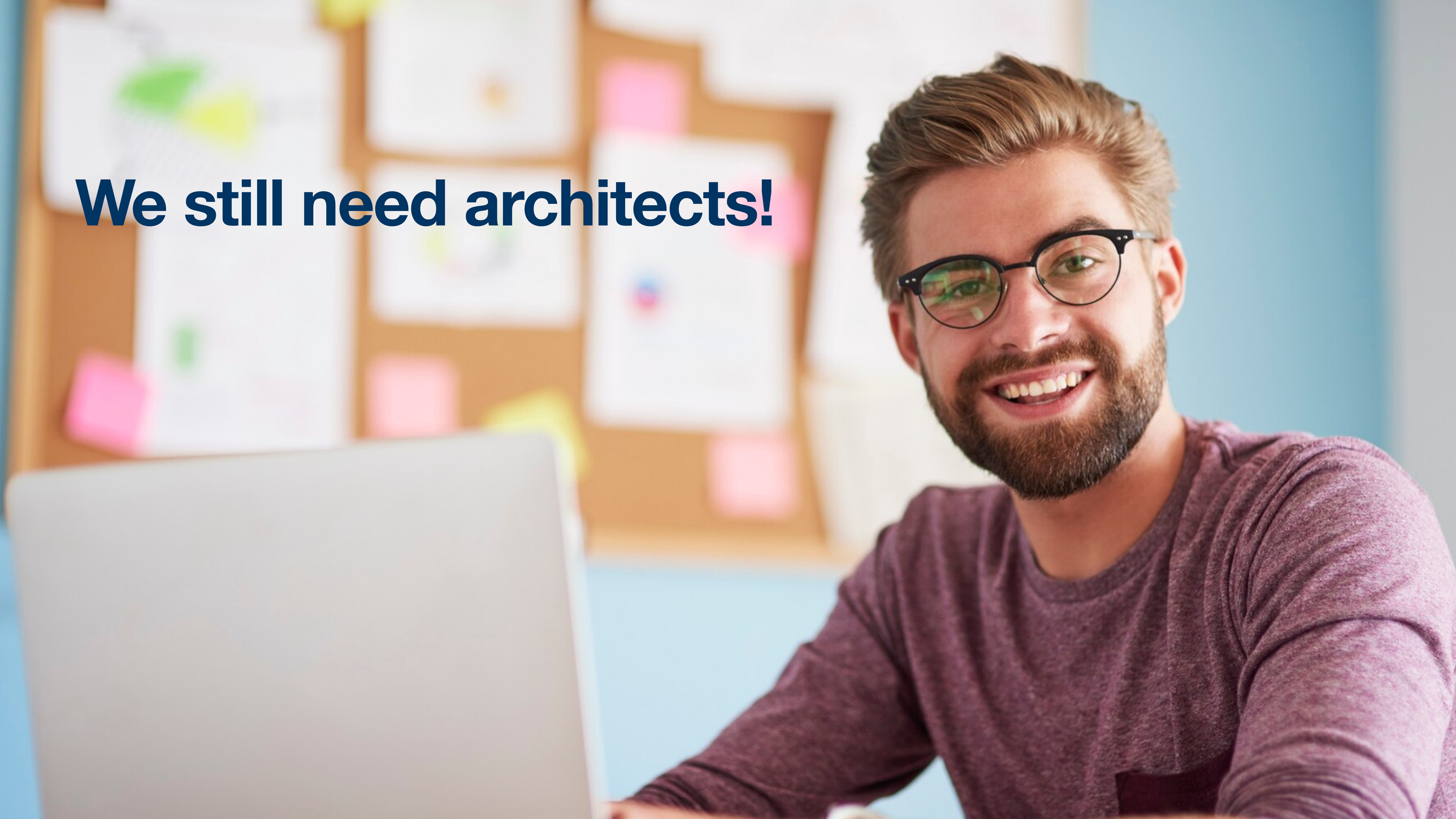
4. Implement User Authentication and Security:

- Implement OpenID for user authentication.
- Implement encryption, access control, and auditing mechanisms to ensure data security

Reger

Send a message

We still need architects!



AI can do the “ground level” work.

But we still need humans for large orchestration.

I.E., software architects.

