

imposter's
The ~~architect's~~ guide to
generative AI

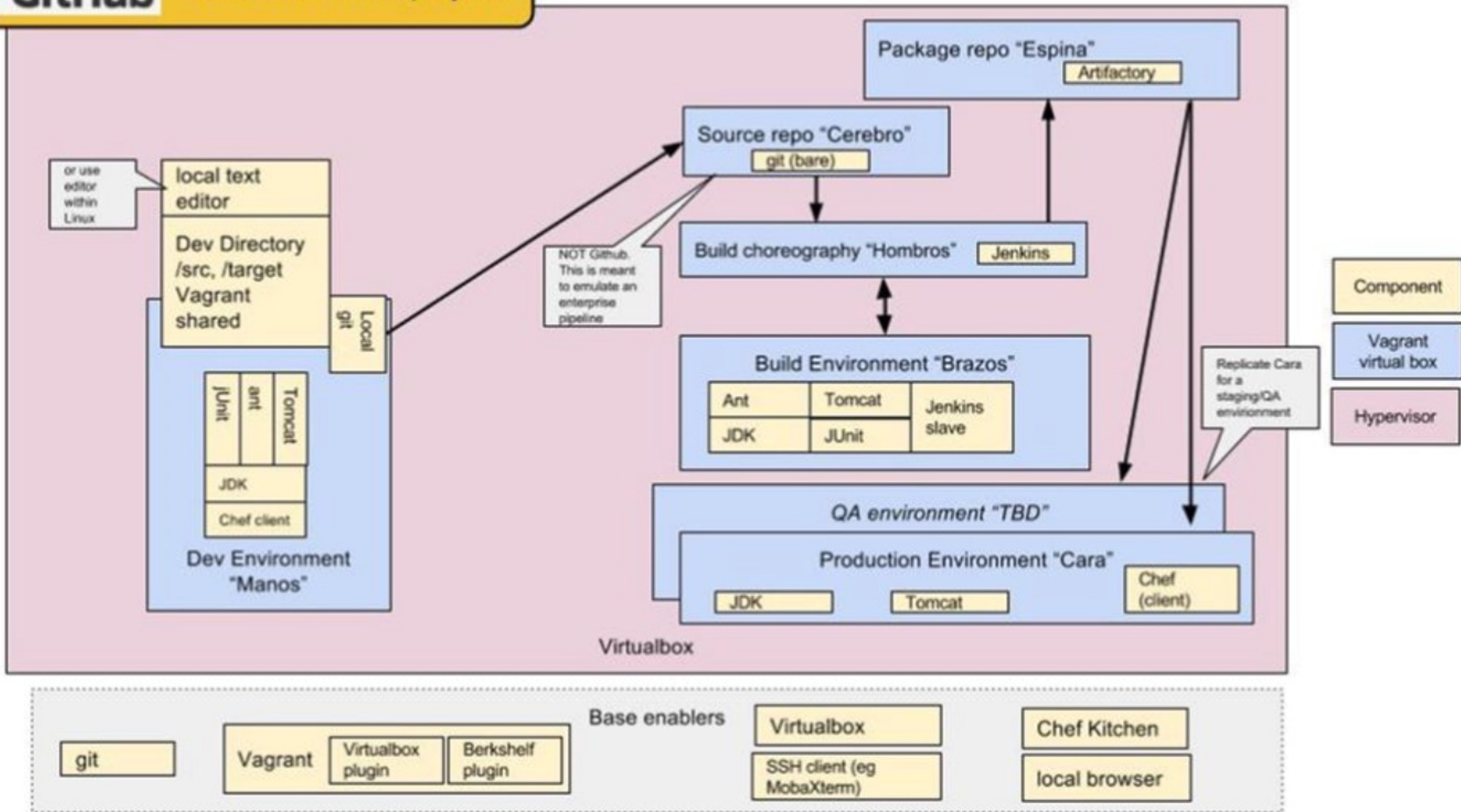
Charles Betz
Vice President
Research Director





sigh








General Value Propositions

- **Identifying similarities in complex data.**
- **Knowledge management and articulation.**
- **Narrative ideation.**
- **Natural language interaction with digital systems.**
- **Image recognition and generation**

As with steam, electricity, and computing itself, domain specific value propositions are too numerous to list.





What as an EA do I need to know about GenAI?

What are GenAI's impacts on EA?





- WSJ Investor of the Year
- Star of *Erin Brockovitch* and *Mystic Pizza*
- Granddaughter of Richard Dreyfuss

LIONEL HUTZ WOULD BE ASHAMED —

Lawyers have real bad day in court after citing fake cases made up by ChatGPT

Lawyers fined \$5K and lose case after using AI chatbot "gibberish" in filings.

“Gibberish,” “borders on nonsensical”

The fake *Varghese v. China Southern Airlines* excerpt submitted by LoDuca and Schwartz, supposedly issued by the US Court of Appeals for the 11th Circuit, "shows stylistic and reasoning flaws that do not generally appear in decisions issued by United States Courts of Appeals," Castel wrote. "Its legal analysis is gibberish... The summary of the case's procedural history is difficult to follow and borders on nonsensical."

The *Varghese* decision also "includes internal citations and quotes from decisions that are themselves non-existent," the judge noted. The other fake precedents submitted to the court had similar flaws.



It's not just hallucination, it's also plagiarism. "The complaint used the example prompt "Write me a song about the death of Buddy Holly," which led the large language model to spit out the lyrics to Don Mclean's "American Pie" word for word." Universal Music Group vs Anthropic.

**It is not a singularity – yet.
It is an architecture.**



Langchain, APIs, RAG, etc

Fine tune

Train a model

Invest here

Be skeptical

Stop the meeting

How not to do it

To win at GenAI, we need to train a Large Language Model on ALL of our corporate data by next month! No excuses!

