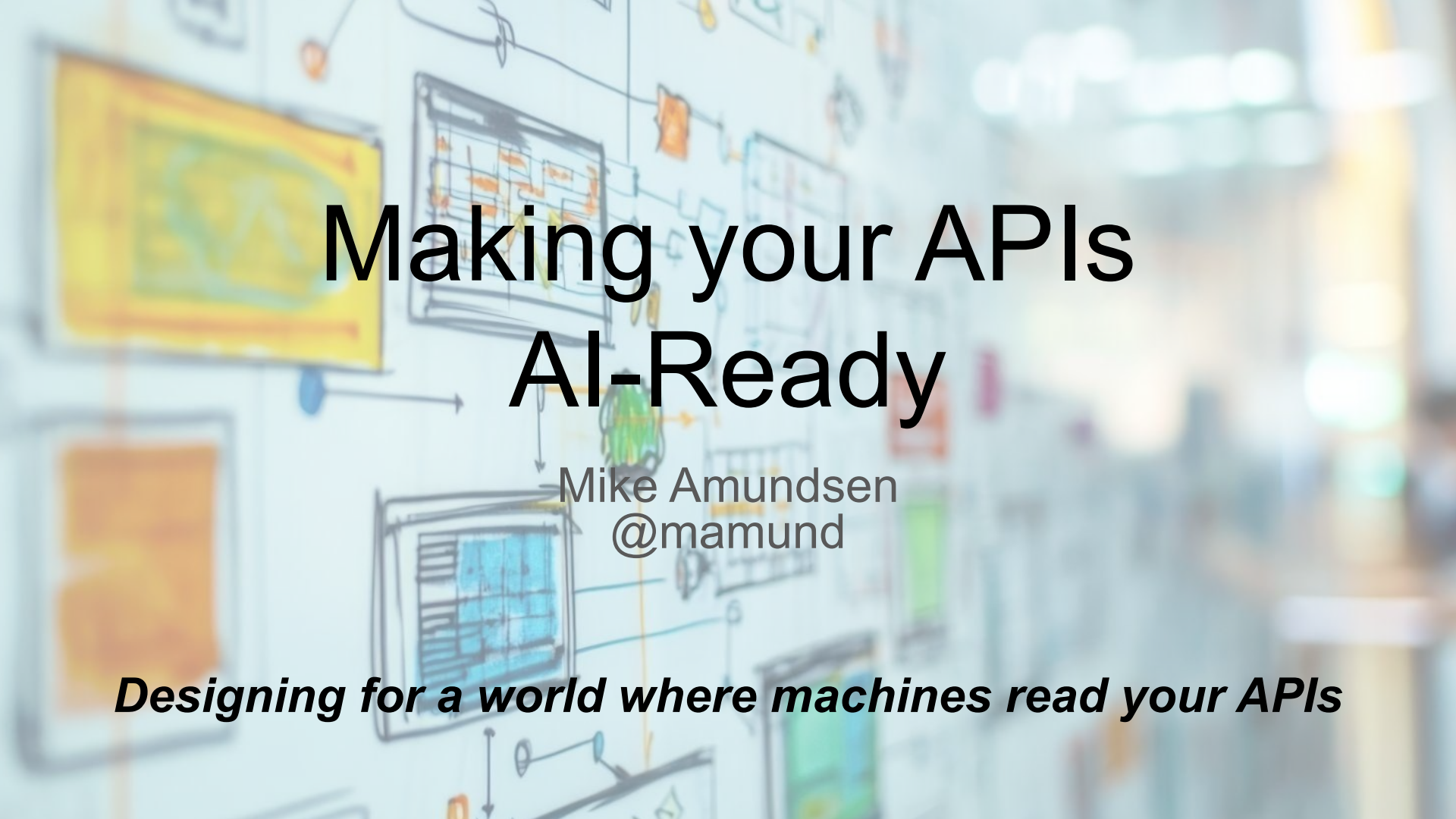
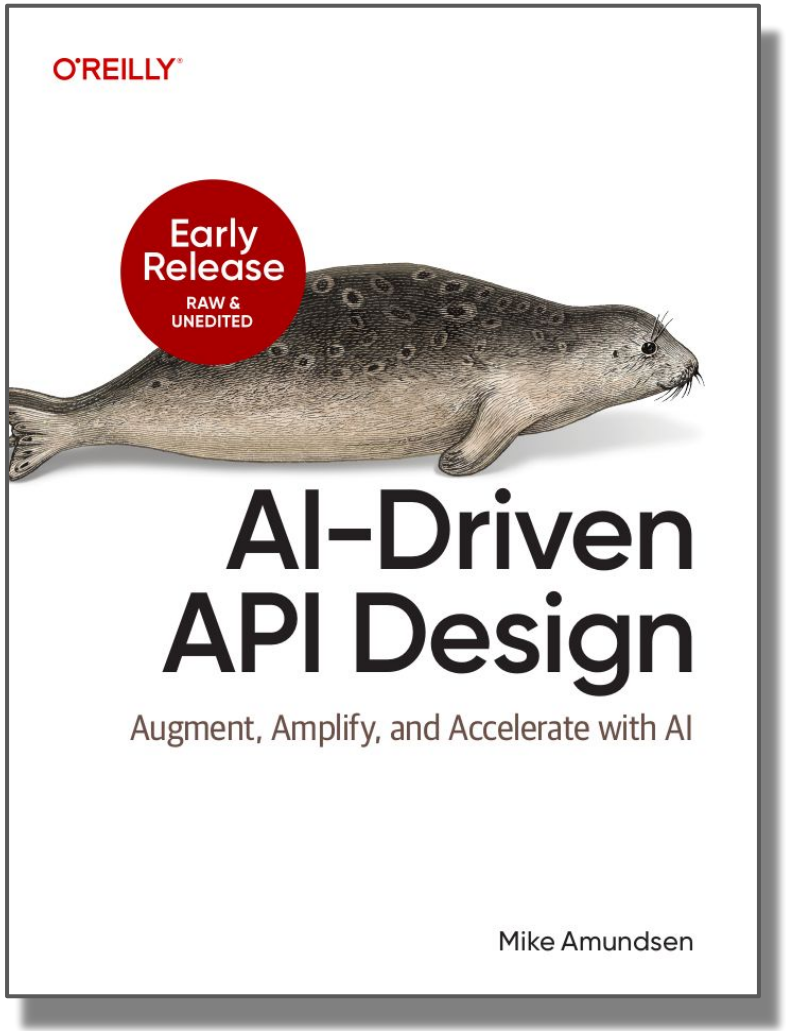


