Latest Trend on Software Development Techniques

Based on Technology Radar Vol 32, Techniques Section.

Martin Fowler Blog

ThoughtWorks Technology Radar

ThoughtWorks Technology Radar Techniques Section

MB Technology Radar

What is Technology Radar?

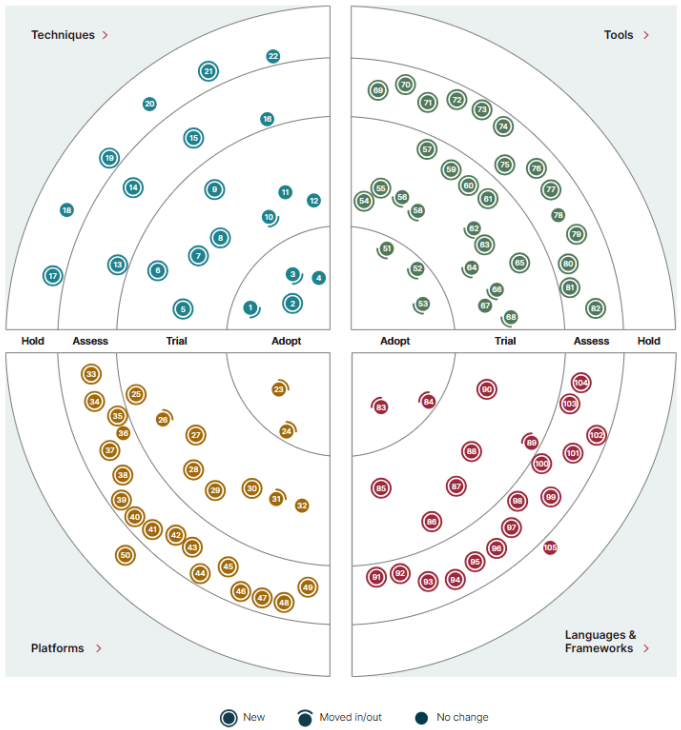
An opinionated guide to today's technology landscape.

Four Quadrants

- Techniques - Development practices & methods
- Tools - Software development tools
- Platforms - Infrastructure & platforms
- Languages & Frameworks - Programming languages & frameworks

Recommendation Ring

- Adopt - Proven, should seriously consider
- Trial - Ready for use, not completely proven
- Assess - Look at closely, evaluate fit
- Hold - Proceed with caution



Techniques Overview

Focus on development practices and methodologies

Techniques Overview

Adopt (4 techniques)

Data product thinking

Fuzz testing

Software Bill of Materials

Threat modeling

Trial (13 techniques)

API request collection as API product artifact

Architecture advice process

GraphRAG

Just-in-time privileged access management

Model distillation

Prompt engineering

Small language models

Using GenAI to understand legacy codebases

Assess (4 techniques)

AI-friendly code design

AI-powered UI testing

Competence envelope as a model for understanding system failures

Structured output from LLMs

Hold (6 techniques)

AI-accelerated shadow IT

Complacency with AI code

Local coding assistants

Replacing pair programming with AI

Reverse ETL

SAFe™

Adopt

We feel strongly that the industry should be adopting these items

Data Product Thinking

Ring: Adopt

What is it?

- Treat **data** as a **product** with its own lifecycle
- Focus on **consumer needs** and quality standards
- Work backward from **use cases**
- Comprehensive lifecycle management

Key Benefits

- Improved data discoverability
- Better governance and compliance
- AI-ready data preparation

Implementation

Modern Tools

DataHub, Collibra, Atlan, Informatica

Data Domain

Business & technical metadata

Approach

Consumer-centric design

Why Now?

Essential for scaling AI initiatives - Works with data mesh or lakehouse architectures