Sure, we'll have a model that sounds a lot like you, including inaccurate and unsupported mansplaining.

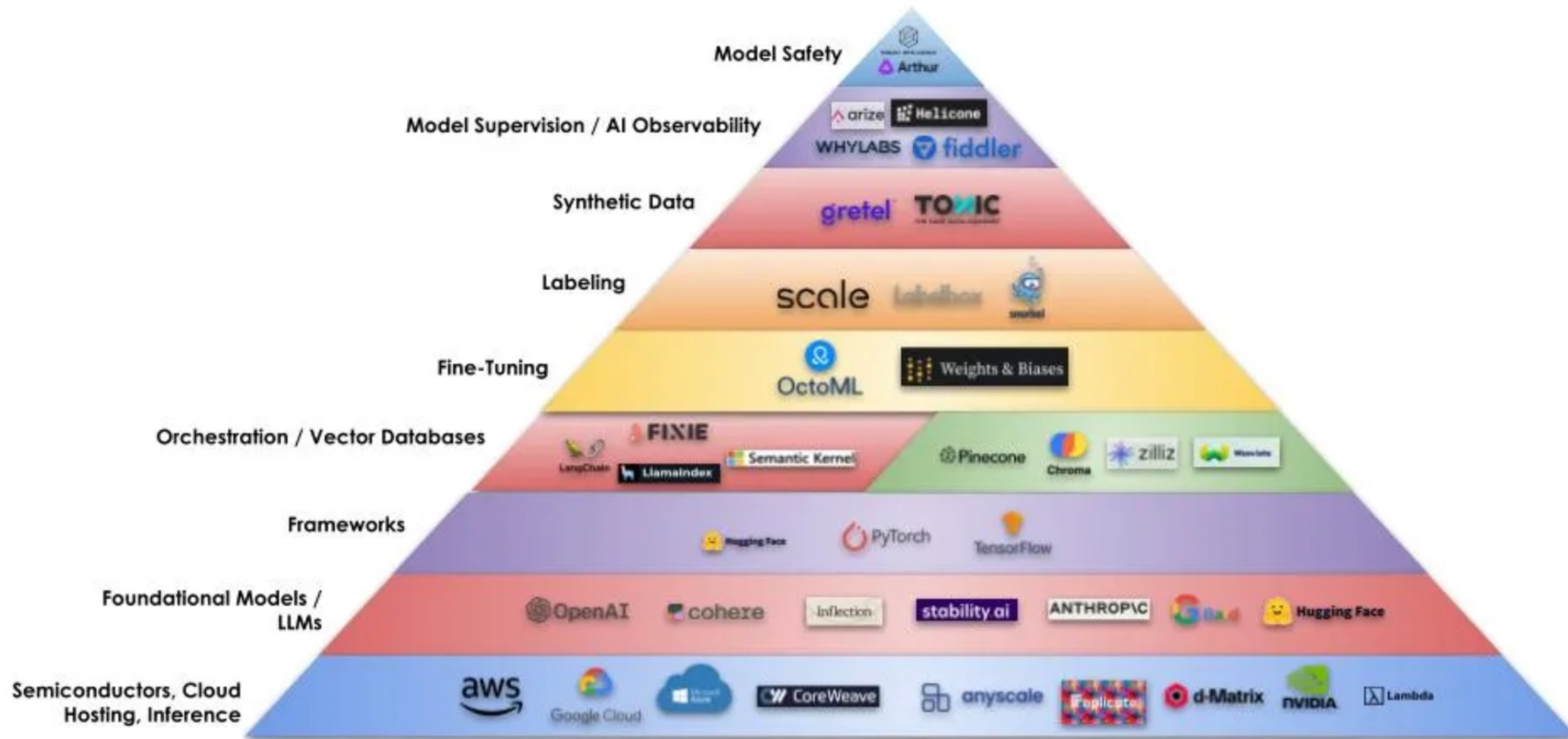




[A foundational LLM]... has no connection to authoritative sources, so it tends to hallucinate or speculate incorrectly.

(David Shapiro)

