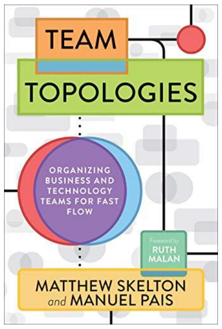
kindle



YOUR KINDLE NOTES FOR:

Team Topologies: Organizing Business and Technology Teams for Fast Flow

by Matthew Skelton, Manuel Pais, Ruth Malan

Free Kindle instant preview: https://a.co/9eRCVE0

74 Highlights

Highlight (Yellow) | Location 258

Teams are always works in progress,

Highlight (Yellow) | Location 271

there is huge value in agreeing to a coherent vocabulary and way of working together across teams to achieve good software delivery.

Highlight (Yellow) | Location 383

many organizations are still organizing their people and teams in ways that are counterproductive to modern software development and operations.

Highlight (Yellow) | Location 471

Team Topologies provides four fundamental team types—stream-aligned, platform, enabling, and complicated-subsystem—and three core team interaction modes—collaboration, X-as-a-Service, and facilitating.

Highlight (Yellow) | Location 493

“homomorphic force.”

Highlight (Yellow) | Location 508

The key takeaway here is that thinking of software architecture as a standalone concept that can be designed in isolation and then implemented by any group of teams is fundamentally wrong.

Highlight (Yellow) | Location 510

that is why monoliths need to be broken down (in particular, any indivisible software part that exceeds the cognitive capacity of any one team)

Highlight (Yellow) | Location 533

We need to put the team first, advocating for restricting their cognitive loads. Explicitly thinking about cognitive load can be a powerful tool for deciding on team size, assigning responsibilities, and establishing boundaries with other teams.

Highlight (Yellow) | Location 566

[Conway’s law] creates an imperative to keep asking: “Is there a better design that is not available to us because of our organization?”

Highlight (Yellow) | Location 611

“Given any [particular] team organization, there is a class of design alternatives which cannot be effectively pursued by such an organization because the necessary communication paths do not exist.”5

Highlight (Yellow) | Location 675

“Team assignments are the first draft of the architecture.

Highlight (Yellow) | Location 694

Organization Design Requires Technical Expertise

Highlight (Yellow) | Location 698

“if we have managers deciding . . . which services will be built, by which teams, we implicitly have managers deciding on the system architecture.”

Highlight (Yellow) | Location 702

it is very ineffective (perhaps irresponsible) for organizations that build software systems to decide on the shape, responsibilities, and boundaries of teams without input from technical leaders.

Highlight (Yellow) | Location 706

More than ever I believe that someone who claims to be an Architect needs both technical and social skills, they need to understand people and work within the social framework. They also need a remit that is broader than pure technology—they need to have a say in organizational structures and personnel issues, i.e. they need to be a manager too.12

Highlight (Yellow) | Location 714

not all communication and collaboration is good. Thus it is important to define “team interfaces” to set expectations around what kind of work requires strong collaboration and what doesn’t. Many organizations assume that more communication is always better, but this is not really the case.

Highlight (Yellow) | Location 717

What we need is focused communication between specific teams.

Highlight (Yellow) | Location 724

logically, two teams shouldn’t need to communicate based on the software architecture design, then something must be wrong if the teams are communicating.

Highlight (Yellow) | Location 907

The danger of allowing multiple teams to change the same system or subsystem is that no one owns either the changes made or the resulting mess.

Highlight (Yellow) | Location 911

Every part of the software system needs to be owned by exactly one team. This means there should be no shared ownership of components, libraries, or code. Teams may use shared services at runtime, but every running service, application, or subsystem is owned by only one team. Outside teams may submit pull requests or suggestions for change to the owning team, but they cannot make changes themselves. The owning team may even trust another team so much that they grant them access to the code for a period of time, but only the original team retains ownership.

Highlight (Yellow) | Location 944

Looking to reward individual performance in modern organizations tends to drive poor results and damages staff behavior.