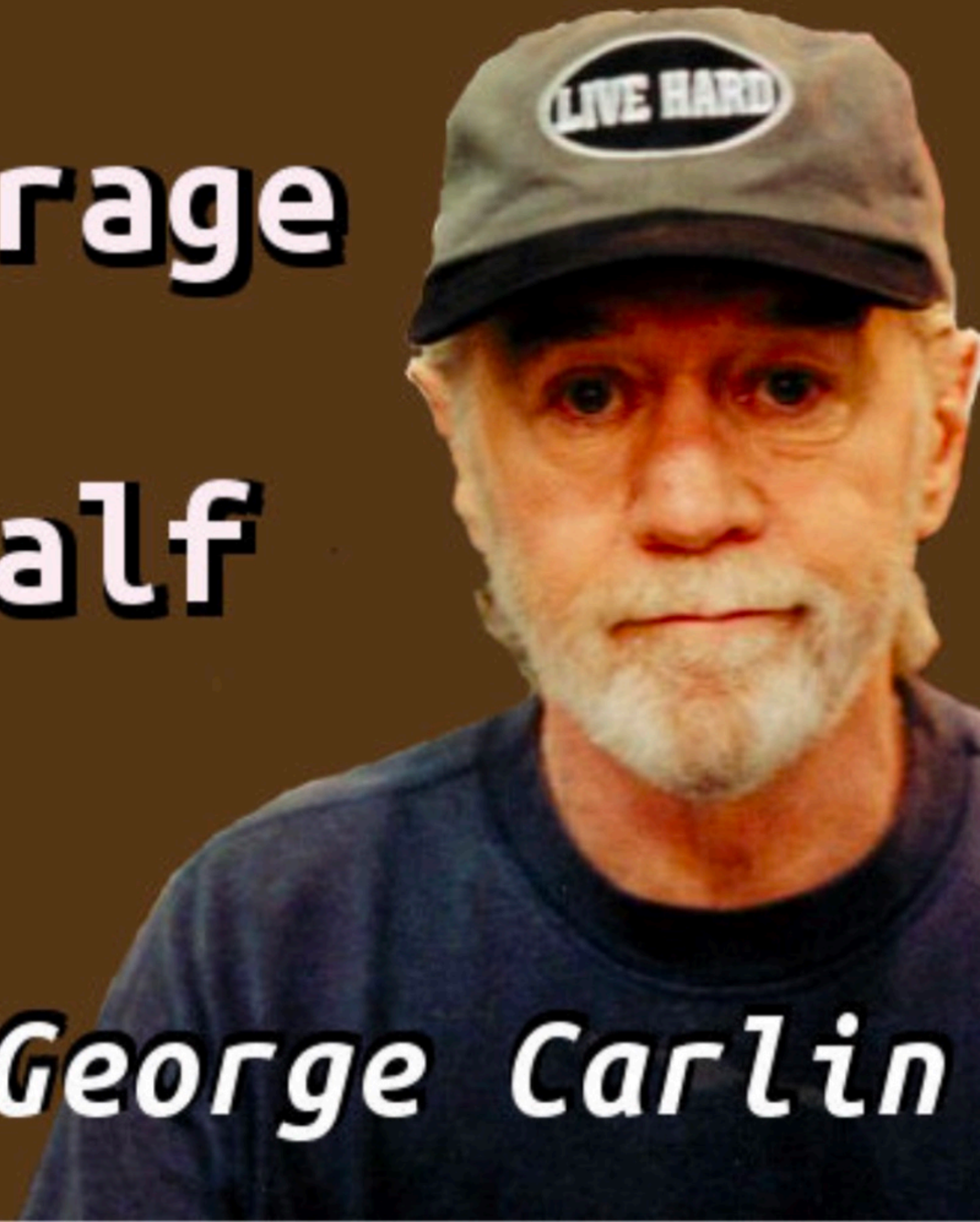
There are other issues with AI generated code.

Let’s look at some of them.

GIGO - Garbage In, Garbage Out

- GPT-3 and Copilot were trained on public code in GitHub.
- Not all of it is good!
- Not all of it is correct!
- Some of it is obsolete!
- Newly generated code could spread poor quality or incorrect code.

“Think of how stupid the average ~~person~~ programmer is, and then realize half of them are stupider than that.”



George Carlin

Hallucinations

- GPT doesn’t really “understand” what it’s asked to do.
- It’s a mime, a mimic, a parrot - it tries to put together something that looks like what the answer should be.
- This works fairly well if it’s been trained on similar items.
- But if it can’t find an answer, it will put together something that looks like what you want, *even if it has to make something up.*
- We call these “hallucinations”.

Example hallucination

4. Create the new pod using the `kubectl create` command and specify a unique name directly:

```lua
kubectl create -f pod_copy.yaml --name=xxx-copy-20230613
```

Replace `xxx-copy-20230613` with the desired unique name for the new pod.