The Building Blocks of Generative AI



Attention Is All You Need

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

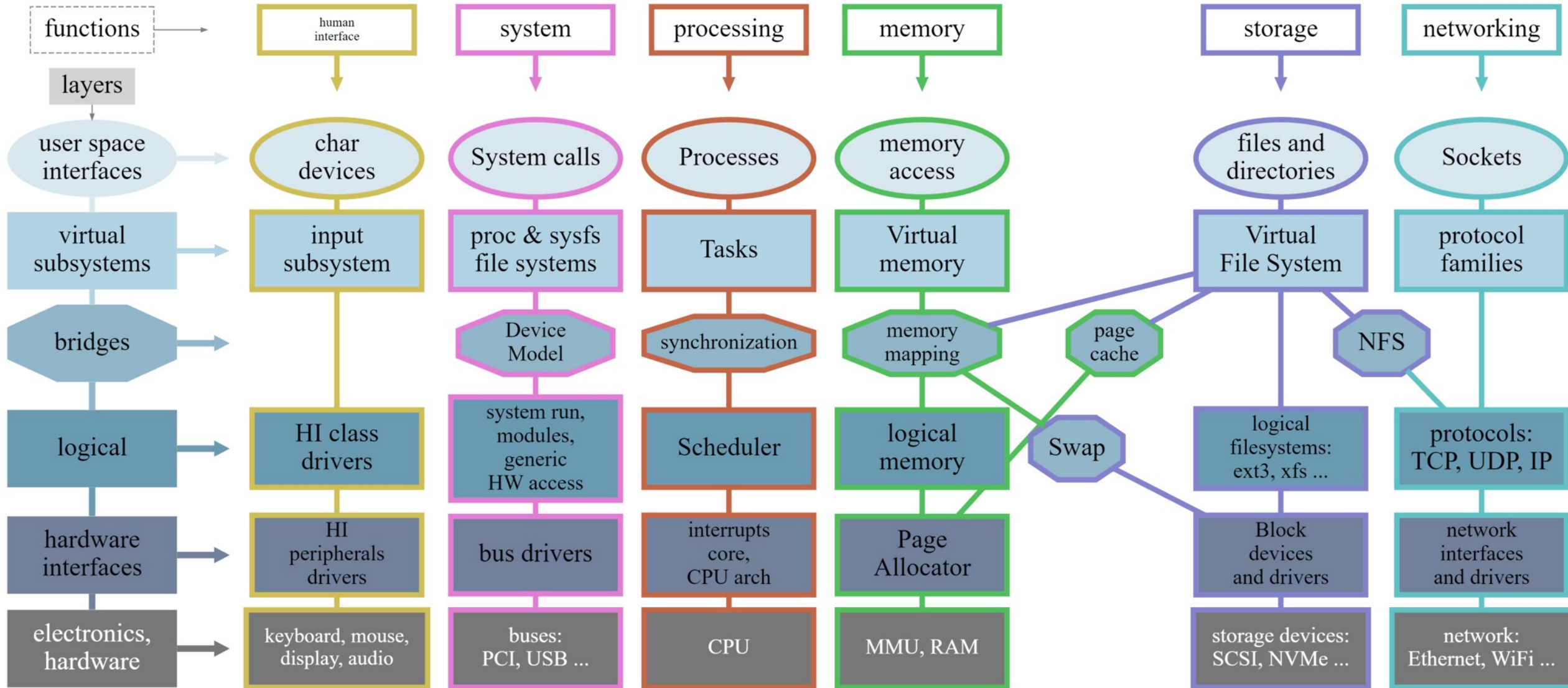
Efficient Estimation of Word Representations in Vector Space

The training complexity of this architecture is proportional to

$$Q = C \times (D + D \times \log_2(V)), \quad (5)$$

where C is the maximum distance of the words. Thus, if we choose $C = 5$, for each training word we will select randomly a number R in range $\langle 1; C \rangle$, and then use R words from history and