Highlight (Yellow) | Location 988

If we stress the team by giving it responsibility for part of the system that is beyond its cognitive load capacity, it ceases to act like a high-performing unit and starts to behave like a loosely associated group of individuals, each trying to accomplish their individual tasks without the space to consider if those are in the team’s best interest.

Highlight (Yellow) | Location 997

A simple and quick way to assess cognitive load is to ask the team, in a non-judgmental way: “Do you feel like you’re effective and able to respond in a timely fashion to the work you are asked to do?”

Highlight (Yellow) | Location 1307

The first anti-pattern is ad hoc team design.

Highlight (Yellow) | Location 1311

The other common anti-pattern is shuffling team members.

Highlight (Yellow) | Location 1316

Organizations must design teams intentionally by asking these questions: Given our skills, constraints, cultural and engineering maturity, desired software architecture, and business goals, which team topology will help us deliver results faster and safer? How can we reduce or avoid handovers between teams in the main flow of change? Where should the boundaries be in the software system in order to preserve system viability and encourage rapid flow? How can our teams align to that?

Highlight (Yellow) | Location 1397

there is no “right” topology, but several “bad” topologies for any one organization.

Highlight (Yellow) | Location 1435

Non-blocking dependencies often take the form of self-service capabilities (e.g., around provisioning test environments, creating deployment pipelines, monitoring, etc.) developed and maintained by other teams.

Highlight (Yellow) | Location 1532

it is essential to detect and track dependencies and wait times between teams.

Highlight (Yellow) | Location 1537

three different categories of dependency: knowledge, task, and resource dependencies.

Highlight (Yellow) | Location 1618

reduced ambiguity around organizational roles is a key part of success in modern organization design.

Highlight (Yellow) | Location 1632

A “stream” is the continuous flow of work aligned to a business domain or organizational capability.

Highlight (Yellow) | Location 1676

independent streams of change.

Highlight (Yellow) | Location 1698

with continuous design we “treat ideas such as service, feedback, failure, and learning as first-class concepts,

---

Highlight (Yellow) | Location 1774

> Trying to fix engineering issues by mandating them from above is doomed to failure, as you really need buy-in from the folks working at the coalface.

Highlight (Yellow) | Location 1803

> The goal of this team is to reduce the cognitive load of stream-aligned teams working on systems that include or use the complicated subsystem. The team handles the subsystem complexity via specific capabilities and expertise that are typically hard to find or grow.

Highlight (Yellow) | Location 1831

> The platform team's knowledge is best made available via self-service capabilities via a web portal and/or programmable API

Highlight (Yellow) | Location 2160

> accidentally unclear boundaries between different teams and their responsibilities.

Highlight (Yellow) | Location 2170

> "distributed monolith" and results in teams lacking autonomy over their services, as almost all changes require updates to other services.

Highlight (Yellow) | Location 2182

> A joined-at-the-database monolith is composed of several applications or services, all coupled to the same database schema, making them difficult to change, test, and deploy separately.

Highlight (Yellow) | Location 2187

> A monolithic build uses one gigantic continuous-integration (CI) build to get a new version of a component.

Highlight (Yellow) | Location 2191

> A monolithic release is a set of smaller components bundled together into a "release."

Highlight (Yellow) | Location 2197

> monolithic model is software that attempts to force a single domain language and representation (format) across many different contexts.

Highlight (Yellow) | Location 2202

Monolithic thinking is “one size fits all” thinking for teams that leads to unnecessary restrictions on technology and implementation approaches between teams.

Highlight (Yellow) | Location 2206

enforcing standardization upon teams actually reduces learning and experimentation, leading to poorer solution choices.

Highlight (Yellow) | Location 2223

A fracture plane is a natural seam in the software system that allows the system to be split easily into two or more parts.

Highlight (Yellow) | Location 2344

Does the resulting architecture support more autonomous teams (less dependent teams) with reduced cognitive load (less disparate responsibilities)?