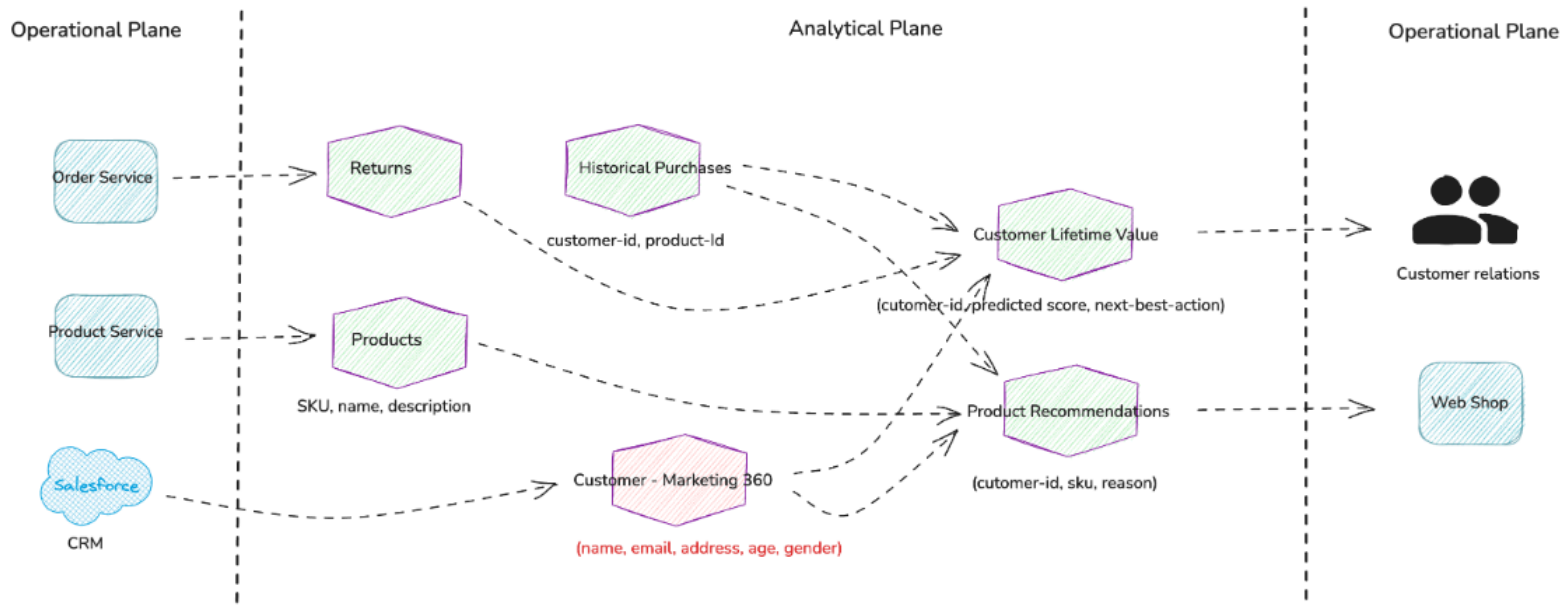
What are Data Products?

Discoverable	Data consumers should be able to easily explore available data products, locate the ones they need, and determine if they fit their use case.
Addressable	A data product should offer a unique, permanent address (e.g., URL, URI) that allows it to be accessed programmatically or manually.
Understandable (Self Describable)	Data consumers should be able to easily grasp the purpose and usage patterns of the data product by reviewing its documentation, which should include details such as its purpose, field-level descriptions, access methods, and, if applicable, a sample dataset.
Trustworthy	A data product should transparently communicate its service level objectives (SLOs) and adherence to them (SLIs), ensuring consumers can trust it enough to build their use cases with confidence.
Natively Accessible	A data product should cater to its different user personas through their preferred modes of access. For example, it might provide a canned report for managers, an easy SQL-based connection for data science workbenches, and an API for programmatic access by other backend services.
Interoperable (Composable)	A data product should be seamlessly composable with other data products, enabling easy linking, such as joining, filtering, and aggregation, regardless of the team or domain that created it. This requires supporting standard business keys and supporting standard access patterns.
Valuable on its own	A data product should represent a cohesive information concept within its domain and provide value independently, without needing joins with other data products to be useful.
Secure	A data product must implement robust access controls to ensure that only authorized users or systems have access, whether programmatic or manual. Encryption should be employed where appropriate, and all relevant domain-specific regulations must be strictly followed.

These are not Data Products!