Making your APIs AI-Ready

Mike Amundsen
@mamund

Designing for a world where machines read your APIs


Mike Amundsen
@mamund


O'REILLY
Early Release
RAW & UNEDITED
AI-Driven API Design
Augment, Amplify, and Accelerate with AI
Mike Amundsen


The New Challenge

The New Challenge : Overview

- AI and the API Landscape
- Machines and APIs
- Not-so-AI-Ready APIs
- Five Shifts Ahead

The New Challenge : AI and the API Landscape

- AI is everywhere!
- APIs are no longer just for humans
- Agents, bots, orchestrations, etc. “see” differently

Similar to the shift from green screen to UI

The New Challenge : Machines and APIs

- Machines don’t learn the way we do
- Machines rely on context, not instructions
- Machines recognize patterns
- Machines look for “hooks”

Humans supply missing details, bots do not

The New Challenge : Not-So-AI-Ready APIs

- Agents mis-interpret your API responses
- Agents make selection errors when attempting actions
- Agents fail to repeat similar workflows
- Agents don’t give up, the hallucinate

Inconsistencies are “fixed” by humans,
but not by machines

The New Challenge : Five Shifts Ahead

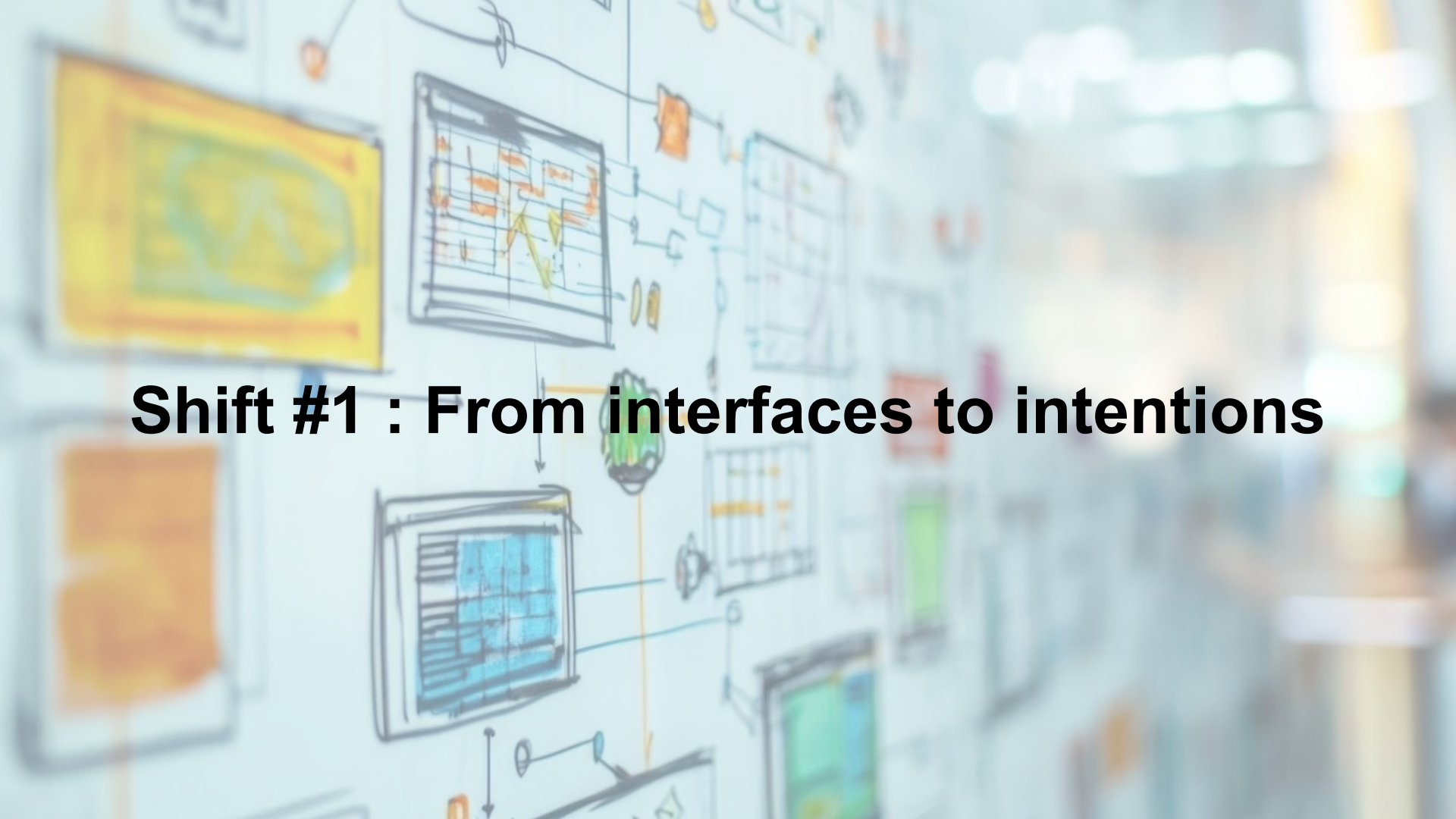
- We can improve things by focusing on five things
  - From interface to intention
  - Make context machine-readable
  - Standardize interactions, not just endpoints
  - Enable discovery
  - Support observation, iteration, and adaptation