Linux kernel diagram

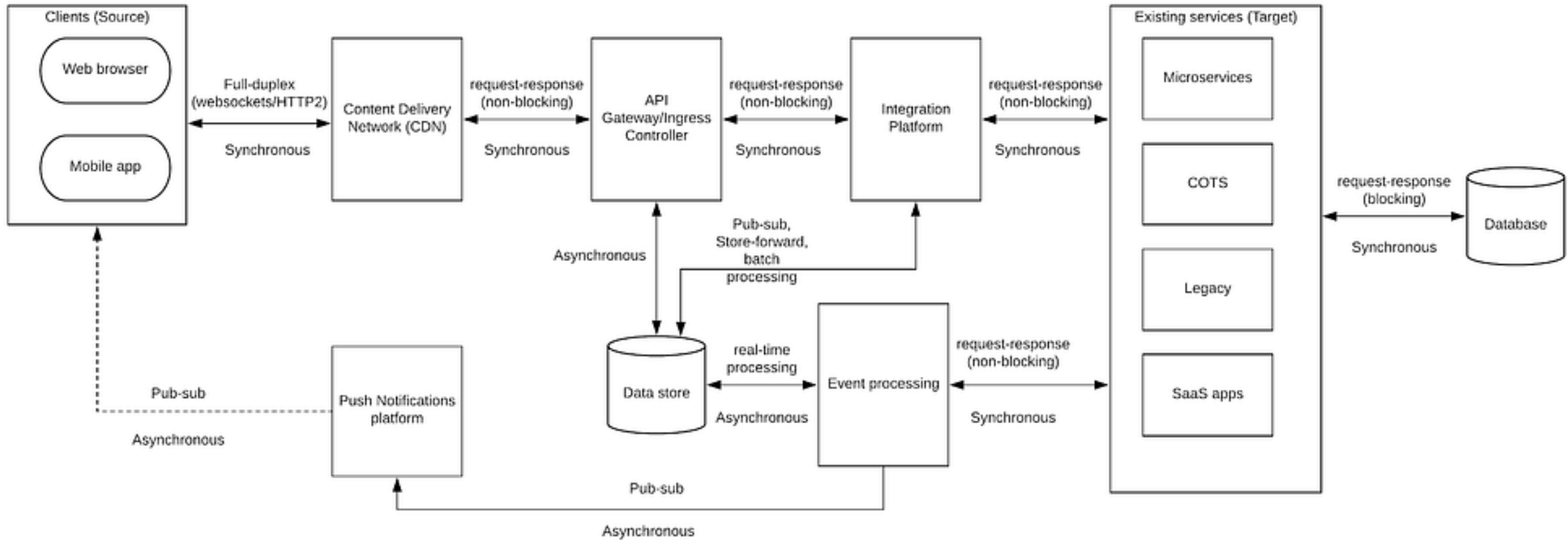




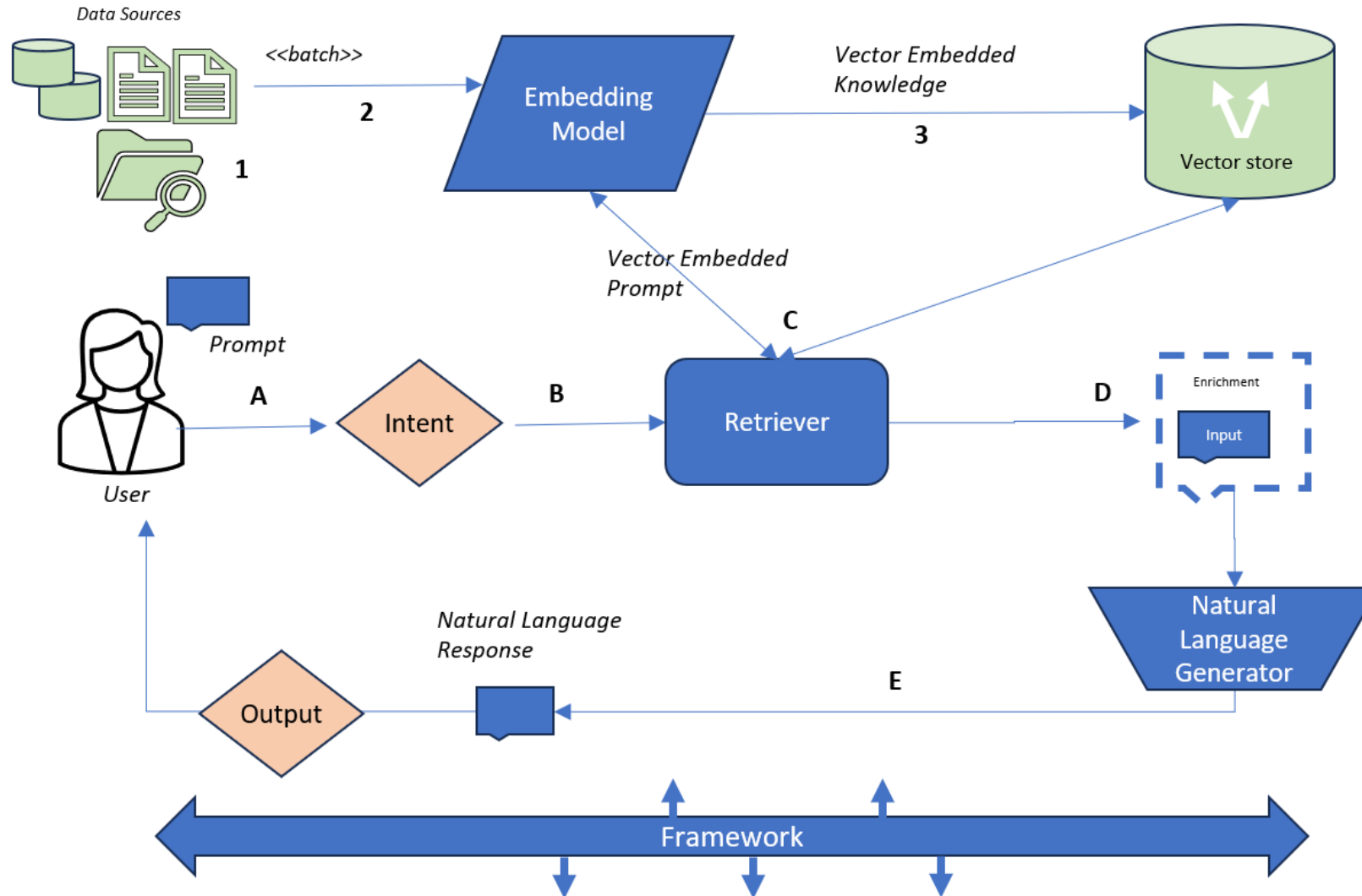
DEWALT

DEWALT
18V XR
LITHIUM
3.0Ah

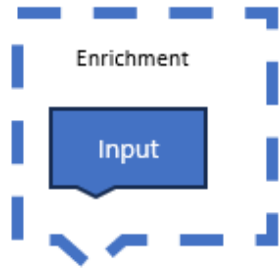
18V



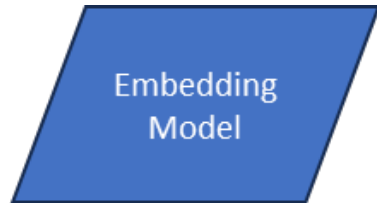
Prompt enrichment (RAG)



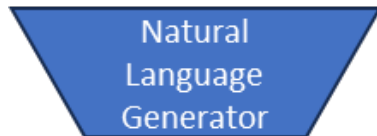
Primitives I



The prompt is the query, what the GenAI architecture responds to. In the simplest model, the end user generates the prompt. Prompts can be simple questions or complex instructions, guiding the model's output. Prompts originated with the user and then may be enriched in various ways to get the best results.

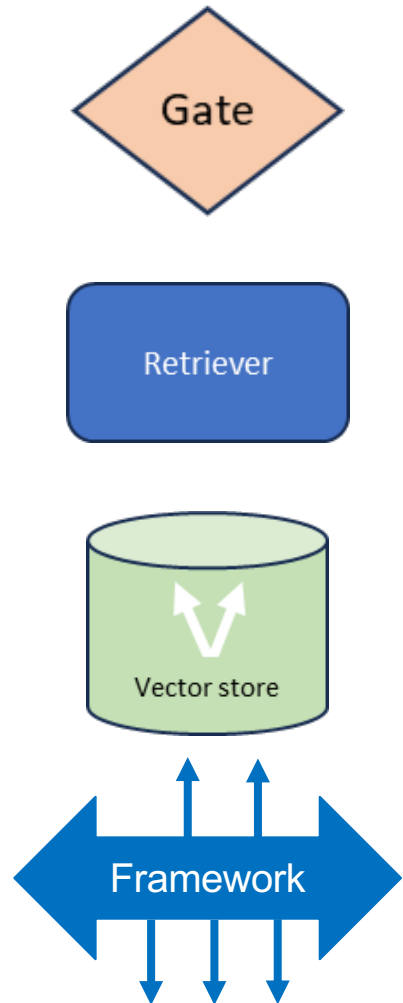


Embeddings compress complex data into simple yet powerful multi-dimensional representations ("vectors"). The embedding model (such as BERT) encodes words, sentences, and paragraphs into embeddings which are then stored in the vector database.