Name	Reasons	Missing Characteristic
Data warehouse	Too large to be an independent composable unit.	▪ not interoperable ▪ not self-describing
PDF report	Not meant for programmatic access.	▪ not interoperable ▪ not native-access
Dashboard	Not meant for programmatic access. While a data product can have a dashboard as one of its outputs or dashboards can be created by consuming one or more data products, a dashboard on its own do not qualify as a data product.	▪ not interoperable ▪ not native-access
Table in a warehouse	Without proper metadata or documentation is not a data product.	▪ not self-describing ▪ not valuable on its own
Kafka topic	They are typically not meant for analytics. This is reflected in their storage structure — Kafka stores data as a sequence of messages in topics, unlike the column-based storage commonly used in data analytics for efficient filtering and aggregation. They can serve as sources or input ports for data products.	▪ not analytical data

Data Products



Fuzz Testing

Ring: Adopt

What is it?

- Feed software systems **invalid input**
- Observe behavior and responses
- Identify unexpected failures
- Long-standing technique gaining relevance

Why Important Now?

- More AI-generated code in production
- Growing complacency with AI outputs

Implementation

HTTP Endpoints

Bad requests → expect 4xx, but getting 5xx errors

Automation

Automated fuzzing tools

CI/CD

Integration with pipelines

Software Bill of Materials

Ring: Adopt

Popular Tools

- Syft
- Trivy
- Snyk

Why Critical?

Supply Chain Security

Vulnerability Management

Compliance Requirements

AI Systems Need SBOMs Too!

Regulatory Support

- UK Government's AI Cyber Security Code
- CISA's AI Cybersecurity Collaboration Playbook

Threat Modeling

Ring: Adopt

What is it?

- **Identify and classify** potential threats
- Regular practice throughout software lifecycle
- Works alongside other security practices
  - cross-functional security requirements to address common risks
  - automated security scanners for continuous monitoring
- Especially important as AI-driven development becomes more prevalent

Why Essential?

Avoid the **"security sandwich"** anti-pattern while maintaining agility

What is Security Sandwich?

- Traditonally, security is relied on up-front specification followed by validation at the end.
- This “Security Sandwich” approach is hard to integrate into Agile teams, since much of the design happens throughout the process, and it does not leverage the automation opportunities provided by continuous delivery.

STRIDE as a Practical Aid

- **S**poofed identity: *Is there Authentication? Should there be?* - Attackers pretending to be legitimate users through stolen credentials, phishing, or social engineering.
- **T**ampering with input: *What about nasty input?* - Attackers modifying data, code, or memory maliciously to break your system's trust boundaries.
- **R**epudiation: *Does the system show who is accountable?* - When something goes wrong, can you prove which user performed an action, or could they plausibly deny responsibility due to insufficient audit trails?
- **I**nformation disclosure: *Is sensitive data inappropriately exposed or unencrypted?* - Unauthorized access to sensitive data through poor access controls, cleartext transmission, or insufficient data protection.
- **D**enial of service: *What if we smash it?* - Attacks aiming at making the system unavailable to legitimate users by flooding or breaking critical components.
- **E**levation of privilege: *Can I bypass Authorization? Move deeper into the system?* - Attackers gaining unauthorized access levels, obtaining higher permissions than intended, or moving laterally through your system.

We use these STRIDE cards internally during threat modeling sessions either as printed cards or have them on screen. Another great way to help brainstorm, is to use GenAI. You don't need any fancy tool just prompt using a normal chat interface. Give some context on the dataflow and tell it to use STRIDE- most of the time you'll get a really helpful list of threats to consider.

STRIDE cards

Trial

Worth pursuing - ready for use but not completely proven

API Request Collection as Product Artifact

Ring: Trial

Concept

- Treat API collections as **API product artifacts**
- Guide developers through key workflows
- Stored in repository, integrated in release pipeline
- Part of comprehensive API-as-a-product strategy

Benefits

- Rapid developer onboarding
- Working examples with auth
- Realistic test data

Implementation

Tools

Postman | Bruno | Insomnia

📦 Repository Strategy

Version control alongside code

CI/CD Integration

Keep collections up-to-date automatically

Architecture Advice Process

Ring: Trial

Problem with ARBs

Architecture Review Boards create bottlenecks and hinder workflow

Key Benefits

- Optimize development flow without compromising architectural quality
- Collaborative decision making

How It Works

1. Decision Records

A structured doc capturing the proposed decision, advice gathered, alternatives considered, and final outcomes.

Architecture Advisory Forum (AAF)

A regular (e.g. weekly) open meeting where ADRs and decisions are discussed publicly. It includes representatives across teams (e.g. product, infra, compliance), promoting transparency and common understanding.