We can improve our chances
by planning ahead

The New Challenge : Review

- AI and the API Landscape
  - APIs are no longer just for humans
- Machines and APIs
  - Humans supply missing details, bots do not
- Not-so-AI-Ready APIs
  - Inconsistencies are “fixed” by humans, but not by machines
- Five Shifts Ahead
  - We can improve our chances by planning ahead

Making your APIs AI-Ready
requires deliberate design



Shift #1 : From interfaces to intentions

From Interfaces to Intentions : Overview

- Why focus on Intent?
- Why this matters
- Designing for intention
- What we can do right now

From Interfaces to Intentions

"We must design for the way users behave, not for how we would wish them to behave."

– Don Norman

From Interfaces to Intentions

"We must design for the way machines behave, not for how we would wish them to behave."

– Don Norman

From Interface to Intention : Why Focus on Intent?

- Machines don’t care about UI
- They care about intended results
- Communicate purpose, not protocol details

Machine consumers don’t need UI,
they need intention.

From Interface to Intention : Why this matters

- Machines are not curious
- They don’t explore the UI/docs as humans do
- Machines recognize explicit instructions
- HTTP verbs are not sufficient for machines

Humans can deduce, a bot cannot

From Interface to Intention : Designing for Intent

- Machines benefit from designs that explain
- Use labels, descriptions, context clues
- Make meaning visible

Typical API example

```
{
  "name": "create",
  "method": "POST",
  "encType": "application/json",
  "fields": [ ... ]
}
```

AI-Ready example

```
{
  "title": "Register New User",
  "description": "Create a new user record by supplying required
identity fields. Agents should collect givenName and familyName,
then submit the completed object to initiate registration.",
  "name": "registerNewUser",
  "method": "POST",
  "encType": "application/json",
  "fields": [ ...]
}
```

Provide context and guidance for AI-clients

From Interface to Intention : What we can do right now

- Rename endpoints to reflect purpose of the action
- In OpenAPI, use **`operationId`** to express intent
- Provide plain-language descriptions for all actions

LLMs are geared for language, add more descriptions to your API

From Interfaces to Intentions : Review

- Why focus on Intent?
  - Machines don’t care about UI
- Why this matters
  - Machines recognize explicit instructions
- Designing for intention
  - Make meaning visible
- What we can do right now
  - Provide plain-language descriptions

Adding context is the easiest way to improve AI-agent success

Shift #2 : Make Context Machine Readable

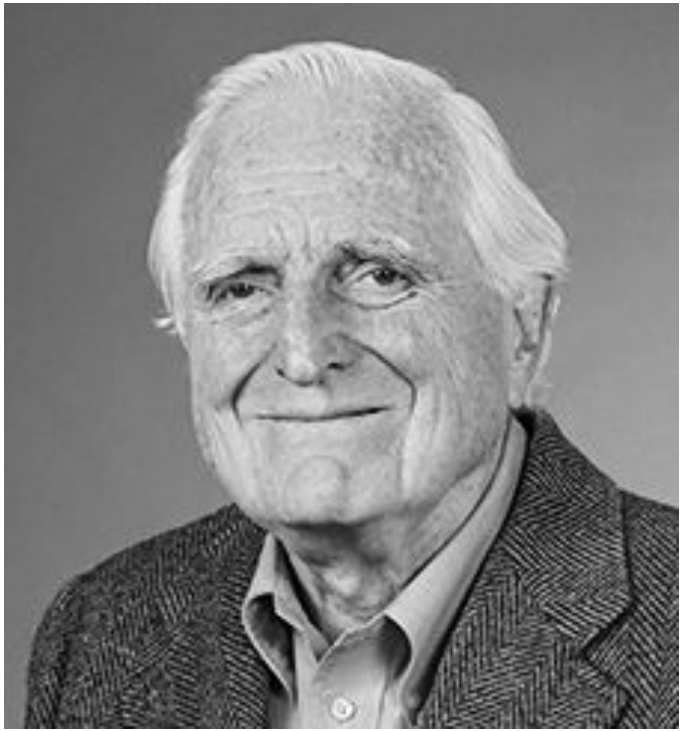
Make Context Machine-Readable : Overview

- Metadata is king
- What this means
- Focus on details
- What we can do right now

Make Context Machine-Readable

"The better we get at getting better, the faster we will get better."

– Douglas Engelbart



Make Context Machine-Readable : Metadata

- Be relentless about adding context
- Agents make better selections when they have more metadata
- Better context === better bots

Bots get better when the context gets better

Make Context Machine-Readable : What this means

- Think of your design assets as metadata
- Vocabulary, sets of operations, message schema
- Deeper descriptions are better than wider set of operations

Share all you can with your AI agents

Make Context Machine-Readable : Focus on details

- Be explicit about versioning
- Include possible “next steps” as detailed links/forms
- Provide hooks to additional metadata (e.g. machine docs)

Context is not just docs, it is code

Make Context Machine-Readable : What we can do now

- Add external metadata (APIs.json, ALPS, etc.)
- Embed capability statements (links, forms, inputs, etc.)
- Embrace domain design (consistent API vocabularies)