This approach explicitly sets the name during the `kubectl create` command, bypassing any potential conflicts with existing resources in the cluster.

Please note that if the pod copy still fails to be created, there might be other factors or issues at play. It's recommended to check the Kubernetes cluster logs or consult with your cluster administrator for further assistance in troubleshooting the problem.

kubectl does not recognize the --name option

I apologize for the incorrect information provided. You are correct that `kubectl` does not have a `--name` option for specifying the name during the `create` command. Please disregard that suggestion.

“Monkey See, Monkey Do”

- GPT relies on being trained on similar data
- What of brand new languages and technologies?
- At first, extremely limited training data is available… right when it’s needed the most!
  - E.g., Rust was created in 2006. How many example programs were available by 2007 or 2008?
  - Mojo was created in May 2023. How many example Mojo programs do you think GPT has trained on by now?
- With little training data, GPT is more prone to “hallucinations”.

Security Concerns

- Every time ChatGPT interacts with you… it learns from you.
- If private information gets into an LLM, it is almost impossible to remove.
  - Passwords.
  - Intellectual property.
  - Trade secrets.
- Some corporations forbid the use of these models.
  - Or require anonymized, “air gapped” use.

Debugging

- Junior programmers can easily generate code with AI tools.
- But is it correct?
- If an error occurs in AI-generated code, can a human who has never looked at the code understand it well enough to debug it?
- GPT and Copilot can sometimes debug code… but pasting it in a prompt could leak confidential data or custom IP.

Upgrades, Modernization, And Legacy Code

- Some common tasks are easier for humans than AI.
  - Changing frameworks - e.g., from Lombok to slf4j, Angular to Angular 2, Angular to React, Java Dates to the JDK 8 Date/Time API.
  - Some still needs human eyes: working with CSS, for example.
  - Major architectural changes, such as converting to a DDD design.
  - Understanding when requirements change and the impact on the code.

Industry Reluctance

- We’re already seeing pushback against AI in other fields.
  - Law
    - AI “lawyers” in the courtroom? “Made up” legal precedents?
  - Hollywood (see the current strike)
    - Copying old plots, scripts, characters, character arcs?
- Could software development be next?
- Can AI-generated software be copyrighted?
  - In March 2023, the U.S. Copyright Office said that works produced with the help of AI “may” be copyrightable.
  - It’s a fuzzy area for now.

Critical Software

- Some software is too important to trust to AI tools that “hallucinate”.
  - Avionics software.
  - Full self-driving car software.
  - Embedded software for medical devices like pacemakers.
  - Embedded software for spacecraft and satellites.

Lack Of Contextual Awareness

- AI tools do not know what other software exists at a company.
- They do not know company standards or practices.
- They do not know whether existing software could be tweaked or reused.
- They do not know why certain decisions were made (e.g., GCP instead of AWS).
- They do not understand security concerns.

Lack Of Creativity

- Could an LLM have created:
  - DVR time-shifting software?
  - The MPEG-4 encoding algorithm?
  - LL and LR grammars and general compiler theory?
  - The Internet protocols?
  - Figuring out how to self-drive a car in different environments with different laws?
- There will always be room for pure original thinking.

Okay, But What Can It Do Well?

- Low-level code generation: REST APIs, configuration, database access, etc.
- Code optimization.
- “Greenfield” development.
- Documentation generation.
- Test case generation.
- Basically, the kind of tasks you hand off to a junior developer.

What’s A Software Developer To Do?