3. Team-sourced Architectural Principles

Guiding standards created collaboratively by delivery teams to steer decisions and help identify principle-conflicts in ADRs.

4. Technology Radar

A curated, organizational view of current and emerging technologies, informing and evolving the shared tech landscape.

Elements of an ADR	
name	**description**
title	which includes a unique identifier, and the decision itself (e.g. “ADR001 - Use AKS for Kubernetes Pods”)
status	typically “Draft”, “Proposed”, “Adopted”, “Superseded” and “Retired”
decision	the decision that has been taken in a few sentences (frequently bold or italicized so it stands out)
context	the forces and current contextual circumstances which have necessitated this decision
options considered	each option considered, described briefly, with pros and cons. (Typically the option proposed / adopted comes first in this list)
consequences	the ramifications of this decision, both positive and negative
advice	this reflects the raw outputs from following the Advice Process. It is here that all advice given is recorded. This ought to include the name of the advice giver, and the date the advice was given. This can frequently take the forms of comments, and if these are provided directly by the advice-giver, then recording the meta-data is automatic.

GraphRAG

Ring: Trial

What is GraphRAG?

- **Graph + Retrieval Augmented Generation**
- Enhanced knowledge representation using graph structures
- Better context understanding for AI systems
- Improved multi-hop reasoning capabilities

Key Advantages

- Relationship-aware retrieval
- Multi-hop reasoning support
- Enhanced knowledge connections

Implementation Approach

Graph Construction

Build knowledge graphs from your data

Enhanced Retrieval

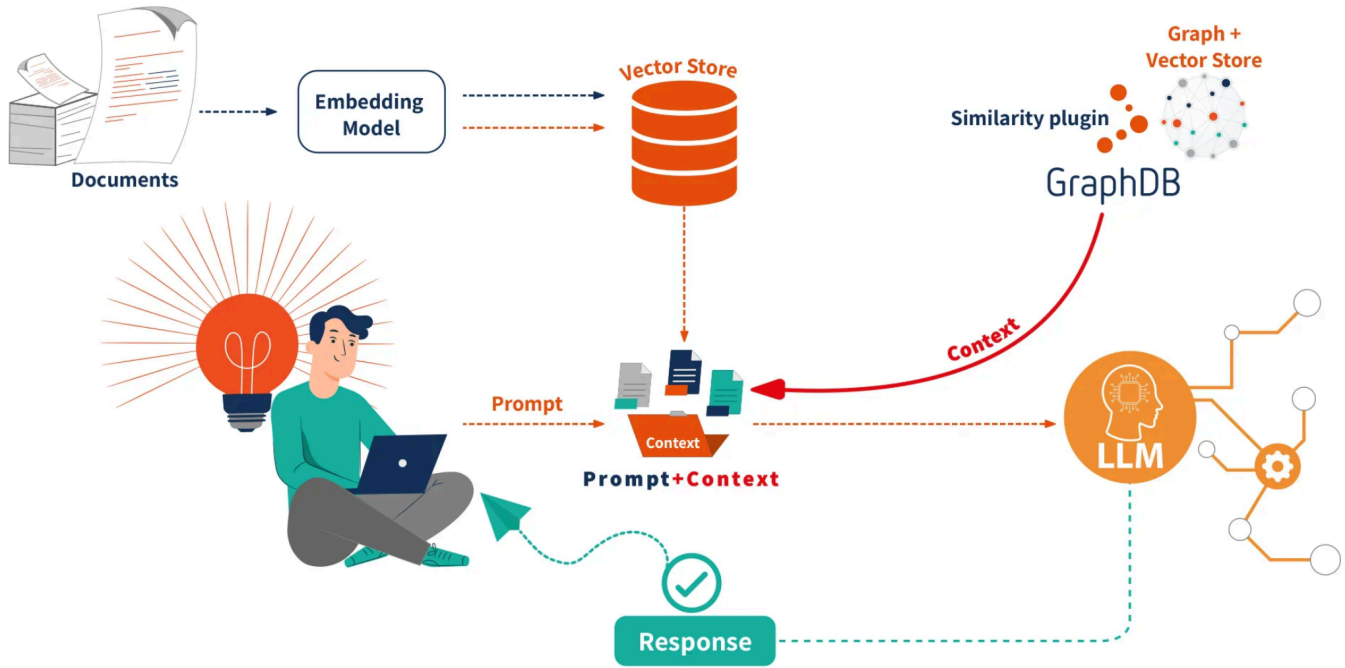
Traverse graph relationships for context