Pretty-print ☑


```
{
  "id": "5c137c58-f763-46bc-8a55-7aee5a8834c8",
  "title": "Write documentation",
  "description": "Create API docs from ALPS",
  "dueDate": "2025-06-01",
  "status": "active",
  "priority": "3",
  "assignedUser": "Alice",
  "_links": {
    "self": {
      "href": "/tasks/5c137c58-f763-46bc-8a55-7aee5a8834c8",
      "method": "GET",
      "args": []
    },
    "update": {
      "href": "/tasks/5c137c58-f763-46bc-8a55-7aee5a8834c8/edit",
      "method": "POST",
      "args": [
        "title",
        "description",
        "dueDate",
        "status",
        "priority",
        "assignedUser"
      ]
    },
    "setDueDate": {
      "href": "/tasks/5c137c58-f763-46bc-8a55-7aee5a8834c8/dueDate",
      "method": "PUT",
      "args": [
        "dueDate"
      ]
    },
    "updateStatus": {
      "href": "/tasks/5c137c58-f763-46bc-8a55-7aee5a8834c8/status",
      "method": "PUT",
      "args": [
        "status"
      ]
    },
    "assignUser": {
      "href": "/tasks/5c137c58-f763-46bc-8a55-7aee5a8834c8/assign",
      "method": "PUT",
      "args": [
        "assignedUser"
```

Documentation becomes part of API design

Make Context Machine-Readable : Summary

- Metadata is king
  - Better metadata == better bots
- What this means
  - Design assets are runtime assets
- Focus on details
  - Be explicit, verbose
- What we can do right now
  - Add metadata, embed capabilities, embrace domain-thinking

Metadata is the new interface

Shift #3 : Standardize Interactions

Standardize Interactions : Overview

- Consistency is power
- The benefits of consistency
- The downsides of inconsistent designs
- What we can do right now

Standardize Interactions

"A good architecture is not created in a vacuum."

– Roy Fielding

Standardize Interactions : Consistency is power

- Predictability is the cornerstone of autonomy
- Machines rely on patterns, not exceptions
- Consistency reduces overhead for agents

Agents recognize previous patterns

Standardize Interactions : The benefits of consistency

- Uniform patterns allow reuse of ai-agent strategies
- Reduce “surprise” to increase predictability
- Fewer design exceptions means fewer runtime errors

Consistent interactions means better results

Standardize Interactions : The downsides of inconsistency

- Inconsistent application of patterns confuses bots
- Irregular error responses increased overhead for agents
- Unexpected requirements (e.g. inputs) can lead to bugs

Inconsistency encourages bugs

Standardize Interactions : What we can do right now

- Adopt consistent error reports (RFC 9457 - Problem Details)
- Make retries possible and explicitly defined
- Use idempotent change operations (HTTP PUT)
- Avoid novel approaches (HTTP PATCH for all changes)
- Develop (and share) your company’s API Style Guide

```
{
  "type": "https://example.com/errors/invalid-input",
  "title": "Invalid input",
  "status": 400,
  "detail": "The submitted data is missing one or more required
fields.",
  "instance": "/registrations/12345",
  "invalidFields": [
    {
      "name": "givenName",
      "reason": "This field is required."
    }
  ]
}
```

Agents don’t appreciate creative solutions

Standardize Interactions : Summary

- Consistency is power
  - Machines rely on patterns
- The benefits of consistency
  - Reduce “surprise” to increase predictability
- The downsides of inconsistent designs
  - Inconsistency encourages bugs
- What we can do right now
  - Avoid novel approaches

Machines rely on patterns, not exceptions

Shift #4 : Enable Discovery

Enable Discovery : Overview

- Stop hiding capabilities
- Why this is important
- Beyond documentation
- What we can do right now

Enable Discovery

"Everything is deeply intertwingled"

– Ted Nelson

Enable Discovery : Stop hiding capabilities

- If bots can’t find it, they won’t use it
- Machines depend on in-message signals
- Discovery works at both the service and task level
- Bots are tuned for discovery

Discovery is a feature, not an afterthought

Enable Discovery : Why this is important

- Machine won’t look for external docs
- Supplying just the API docs is rarely enough
- Machines recognize and react to “signals”
- Think of your API as a “maze” the agent needs to solve

Help machines navigate the maze

Enable Discovery : Beyond documentation

- Prose documentation is good, but not sufficient
- Machines don’t make leaps of though or assumptions
- Without explicit signals, machines hallucinate

Enable Discovery


```
- {
    id: "fb4a2427-2cff-41b6-a61b-e96bccacfa0e",
    title: "Initial Task",
    description: "This is a sample task",
    dueDate: "2025-06-01",
    status: "active",
    priority: 3,
    assignedUser: "alice",
  - _links: {
      - self: {
            href: "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e",
            method: "GET"
        },
      - edit: {
            href: "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/edit",
            method: "POST",
          + args: [ … ]
        },
      - updateStatus: {
            href: "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/status",
            method: "PUT",
          - args: [
                "id",
                "status"
            ]
        },
      - assignUser: {
            href: "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/assignee",
            method: "PUT",
          - args: [
                "id",
                "assignedUser"
            ]
        },
      - setDueDate: {
```

Enable Discovery


```
- {
    id: "fb4a2427-2cff-41b6-a61b-e96bccacfa0e",
    title: "Initial Task",
    description: "This is a sample task",
    dueDate: "2025-06-01",
    status: "active",
    priority: 3,
    assignedUser: "alice",
  - _links: {
      - self: {
            href: "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e",
            method: "GET"
        },
      - edit: {
            href: "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/edit",
            method: "POST",
          + args: [ … ]
        },
      - updateStatus: {
            href: "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/status",
            method: "PUT",
          - args: [
                "id",
                "status"
            ]
        },
      - assignUser: {
            href: "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/assignee",
            method: "PUT",
          - args: [
                "id",
                "assignedUser"
            ]
        },
      - setDueDate: {
```