The NLG is the best known component of this architecture, due to ChatGPT. They are based on neural networks trained with massive amounts of data. NLGs are able to answer general knowledge questions through essentially calculating the probability of one word following another. They typically do not remember their sources of knowledge (they can't cite their answers, -- they are not "grounded") and are prone to "coherent nonsense" (aka hallucinations).

Primitives II



Gates can classify the type of prompt (summarize vs describe) for more efficient processing, and ensure that the AI architecture produces safe, secure, and unbiased outputs. These can be implemented via model training, or pre-or post-processing filters (e.g. the AI refuses to answer questions about how to build weapons, and evaluates its output for racist implications).

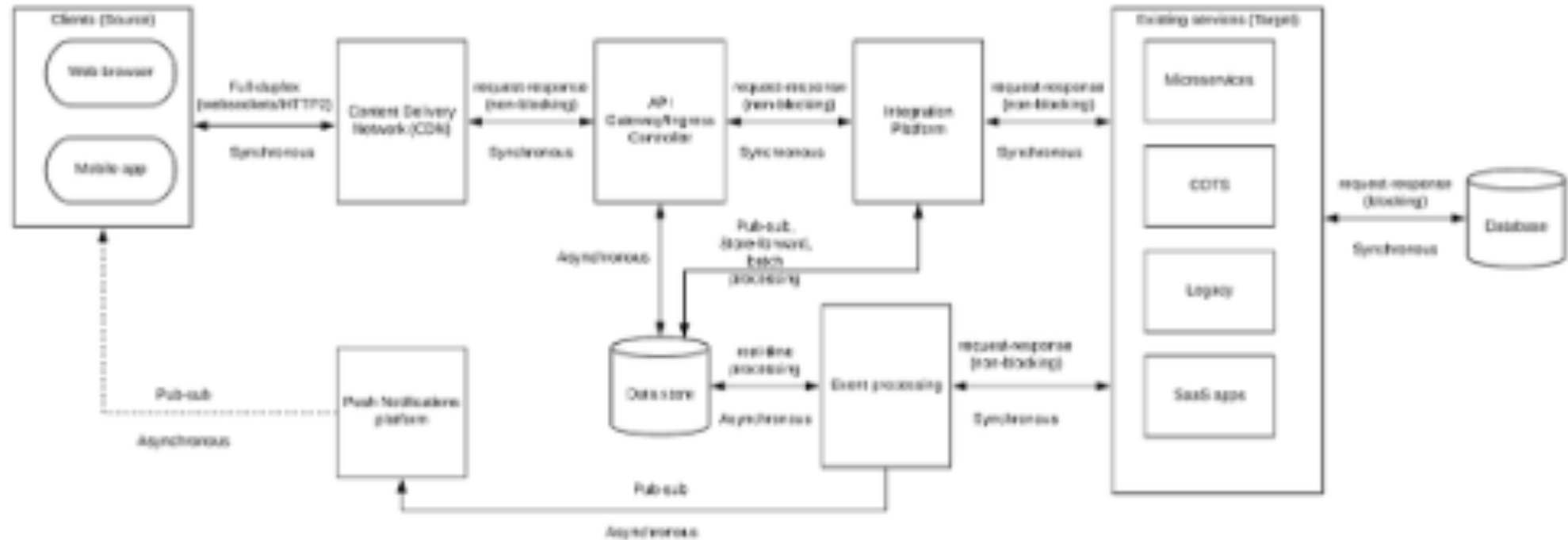
Code that processes an input and retrieves relevant context from one or more Knowledge Stores. Retrievers may call embedding models to encode input prompts and intent metadata to be compared to Vector Stores. Types of Retrievers include keyword, sparse passage, dense passage and ensemble. Dense passage and ensemble are the current state of the art in GenAI applications.

Knowledge stores are a class of building blocks that form an intelligence layer supporting GenAI powered applications. Since generative AI model training and fine tuning is expensive and difficult, the model's knowledge is often fixed in time and again is not grounded. Current knowledge is important context to ensure outputs are helpful. Specific knowledge stores include Vector Stores, Search Indices, Knowledge Graphs, etc.

Frameworks provide an overall execution architecture for the pipeline interactions, abstracting direct interaction with LLMs and other building blocks into one common, easy to program environments.



AI as architecture assistant



can you explain this diagram?

1. Clients (Source)

- **Web browser & Mobile app:** These are the client-side applications or interfaces through which end-users interact with the system.
- The diagram shows two methods for the clients to communicate with the backend:
 - **Synchronous:** The client sends a request and waits for a response. This is typical of many web and mobile interactions.
 - **Full-duplex (websockets/HTTP2):** This is a bi-directional communication channel. Unlike traditional HTTP where the client sends a request and waits for a response, in a full-duplex mode (like websockets), both client and server can send messages at any time.

2. Content Delivery Network (CDN)

- CDNs are used to deliver content to clients efficiently. Clients connect to the nearest edge location of a CDN, reducing latency. The CDN communicates synchronously using a non-blocking request-response pattern.

3. API Gateway/Ingress Controller

- This acts as the entry point to the system's backend. It routes incoming requests to the appropriate service or component.

4. Integration Platform

- Facilitates communication and data flow between various services or components of the system.

5. Push Notifications platform

- Allows the system to send real-time notifications to clients. The notifications can be triggered by various events in the system and are delivered to the clients asynchronously using a publish-subscribe (pub-sub) model.

6. Data store

- This is where the system's data is stored. It supports real-time processing of data, and can communicate with other components both synchronously and asynchronously.

7. Event processing

- This component processes events in real-time, possibly responding to changes or updates in the data.