AI Integration

Feed enriched context to language models

Use Cases

- Complex document understanding
- Knowledge base question answering
- Research and analysis tasks


Documents
Embedding Model
Vector Store
Graph + Vector Store
Similarity plugin
GraphDB
Context
Prompt
Context
Prompt+Context
LLM
Response

Just-in-Time Privileged Access Management

Ring: Trial

Concept

- **Temporary elevated permissions**
- Grant access only when needed, for limited time
- Reduced attack surface and blast radius
- Zero-trust security model implementation

Security Benefits

- Minimized standing privileges
- Time-bounded access
- Comprehensive audit trail

Implementation

Principle of Least Privilege

Default to minimal permissions

Time-bound Access

Automatically expire elevated permissions

Approval Workflows

Require justification and approval

Monitoring

Real-time access monitoring and alerting

Model Distillation

Ring: Trial

What is Model Distillation?

- Transfer knowledge from **large teacher models** to **smaller student models**
- Enables **on-device inference** and cost-effective deployment
- Retains critical domain knowledge with **minimal accuracy loss**

Why It Matters

- Faster, low-latency inference
- Reduces cloud and compute costs
- Runs on edge and consumer devices

How It Works

1. Teacher Model

High-capacity model generates sample outputs or logits

2. Student Model

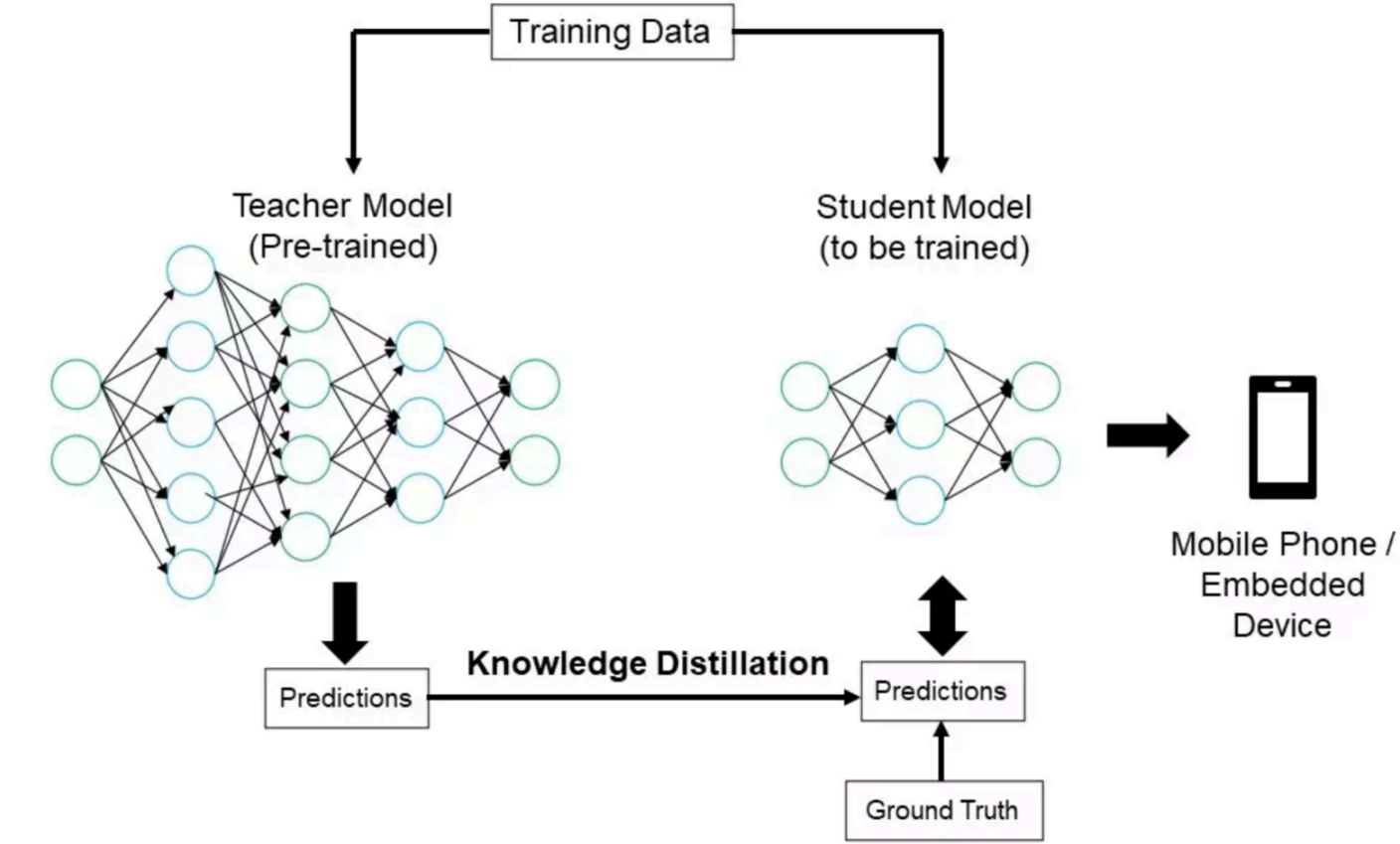
Lightweight model learns to replicate teacher behavior