Enable Discovery


```
{
  "id": "fb4a2427-2cff-41b6-a61b-e96bccacfa0e",
  "title": "Initial Task",
  "description": "This is a sample task",
  "dueDate": "2025-06-01",
  "status": "active",
  "priority": 3,
  "assignedUser": "alice",
  "_links": {
    "self": {
      "href": "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e",
      "method": "GET",
      "description": "Retrieve the current details for this task."
    },
    "edit": {
      "href": "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/edit",
      "method": "POST",
      "args": [],
      "description": "Update one or more attributes of this task."
    },
    "updateStatus": {
      "href": "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/status",
      "method": "PUT",
      "args": [
        "id",
        "status"
      ],
      "description": "Change the status of this task."
    },
    "assignUser": {
      "href": "/tasks/fb4a2427-2cff-41b6-a61b-e96bccacfa0e/assignee",
      "method": "PUT",
      "args": [
        "id",
        "assignedUser"
      ],
      "description": "Assign this task to a specific user."
```

Include navigation metadata in responses

Enable Discovery : What we can do right now

- Maintain and use an API registry
  - Postman, Swagger, etc. for public APIs
  - Kong, WS02, etc. for internal APIs
- Manage vocabularies with tags for findability
- Publish and use API vocabularies

*Enable discovery **across** APIs, too.*

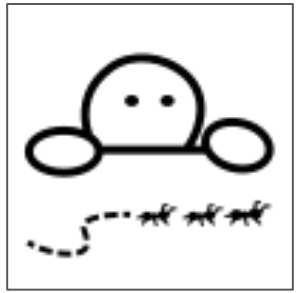
Enable Discovery : Summary

- Stop hiding capabilities
  - If they can’t find it, they won’t use it
- Why this is important
  - Help machines navigate “the maze”
- Beyond documentation
  - Include navigation metadata in responses
- What we can do right now
  - Maintain and use registries and vocabularies

Enabling discovery makes autonomy possible

Shift #5 : Observe, Adapt, Iterate

Observe, Adapt, Iterate : Overview



- Machines pay attention
- When agents fail
- How agents improve
- What we cans do right now

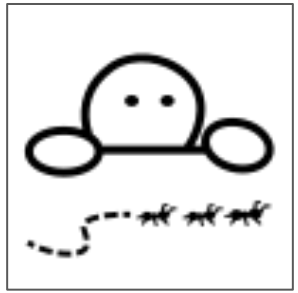
Observe, Adapt, Iterate

"Each pattern is a solution to a problem in a context"

– Christopher Alexander

Observe, Adapt, Iterate : Machines pay attention

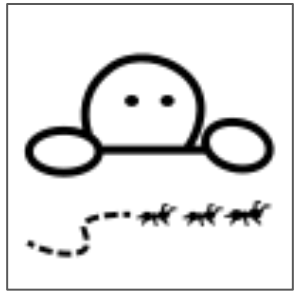
- LLMs are not passive consumers
- Machines work to optimize their path
- As context/content changes, machines adapt



You can influence machine learning by adjusting responses

Observe, Adapt, Iterate : When machines fail

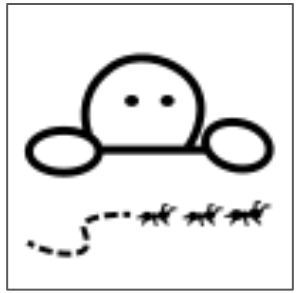
- Agents get confused by inconsistencies in responses
- Agents may repeat mistakes (get stuck)
- Agents will make things up if needed



Machine failures indicate design problems.

Observe, Adapt, Iterate : How agents improve

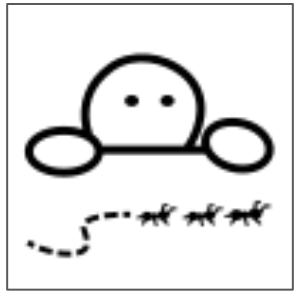
- Agents recognize structured content
- Agents notice local patterns in context
- Agents consolidate content with identifiers



Machines form habits

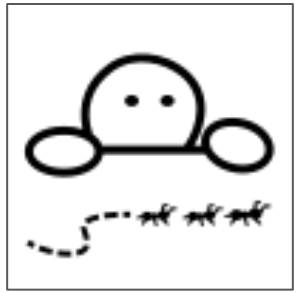
Observe, Adapt, Iterate : What we can do right now

- Set up telemetry / monitoring for your APIs
- Run “traces” on existing LLM conversations
- Note frequent API errors (indicate design problems)
- Identify repeated API sequences (hint to redesign API)

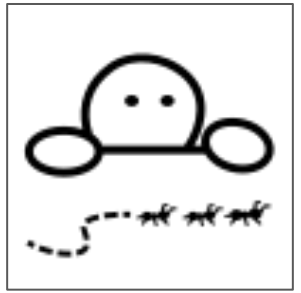


Observability/iteration is your laboratory

Observe, Adapt, Iterate : Summary



- Machines pay attention
  - Adjust responses to influence agents
- When agents fail
  - Failures indicate design problems
- How agents improve
  - Machines form habits
- What we cans do right now
  - Observe and adjust over time



The future belongs to adaptive systems


Bridging LLMs and APIs with MCP

Bridging LLMs & APIs with MCP : Overview

- What is MCP?
- What does an MCP server do?
- MCP and our five shifts
- What we can do right now

Bridging LLMs & APIs with MCP

"Systems work best when the information guiding them is clear, timely, and visible."

- Donella Meadows

Bridging LLMs & APIs with MCP : What is MCP?

- Model-Context Protocol (MCP) is a standard from Anthropic
- MCP makes it possible for LLMs to find and use APIs
- MCP is an adaptor

MCP makes discovery possible

Bridging LLMs & APIs with MCP : What does MCP do?