8. Existing services (Target)

- **Microservices:** Small, independent services that run specific business functions.
- **COTS (Commercial Off-The-Shelf) software:** Pre-built software products used within the system.
- **Legacy:** Older systems or components that are still in use.
- **SaaS apps:** Software-as-a-Service applications that are hosted externally but integrated with the system.
- **Database:** A storage system that supports synchronous, blocking request-response communication.

Overall, the diagram depicts how different components of the system interact with each other, whether it's synchronously or asynchronously, and via which communication pattern (e.g., request-response, pub-sub). The architecture supports a mix of real-time processing, data storage, and integration with both modern and legacy components.



Please draw an entity relationship diagram of a customer order management system in Mermaid UML syntax

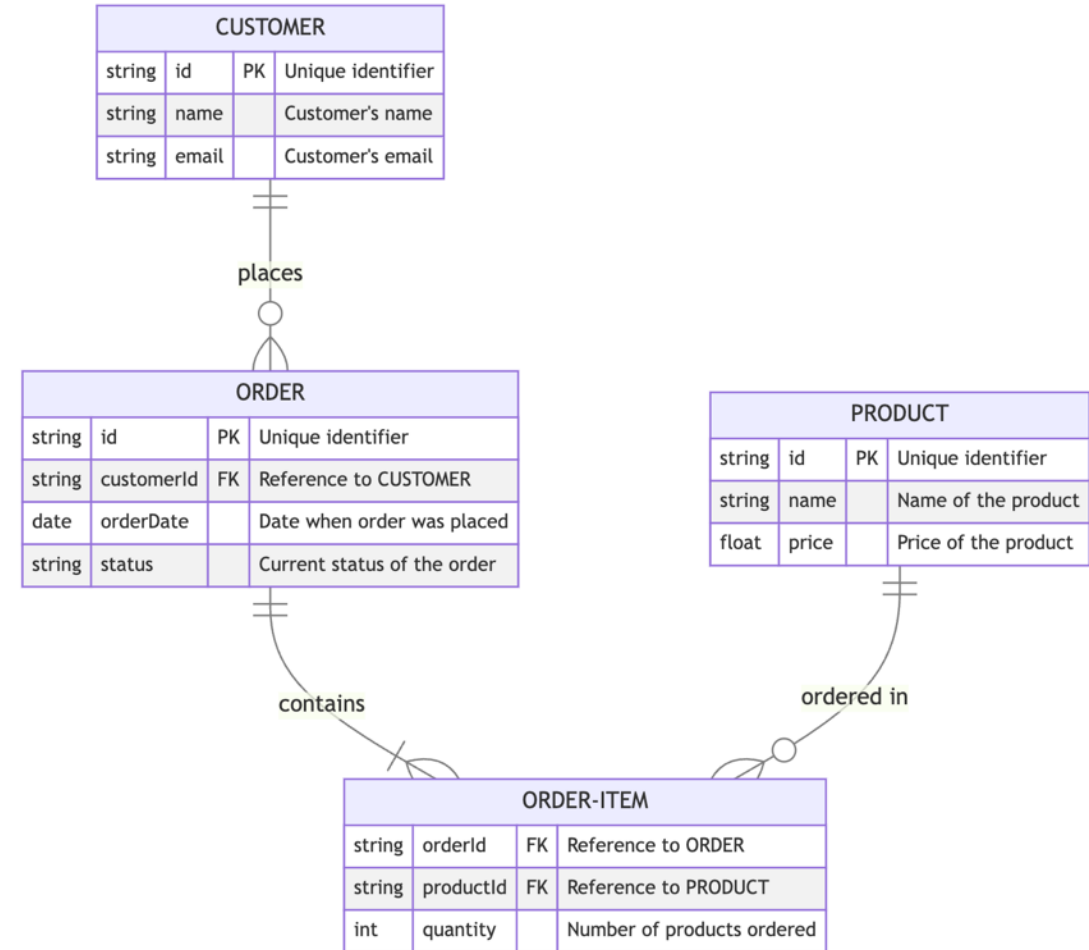


Certainly! Below is an example of how you might represent a simple customer order management system using Mermaid's UML syntax:

```
mermaid
erDiagram
    CUSTOMER ||--o{ ORDER : places
    ORDER ||--}|{ ORDER-ITEM : contains
    PRODUCT ||--}|{ ORDER-ITEM : "ordered in"
```

In this diagram:

- **CUSTOMER** is connected to **ORDER** with a one-to-many relationship, as a customer can place multiple orders.
- **ORDER** is connected to **ORDER-ITEM** with a one-to-many relationship, since an order can contain multiple items.
- **PRODUCT** is connected to **ORDER-ITEM** with a one-to-many relationship, as a product can be in multiple order items (across different orders).
- Each entity has a primary key (PK) denoted as `id`.
- Foreign keys (FK) denote references to other entities.





please draw me an interaction diagram of a retrieval augmented generation system in Mermaid UML syntax