3. Distillation Process

Train student using teacher's outputs

4. Optimization

Can be combined with quantization or pruning for added efficiency


Training Data
Teacher Model
(Pre-trained)
Student Model
(to be trained)
Mobile Phone /
Embedded
Device
Knowledge Distillation
Predictions
Predictions
Ground Truth

Prompt Engineering

Ring: Trial

What is Prompt Engineering?

- Designing and refining prompts for for generative AI models to produce high-quality, context-aware responses
- Helps mitigate hallucinations and improve response quality

Key Techniques

- Few-shot learning examples
- Chain-of-thought reasoning
- Parameter optimization

Insights

Zero-shot > Few-shot?

With reasoning models, fewer examples may perform better.

CoT Limitations

Chain-of-thought may hinder models already tuned for reasoning.

Strategic Model Choice

Balance output quality, latency, and token cost for your use case.

Prompt Engineering's Evolving Role

Still critical for hallucination control and agentic application design.

Small Language Models

Ring: Trial

Advantages

- Lower resource consumption
- Enhanced privacy (local deployment)
- Faster response times
- Cost-effective scaling

Use Cases

Text Classification

Sentiment analysis, content categorization

Named Entity Recognition

Extract specific information from text

Code Completion

IDE integrations and autocomplete

Domain Assistants

Specialized customer service bots

Using GenAI to Understand Legacy Codebases

Ring: Trial

The Challenge

- **Complex legacy systems** with poor documentation
- Knowledge gaps when original developers leave
- Difficult to understand interconnections
- Migration and modernization risks

GenAI Solutions

- Automated code documentation
- Business logic explanation
- Dependency mapping

Assess

Look at closely, but not necessarily trial yet

AI-Friendly Code Design

Ring: Assess • Focus: Code Quality

Core Concept

- **Structure code for AI understanding**
- Improve AI assistant effectiveness
- Better documentation and naming conventions
- Modular, well-organized architecture

Key Principles

- Comprehensive inline documentation
- Descriptive naming conventions
- Modular function design

Implementation Strategies

Self-Documenting Code

Write code that explains its purpose clearly

Clear Patterns

Use consistent architectural patterns

Explicit Dependencies

Make relationships between components clear

Benefits

Better AI code understanding → More accurate suggestions and refactoring

AI-Powered UI Testing

Ring: Assess • Focus: Test Automation

What is it?

- **Automated UI test generation** using AI
- Natural language test descriptions
- Visual regression detection capabilities
- Reduced manual testing effort
- Useful for testing legacy applications with missing selectors or applications that frequently change labels and click paths

Current Limitations

Reliability Concerns

AI-generated tests may be flaky

Context Understanding

Limited business logic understanding

Maintenance

Still requires human oversight

AI Capabilities

- Visual element recognition
- Natural language test creation

Competence Envelope as a Model for Understanding System Failures

Ring: Assess • Focus: System Reliability

Concept

- **competence envelope** — the boundary within which a system can function robustly in the face of failure
- Predict potential failure modes
- Improve overall system reliability
- **Residuality theory** - deliberately introducing stressors and analyzing how the system has adapted to historical stressors over time

Structured Output from LLMs

Ring: Assess • Focus: AI Integration

What is it?