- MCP exposes capabilities and context (manifest)
- MCP standardizes LLM/API interaction (JSON RPC 2.0)
- Publishing an MCP server exposes your APIs to agents

Open

jokes-open-api.yaml

OS (C:) \Users\mikea\OneDrive\Desktop\20...main\2026-04-mcp-jokes-main\external\src

Save

```yaml
openapi: 3.1.0
info:
  title: Joke API
  version: 1.0.0
  description: A very simple API that returns a random tech/programming joke.

servers:
  - url: http://localhost:3001
    description: Local development server

paths:
  /joke:
    get:
      operationId: getJoke
      summary: Get a random joke
      description: Returns a single randomly selected tech/programming joke.
      responses:
        '200':
          description: A random joke response
          content:
            application/json:
              schema:
                type: object
                additionalProperties: false
                required:
                  - joke
                properties:
                  joke:
```

YAML Tab Width: 2 Ln 1, Col 1 IN

```
import express from "express";

const app = express();
const port = 3001;

const JOKES = [
  "There's no place like 127.0.0.1.",
  "Programmer: A machine that turns coffee into code.",
  "In order to understand what recursion is, you must first understand recursion.",
  "Why do programmers prefer dark mode? Because light attracts bugs.",
  "The best thing about a Boolean is even if you are wrong, you are only off by a bit.",
  "I just got fired from the keyboard factory. They said I wasn't putting in enough shifts.",
  "Debugging: Being the detective in a crime movie where you are also the murderer.",
  "There are 10 types of people in the world: those who understand binary and those who don't.",
  "A user interface is like a joke. If you have to explain it, it's not that good.",
  "Weeks of coding can save you hours of planning.",
  "There are only two hard things in Computer Science: cache invalidation, naming things, and off-by-one errors
." ,
  "UDP joke: I'd tell you one, but you might not get it.",
  "A SQL query walks into a bar, walks up to two tables and asks: 'Can I join you?'",
  "To understand recursion, you must first understand recursion.",
  "I would tell you a joke about the cloud, but it's highly available.",
  "Git commit early, commit often, and leave no merge conflict behind.",
  "It's not a bug – it's an undocumented feature.",
  "640K ought to be enough for anybody. (…probably not.)"
];

app.get("/joke", (_req, res) => {
  const joke = JOKES[Math.floor(Math.random() * JOKES.length)];
  res.json({ joke });
  console.log("called /joke");
});
```

mca@mamund-ws: ~/.../2026-04-mcp-jokes

```
import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
import { StdioServerTransport } from "@modelcontextprotocol/sdk/server/stdio.js";

// MCP server still exposes the same tech_joke tool.
// The difference is that the joke now comes from an external API.
const server = new McpServer({ name: "tech_joke", version: "1.0.0" });

const JOKE_API_URL = "http://localhost:3001/joke";

console.error("[tech_joke] Starting, registering tools...");

server.registerTool(
  "tech_joke",
  {
    title: "Random Tech Joke",
    description: "Returns a random technology / programming joke"
    // No inputSchema -> no arguments required
  },
  async () => {
    try {
      const response = await fetch(JOKE_API_URL);

      if (!response.ok) {
        throw new Error(`Joke API failed: ${response.status} ${response.statusText}`);
      }

      const data = (await response.json()) as { joke?: string };

      if (!data?.joke || typeof data.joke !== "string") {
        throw new Error("Joke API returned an invalid payload");
      }

      console.error("[tech_joke] tech_joke invoked ->", data.joke.slice(0, 40) + "...");
```

1,1 Top

mca@mamund-ws: ~/.../2026-04-mcp-jokes

```
import { Client } from "@modelcontextprotocol/sdk/client";
import { StdioClientTransport } from "@modelcontextprotocol/sdk/client/stdio.js";
import { ListToolsResultSchema, CallToolResultSchema } from "@modelcontextprotocol/sdk/types.js";

async function listTools(client: Client) {
  const result = await client.request({ method: 'tools/list' }, ListToolsResultSchema);
  console.log('\nTools:\n', JSON.stringify(result, null, 2));
  return result;
}

async function callTechJoke(client: Client) {
  const result = await client.request({
    method: 'tools/call',
    params: { name: 'tech_joke', arguments: {} }
  }, CallToolResultSchema);
  console.log('\ntech_joke =>', JSON.stringify(result, null, 2));
  return result;
}

async function main() {
  const transport = new StdioClientTransport({
    command: 'npx',
    args: ['tsx', 'src/server.ts']
  });

  const client = new Client({ name: 'mcp client', version: '1.0.0' });
  await client.connect(transport);

  await listTools(client);
  await callTechJoke(client);
  // Gracefully close the client & underlying transport so we exit cleanly.
  try {
    await client.close();
```

1,1 Top

```
[tech_joke] Starting, registering tools...
[tech_joke] Ready (stdio connected)

Tools:
 {
  "tools": [
    {
      "name": "tech_joke",
      "title": "Random Tech Joke",
      "description": "Returns a random technology / programming joke",
      "inputSchema": {
        "type": "object",
        "properties": {}
      },
      "execution": {
        "taskSupport": "forbidden"
      }
    }
  ]
}
[tech_joke] tech_joke invoked -> A SQL query walks into a bar, walks up t...

tech_joke => {
  "content": [
    {
      "type": "text",
      "text": "A SQL query walks into a bar, walks up to two tables and asks: 'Can I join you?'"
    }
  ]
}
mca@mamund-ws:~/.../external$
mca@mamund-ws:~/.../external$
```

```
{
  "joke": "Programmer: A machine that turns coffee into code."
}
```

```
{
  "joke": "Programmer: A machine that turns coffee into code.",

  "links": [
    {
      "rel": "another",
      "href": "/joke"
    },
    {
      "rel": "service",
      "href": "/"
    }
  ]
}
```

```
{
  "name": "Tech Joke API",
  "description": "Returns random technology and programming jokes.",
  "version": "1.0.0",

  "links": [
    {
      "rel": "self",
      "href": "http://localhost:3001/"
    }
  ],

  "actions": [
    {
      "rel": "joke",
      "name": "getTechJoke",
      "title": "Get a random tech joke",
      "description": "Returns one randomly selected technology or prog
      "href": "http://localhost:3001/joke",
      "method": "GET",
      "safe": true,
      "response": {
        "contentType": "application/json",
        "schema": {
          "type": "object",
          "properties": {
            "joke": {
              "type": "string"
            }
          },
          "required": [
            "joke"
          ]
        }
      }
    }
  ]
}
```