A Retrieval-Augmented Generation (RAG) system typically involves components such as a retriever (to fetch relevant documents), a generator (to produce text), and a database (to store the documents to be retrieved). Here's a sequence diagram in Mermaid UML syntax that shows how these components might interact during the generation process:

mermaid

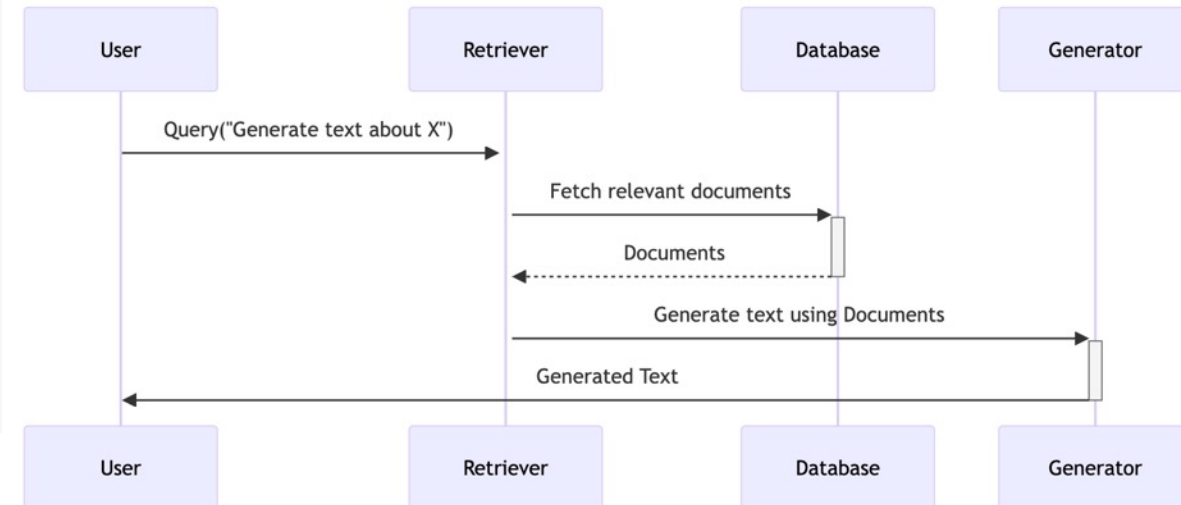
Copy code

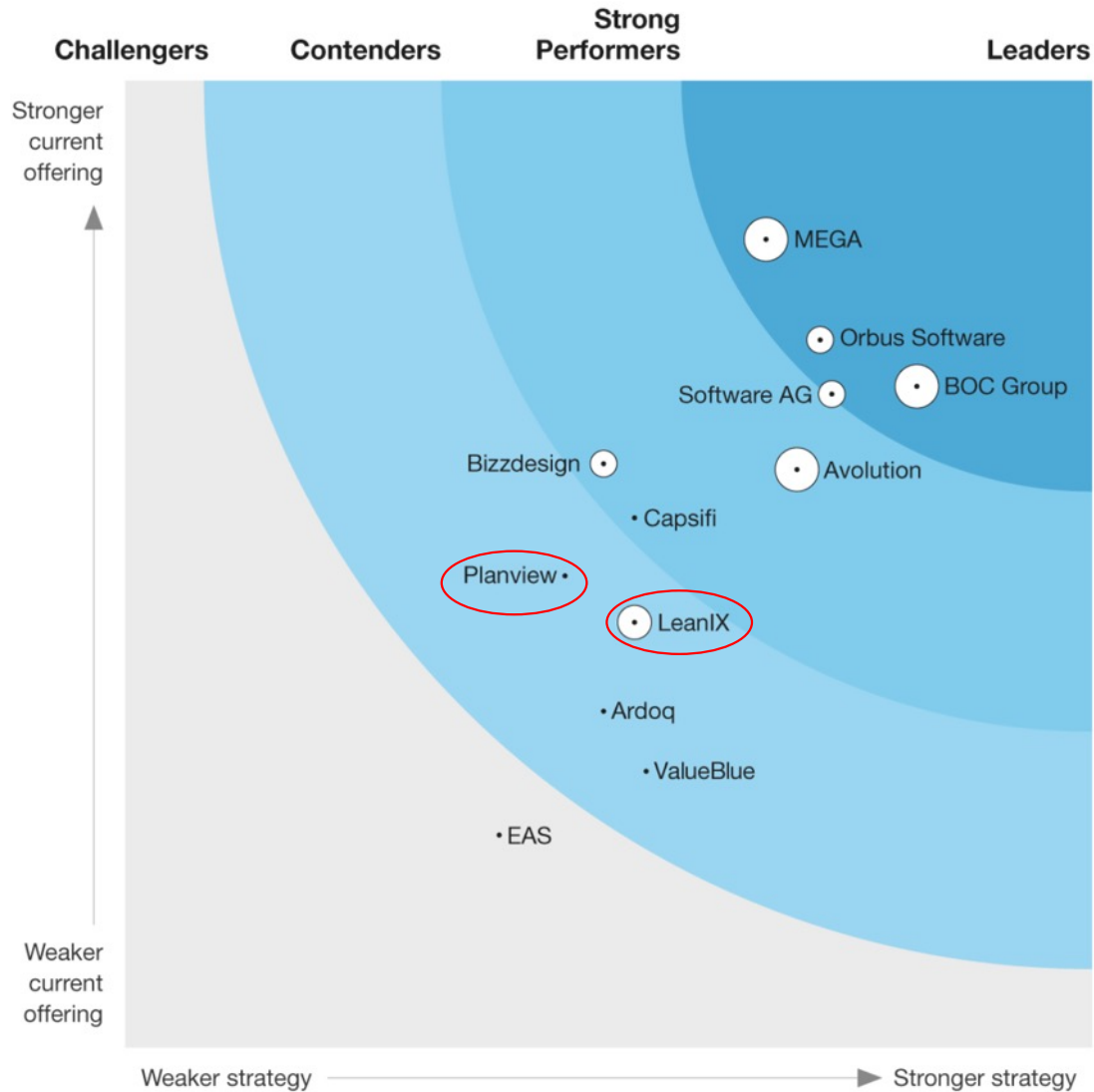
```
sequenceDiagram
    participant User
    participant Retriever
    participant Database
    participant Generator

    User->>Retriever: Query("Generate text about X")
    Retriever->>Database: Fetch relevant documents
    Database-->>Retriever: Documents
    Retriever->>Generator: Generate text using Documents
    Generator-->>User: Generated Text
```

In this interaction:

- The **User** sends a query to the **Retriever** to generate text about a certain topic (X).
- The **Retriever** requests relevant documents from the **Database**.
- The **Database** returns the documents back to the **Retriever**.
- The **Retriever** then sends these documents to the **Generator**.
- The **Generator** uses the documents to generate the text and sends the generated text back to the **User**.