- **Constrain model responses** to specific formats
- JSON Schema, Pydantic, Zod object definitions
- Reduce hallucinations and improve reliability
- **Enable better function calling and API interactions**

Key Benefits

- Guaranteed output format
- Better API integration
- Reduced hallucinations

Implementation Examples

JSON Schema

Define exact structure requirements

Pydantic Models

Python data validation and parsing

Zod Objects

TypeScript-first schema validation

Function Calling

Enable AI to use tools and APIs

Current Status

OpenAI now supports structured output natively

Hold

Proceed with caution

AI-Accelerated Shadow IT

Ring: Hold • Risk: High

The Problem

- **Non-coders building software** with AI assistance
- No-code platforms + AI API integration
- Ungoverned, potentially insecure applications
- Data scattered across more systems

Examples of Risk

- Chat messages → ERP API calls via AI
- Internal utility apps by non-developers
- AI as "duct tape" for integrations

Risks and Concerns

Security Vulnerabilities

Ungoverned applications with poor security

Data Sprawl

Information scattered across systems

Recommendation

Carefully weigh rapid problem-solving vs. long-term stability

Complacency with AI-Generated Code

Ring: Hold • Risk: Code Quality

Research Evidence

- **GitClear 2024 data**: Increased duplicate code and churn
- **Declining refactoring** activity in commit histories
- **Microsoft research**: AI confidence ↑ critical thinking ↓
- **"Vibe coding"** - minimal review of AI output

Quality Concerns

- More duplicate code
- Increased code churn
- Less refactoring activity

Specific Risks

Skill Atrophy
Loss of critical thinking and code review skills

Large Change Sets
AI agents generate code too large to review properly

Inappropriate Use
"Vibe coding" acceptable for prototypes, not production

Strong Caution

Maintain rigorous review processes for production code

Local Coding Assistants

Ring: Hold • Risk: Limited Capability

Why Organizations Want This

- **Code confidentiality concerns** with cloud AI
- No external data transmission
- Complete control over AI processing
- Compliance with security policies

Current Limitations

- Smaller, less capable models
- Limited context windows
- No tool integrations

What Works Locally

Code Completion

Xcode predictive completion, JetBrains full-line

Simple Queries

Basic coding questions and syntax help

Tools Available

Continue with Ollama, Qwen Coder

Recommendation

Proceed with low expectations - significant capability gap vs. cloud

Replacing Pair Programming with AI

Ring: Hold • Risk: Team Collaboration

The Question

- Can AI provide the same benefits as human pairing?
- GitHub Copilot calls itself "your AI pair programmer"
- Tempting to replace human collaboration with AI

What AI Provides

- Help getting unstuck
- Learning new technologies
- Faster tactical work

What AI Cannot Provide

Team Collaboration

No knowledge sharing between team members

Process Benefits

Low WIP, reduced handoffs, continuous integration

Collective Ownership

Shared understanding of codebase architecture

Key Point

AI helps **individuals**, pair programming helps **teams**

Reverse ETL

Ring: Hold • Risk: Architecture

What is Reverse ETL?

- Moving data **back** from analytics systems to transaction systems
- Opposite direction of traditional ETL
- Can be legitimate in transitional architectures
- Often used to centralize business logic

Why Hold?

Centralization Anti-pattern

Vendors use it to move business logic to their platform

Data Architecture Issues

Exacerbates problems of centralized data platforms

Complexity Growth

Creates sprawling, hard-to-manage data flows

Legitimate Use Cases

Multi-source data aggregation

Transitional architecture patterns

Recommendation

Exercise extreme caution when introducing reverse ETL flows

SAFe™ (Scaled Agile Framework)

Ring: Hold • Focus: Agile

Why Organizations Adopt SAFe

- **Complexity of becoming agile** at scale
- Hope for a "simple, process-based shortcut"
- Framework promises structure and governance
- Widespread industry adoption