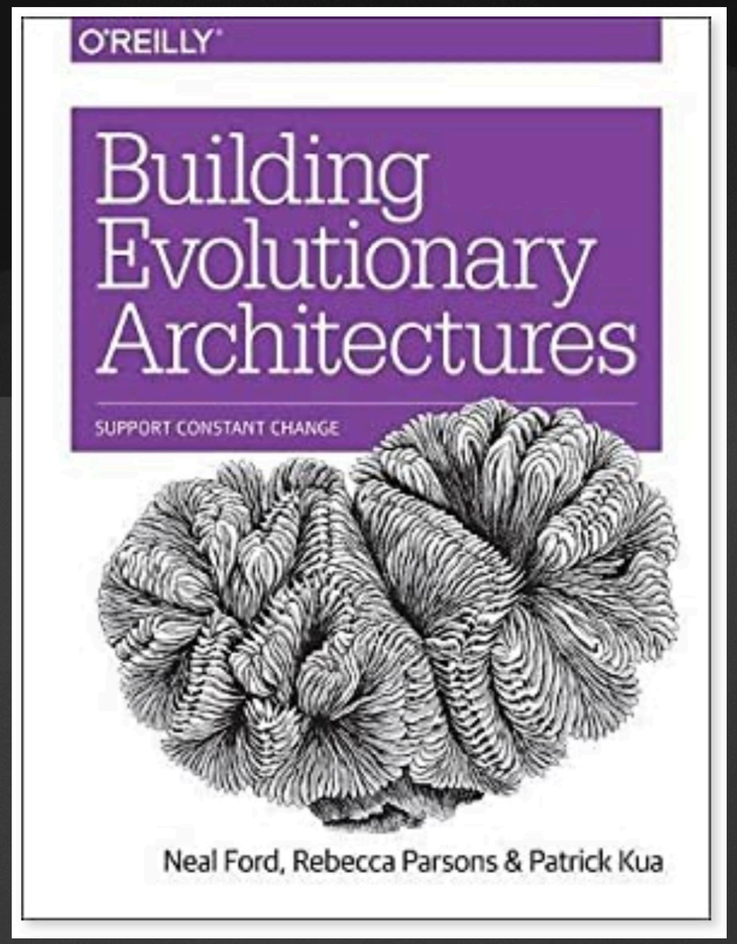
Focus on architecture.
Not coding.

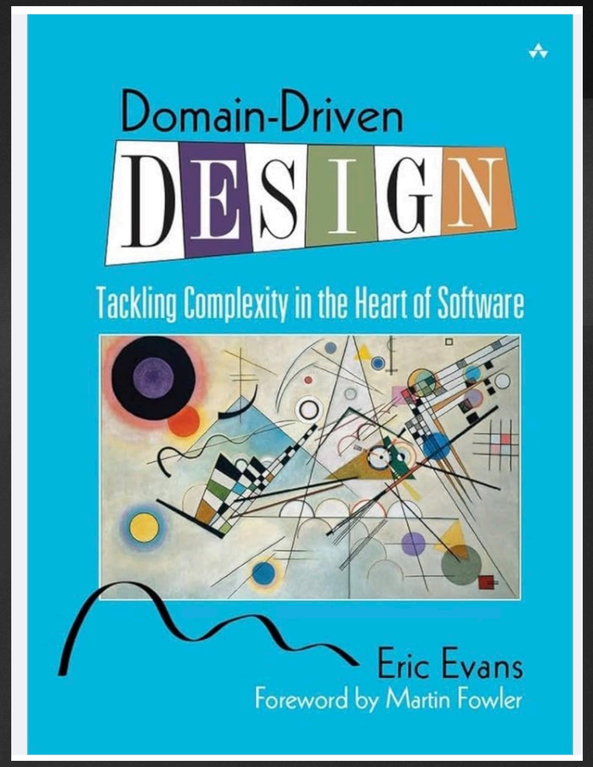
Junior Developers

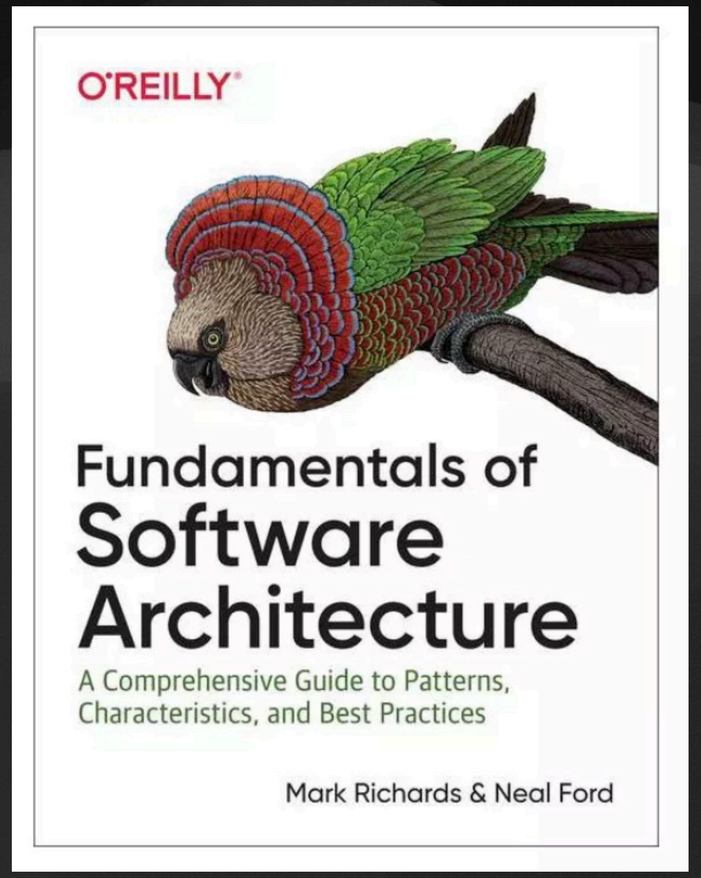
- Don’t just learn a language or framework.
  - Learn which languages are best in which situations.
  - Learn the common idioms.
- Look beyond: pricing, availability of libraries, availability of programmers.
- Learn the architectures that are best implemented in various languages.
- Learn how to create great prompts for code generation.
- Learn how to understand, follow, test, and debug AI-generated code.