MCP exposes API capabilities and standardizes API interaction

Bridging LLMs & APIs with MCP : The Five Shifts

- Intentions : MCP focuses on describing API intent
- Context : MCP adds capability details to the LLM context
- Standardization : MCP standardizes API/LLM interactions
- Discovery : MCP advertises actions as LLM “tools”
- Observation : MCPs generate signals that guide improvement

MCP enables your shift to AI-Ready APIs

Bridging LLMs & APIs with MCP : What we can do now

- Add “intention-style” identifiers to your API operations
- Focus on clear, complete input schemas
- Standardize response formats (including errors)

Improved context metadata empowers your MCP servers

Bridging LLMs & APIs with MCP : Summary

- What is MCP?
  - An adaptor protocol for your APIs
- What does an MCP server do?
  - exposes capabilities, standardizes interactions
- MCP supports the five shifts
  - It enables your shift to AI-Ready APIs
- What we can do right now
  - Improving your context metadata empowers MCP servers

MCPs are the bridge between LLMs and your API


Bridging LLMs and APIs with MCP


Review

Review

- The New Challenge
- From interfaces to Intentions
- Make Context Machine-Readable
- Standardize Interactions
- Enable Discovery
- Observe, Adapt, Iterate
- Bridging LLMs & APIs with MCP

The New Challenge : Review

- AI and the API Landscape
  - APIs are no longer just for humans
- Machines and APIs
  - Humans supply missing details, bots do not
- Not-so-AI-Ready APIs
  - Inconsistencies are “fixed” by humans, but not by machines
- Five Shifts Ahead
  - We can improve our chances by planning ahead

From Interfaces to Intentions : Review

- Why focus on Intent?
  - Machines don’t care about UI
- Why this matters
  - Machines recognize explicit instructions
- Designing for intention
  - Make meaning visible
- What we can do right now
  - Provide plain-language descriptions

Make Context Machine-Readable : Review

- Metadata is king
  - Better metadata == better bots
- What this means
  - Design assets are runtime assets
- Focus on details
  - Be explicit, verbose
- What we can do right now
  - Add metadata, embed capabilities, embrace domain-thinking

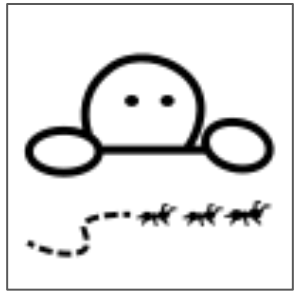
Standardize Interactions : Review

- Consistency is power
  - Machines rely on patterns
- The benefits of consistency
  - Reduce “surprise” to increase predictability
- The downsides of inconsistent designs
  - Inconsistency encourages bugs
- What we can do right now
  - Avoid novel approaches

Enable Discovery : Review

- Stop hiding capabilities
  - If they can’t find it, they won’t use it
- Why this is important
  - Help machines navigate “the maze”
- Beyond documentation
  - Include navigation metadata in responses
- What we can do right now
  - Maintain and use registries and vocabularies

Observe, Adapt, Iterate : Review



- Machines pay attention
  - Adjust responses to influence agents
- When agents fail
  - Failures indicat design problems
- How agents improve
  - Machines form habits
- What we cans do right now
  - Observe and adjust over time

Bridging LLMs & APIs with MCP : Review

- What is MCP?
  - An adaptor protocol for your APIs
- What does an MCP server do?
  - exposes capabilities, standardizes interactions
- MCP supports the five shifts
  - It enables your shift to AI-Ready APIs
- What we can do right now
  - Improving your context metadata empowers MCP servers