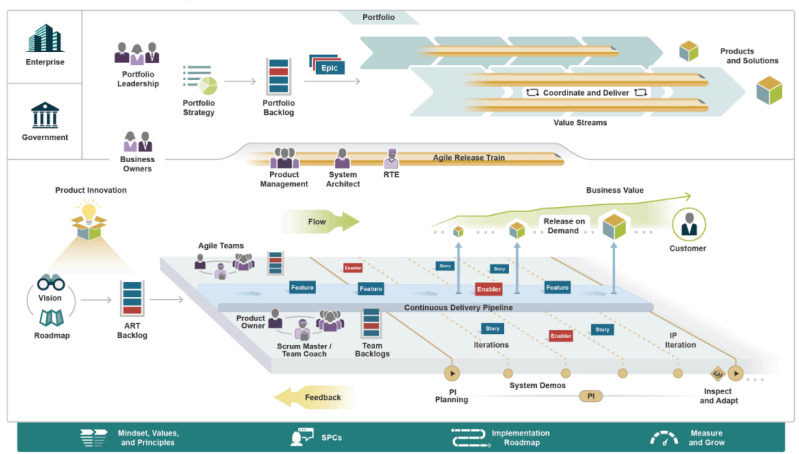
Observed Problems

- Over-standardized, phase-gated processes
- Promotes silos between teams
- Top-down control creates waste
- Discourages engineering talent creativity

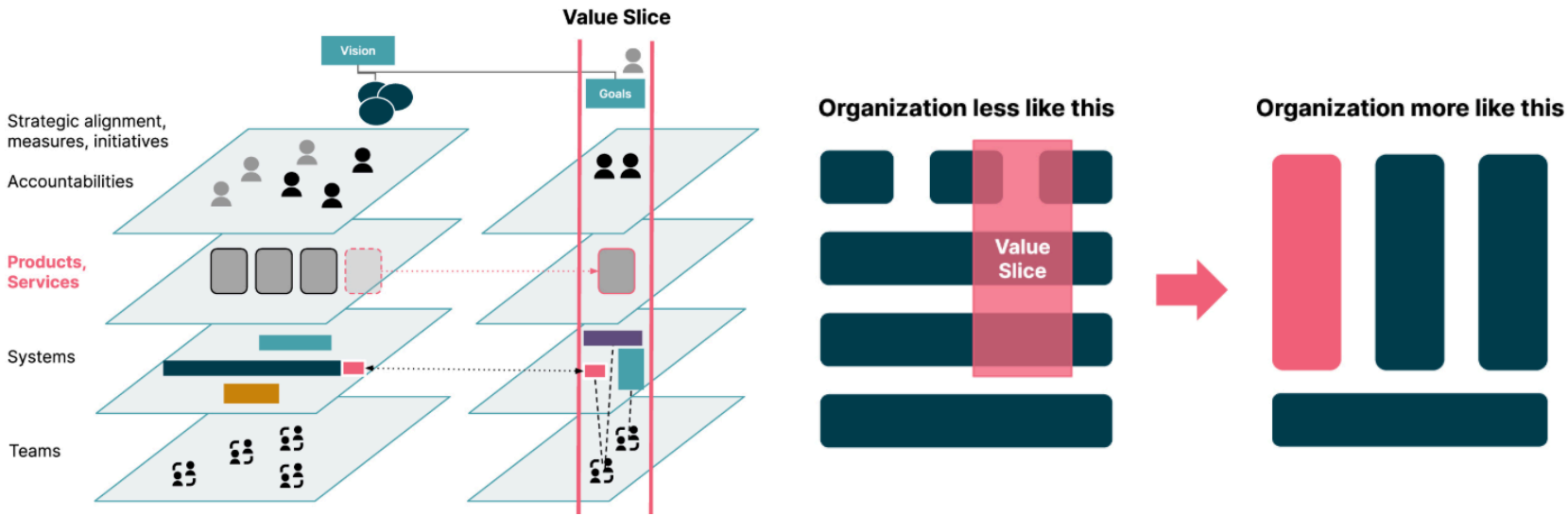
Recommendation

Leaner, value-driven approaches with comprehensive change programs

SAFe Big Picture



Value Slice


Value Slice
Vision
Goals
Strategic alignment, measures, initiatives
Accountabilities
Products, Services
Systems
Teams
Organization less like this
Value Slice
Organization more like this

Questions & Discussion

27 Techniques 4 Adopt 13 Trial 4 Assess 6 Hold

Full radar available at: thoughtworks.com/radar