Senior Developers / Architects

First, study architecture!







Learn architectures and how to implement them

- Layered architecture
- Event driven
- Microkernel
- Microservices
- Space-based architecture (tuple space)
- Client / server
- Broker, peer to peer, etc.

Learn the advantages and disadvantages of each.

Learn when to apply them.

Divining Requirements

- AI-based tools cannot create requirements
- Most domain experts do not know enough about software to specify requirements
- An architect can be the bridge between the domain experts and the AI tools
  - Learn how to guide a domain expert and understand the problem
  - Learn how to help formulate a suitable architecture to solve the problem

Mentoring Junior Developers

- Work with them to teach them how to craft high quality prompts
- Remember to ask for: security, test cases, documentation, design patterns, OWASP checks, etc.
- Show them how to spot hallucinations, and what to do if one occurs.
- Work with them to understand how to understand and debug AI-written code.
- Help them learn architecture.
  - Don’t just say “implement X using framework Y.”
  - Instead, “we’re implementing X using Y instead of W or Z because…”

Ensure Design Specifications Are Met

- Ensure code reviews are held to make sure that AI-generated code is solid
  - Perhaps add pre-commit git hooks to test code
- Lean on AI tools to generate unit tests
- If using Java or C#, look into ArchUnit (archunit.org)
  - It’s similar to JUnit, but for testing architectures
  - Fluent interface - easy to write in an IDE
  - You can write your own rules!

ArchUnit Example

```java
@ArchTest
private final ArchRule no_generic_exceptions = NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS;

@ArchTest
private final ArchRule no_java_util_logging = NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING;

@ArchTest
private final ArchRule loggers_should_be_private_static_final =
        fields().that().haveRawType(Logger.class)
                .should().bePrivate()
                .andShould().beStatic()
                .andShould().beFinal()
                .because("we agreed on this convention");

@ArchTest
private final ArchRule no_jodatime = NO_CLASSES_SHOULD_USE_JODATIME;

@ArchTest
private final ArchRule no_field_injection = NO_CLASSES_SHOULD_USE_FIELD_INJECTION;
```

Another ArchUnit Example

```
import static com.tngtech.archunit.library.Architectures.layeredArchitecture;

@AnalyzeClasses(packages = "com.tngtech.archunit.example.layers")
public class LayeredArchitectureTest {
    @ArchTest
    static final ArchRule layer_dependencies_are_respected = layeredArchitecture().consideringAllDependencies()

            .layer("Controllers").definedBy("com.tngtech.archunit.example.layers.controller..")
            .layer("Services").definedBy("com.tngtech.archunit.example.layers.service..")
            .layer("Persistence").definedBy("com.tngtech.archunit.example.layers.persistence..")

            .whereLayer("Controllers").mayNotBeAccessedByAnyLayer()
            .whereLayer("Services").mayOnlyBeAccessedByLayers("Controllers")
            .whereLayer("Persistence").mayOnlyBeAccessedByLayers("Services");
```

Conclusion

- AI can perform junior developer tasks with guidance
- It’s still a long way from being able to replace software architects
- Software architects need to learn to incorporate AI tools in their process
- Knowledge of various architectures, their tradeoffs, costs, etc. is vital
- Junior developers need to be mentored to work effectively with AI
- Finally, by performing the menial tasks, it can free developers to focus on higher level design and architecture, producing better quality systems.


I'M KELLY I. MORRISON
ANY QUESTIONS?

Daugherty

BUSINESS SOLUTIONS

Work: kelly.morrison@daugherty.com

Home: kellyivymorrison@gmail.com

Kelly Morrison
Manager / Application Architect at Daugherty Business Solutions

GitHub

LinkedIn