Market presence



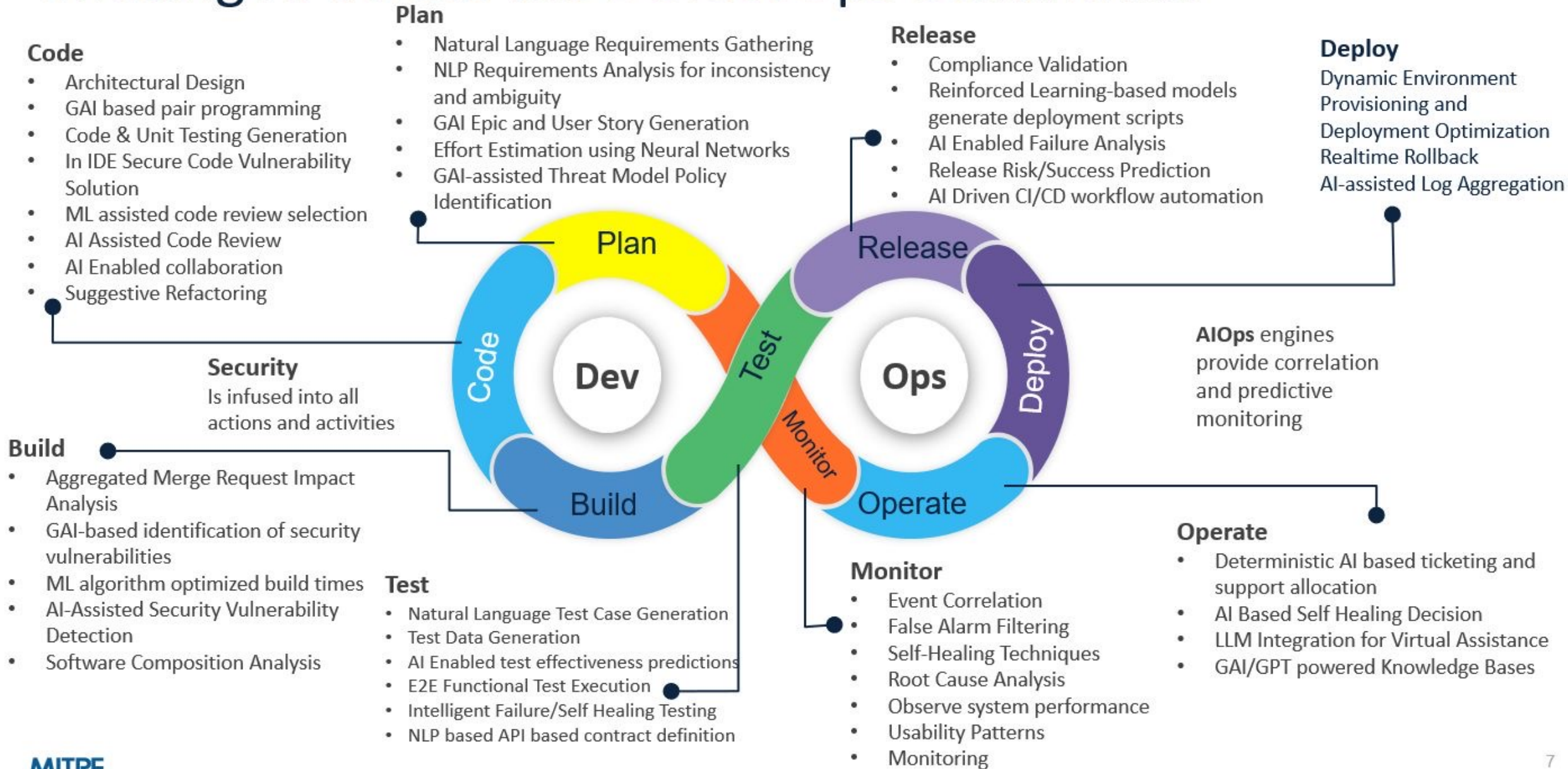
+ Sparx, not evaluated

Current capabilities:

Natural language query and summarization of repository artifacts and collaboration conversations

REQUIRES that you actually have definitions and collaborative information in your repository!

Infusing AI across the DevSecOps Continuum



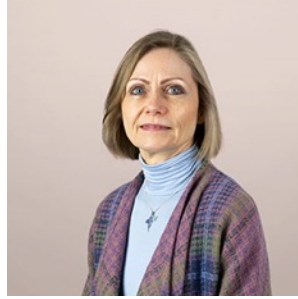
My sources



Rowan Curran
AI, ML, and data science



Mike Gualtieri
AI technologies, platforms, and practices



Martha Bennett
Emerging technologies



Jeff Pollard
Security and risk



Sucharita Kodali
Digital business strategy



Carlos Casanova
AIOps, observability



Michele Goetz
Data management and architecture



Alvin Nguyen
Technical and platform architecture



Julie Mohr
Knowledge management



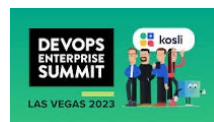
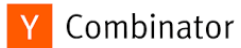
Brandon Purcell
Customer analytics and AI



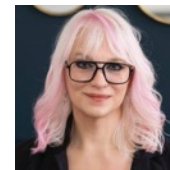
Alla Valente
Governance, risk and compliance



Brian Hopkins
Emerging tech



Dan McCreary



Tracy Bannon



Damon Edwards