Architecting Systems to Architecting Agility

Recurring Patterns Part 1

Part 3 of 7: Complexity & Requirements

Pattern 1 Complexity Creep

When Sophistication Becomes a Challenge

⚙ We often built sophisticated systems.

⚙ But sometimes **complexity became its own challenge**.

Complexity Example IoT Microservices

Interconnectedness Slowing Innovation

- ⚙ Recall large-scale IoT platforms with **interconnected microservices**.
- ⚙ Intended for flexibility but reality differed.
- ⚙ Adding a new sensor required changes across **numerous services**.
- ⚙ Significantly **slowing down**

innovation.

Complexity Example Kubernetes

Over-Engineering Risk

⚙️ Observed teams managing complex costly **Kubernetes clusters**.

⚙️ Extensive configurations even for apps whose scaling needs...

⚙️ ...might have been met by **simpler serverless** architectures initially.

Complexity The Drive vs Pragmatism

Impact on TCO

⚙ Drive to use the "**latest and greatest**" sometimes overshadowed pragmatism.

⚙ Pragmatic need for **simplicity** wasn't always prioritized.

⚙ Leading to higher **Total Cost of Ownership (TCO)**.

Pattern 2 The Requirements Labyrinth

Ambiguity A Frequent Friction Source

- Ambiguity in the early stages was a **frequent source of friction**.
- Ambiguity sown early yields a **bitter harvest** of rework later.

Requirements Example Mobility Epic

Noble Goals Lacking Clarity

- Experienced scenarios Epic defined to "enhance driver safety".
- Noble goal but lacked clear measurable **Key Results (OKRs)**.

Requirements Lacking Definition

User Voice BDD Criteria Missing

- ⚙ Lacked well-defined features in **user voice format** (As a...).
- ⚙ Often missing BDD-style **acceptance criteria** (Given/When/Then).

Requirements The Cost of Unclarity

Iterations Mismatches Rework

- Lack of initial clarity inevitably led to **multiple iterations**.
- Interpretation **mismatches** between POs and developers.
- Significant **rework** discovered only during late-stage testing or UAT.

Requirements Process Formality

DoR DoD Underutilized

⚙ Effective Definitions of Ready (DoR) and Done (DoD)...

⚙ Sometimes treated as **formalities** rather than crucial alignment tools.

⚙ Allowed poorly understood work to proceed causing waste.

Lesson

Impact of These Patterns

⚙ These patterns create significant drag on projects.

⚙ They increase cost delay value and frustrate teams.

Next

More Patterns

- ⚙ Next we'll explore the Efficiency Gap Communication Silos.
- ⚙ And the concept of the Tool Treadmill.

Series Index

Read the Full Article: From Architecting Systems to Architecting Agility...

All resources mentioned are available at **https://agilp.org/pdf/**

Read the Full Article on LinkedIn

Connect & Engage

LinkedIn: https://www.linkedin.com/in/amitabhrjha/

X (Twitter): https://x.com/amitabhrjha

Web: www.agilp.org

Disclaimer & Acknowledgments

The opinions expressed are my own & don't necessarily represent my employer's views. My perspective is constantly evolving, shaped by invaluable interactions with friends, colleagues, mentors, insightful authors, and industry influencers - thank you all! Much of this content, including these carousels, is co-created with AI co-pilots like ChatGPT, Gemini, and Grok. My intent is to synthesize knowledge and share it back with the community.