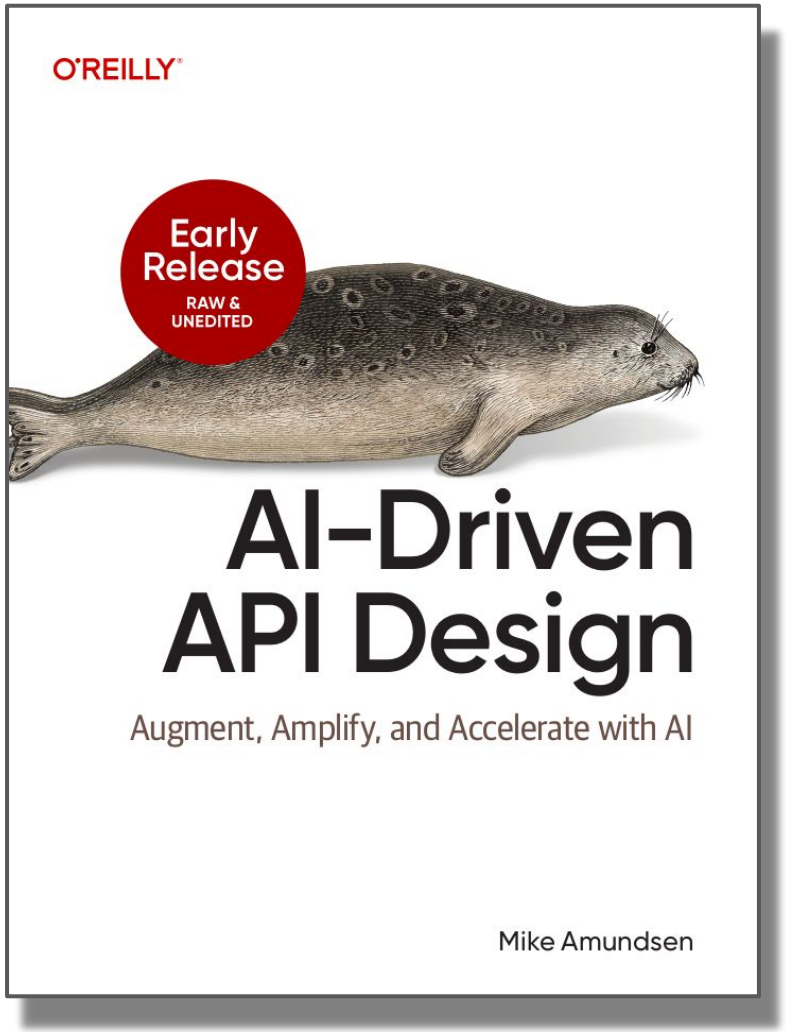
Review

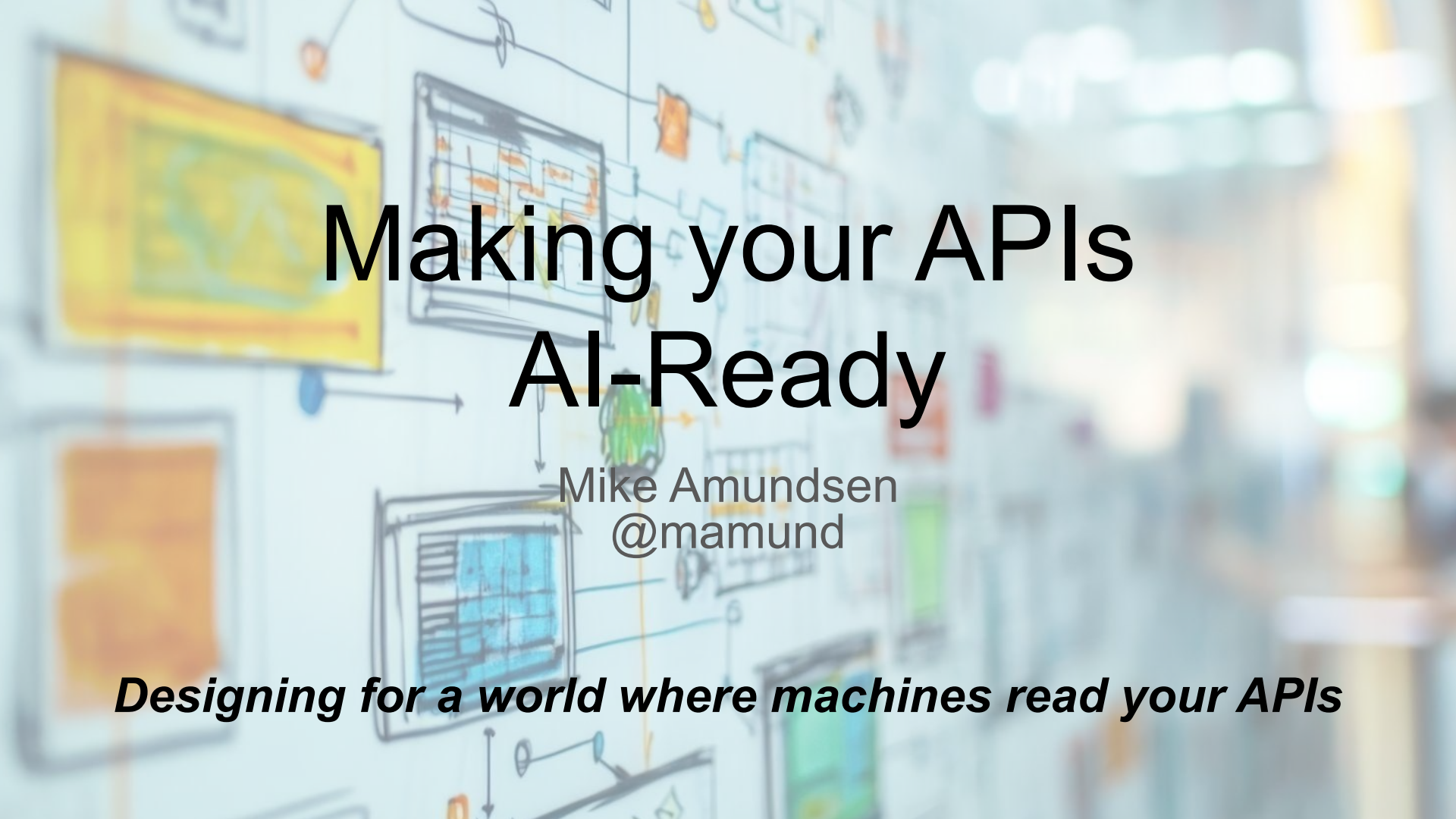
- The New Challenge
  - APIs are no longer only for humans
- From interfaces to Intentions
  - Machines don’t care about UI
- Make Context Machine-Readable
  - Metadata is king
- Standardize Interactions
  - Consistency is power

Review

- The New Challenge
  - APIs are no longer only for humans
- From interfaces to Intentions
  - Machines don’t care about UI
- Make Context Machine-Readable
  - Metadata is king
- Standardize Interactions
  - Consistency is power
- Enable Discovery
  - Stop hiding capabilities
- Observe, Adapt, Iterate
  - Adjust responses to influence agents
- Bridging LLMs & APIs with MCP
  - Enables your shift to AI-Ready APIs


Questions?


O'REILLY
Early Release
RAW & UNEDITED
AI-Driven API Design
Augment, Amplify, and Accelerate with AI
Mike Amundsen



Making your APIs AI-Ready

Mike Amundsen
@mamund

Designing for a world where machines read your APIs