Highlight (Yellow) | Location 2475

When considering the relationship between any teams, a key decision is whether to collaborate with another team to achieve an objective or to treat the other team as providing a service

Highlight (Yellow) | Location 2511

“The roots of Toyota’s success lie not in its organizational structures but in developing capability and habits in its people.”

Highlight (Yellow) | Location 2523

The collaboration team mode is suitable where a high degree of adaptability or discovery is needed, particularly when exploring new technologies or techniques. The collaboration interaction mode is good for rapid discovery of new things, because it avoids costly hand-offs between teams.

Highlight (Yellow) | Location 2574

Relying on something “as a service” requires excellent work from the XaaS team(s)—not easy to achieve—but results in the delivery team having to understand less about non-core aspects of their work, allowing them to deliver more quickly

Highlight (Yellow) | Location 2593

The X-as-a-Service model works well only if the service boundary is well chosen and well implemented, with a good service-management practice from the team providing the service.

Highlight (Yellow) | Location 2763

small group of software and systems architects can be hugely effective within an organization when the remit of architecture is to discover, adjust, and reshape the interactions between teams, and therefore, the architecture of the system.

Highlight (Yellow) | Location 2768

“someone who claims to be an Architect needs both technical and social skills. . . . They also need a remit that is broader than pure technology—they need to have a say in business strategies, organizational structures, and personnel issues, i.e., they need to be a manager too.”

Highlight (Yellow) | Location 2792

Collaborate on potentially ambiguous interfaces until the interfaces are proven stable and functional.

Highlight (Yellow) | Location 2842

when designing modern organizations for building and running software systems, the most important thing is not the shape of the organization itself but the decision rules and heuristics used to adapt and change the organization as new challenges arise; that is, we need to design the design rules, not just the organization.

Highlight (Yellow) | Location 2851

Collaboration is good for rapid discovery and avoiding hand-offs and delays, but the downside is a higher level of cognitive load. Each side of the collaboration needs to understand more about the other side, so the team members have to retain more in their heads. However, this “collaboration tax” is worth it if the organization wants to innovate very rapidly.

Highlight (Yellow) | Location 2857

is best for situations where predictable delivery is more important than rapid discovery.

Highlight (Yellow) | Location 2865

Unnecessary collaboration is particularly expensive, especially as it can mask or hide deficiencies in underlying platforms or capabilities.

Highlight (Yellow) | Location 2879

deliberate use of a change in team interaction to force a beneficial change in delivery capability is the essence of strong, strategic technology leadership.

Highlight (Yellow) | Location 3052

reinforcing cycle of specialization

Highlight (Yellow) | Location 3056

when the team no longer holds a holistic view of the system; thus, it loses the self-awareness to realize when the system has become too large.

Highlight (Yellow) | Location 3094

provides a consistent developer experience (DevEx) for stream-aligned teams

Highlight (Yellow) | Location 3102

Increase flow predictability in higher-level business services (streams) through the use of a “platform wrapper” to “platformize” the lower-level services and

Highlight (Yellow) | Location 3116

By treating operations as rich, sensory input to development, a cybernetic feedback system is set up that enables the organization to self steer.

Highlight (Yellow) | Location 3145

Moving quickly relies on sensory feedback about the environment.

Highlight (Yellow) | Location 3168

Increasingly, software is less of a “product for” and more of an “ongoing conversation with” users.

Highlight (Yellow) | Location 3214

A second effect on performance of creating small, empowered units is to increase the likely speed of adaptation to new information.

Highlight (Yellow) | Location 3258

team is not simply a collection of individuals with the same manager but an entity with its own learning, goals, mission, and reasonable autonomy.

kindle

Highlight (Yellow) | Location 3277

large, up-front designs by software architects are doomed to fail unless the designs align with the way in which the teams communicate.

Highlight (Yellow) | Location 3279

Teams have a greater chance of innovating and supporting a system if they can understand the constituent parts and feel a sense of ownership over the code, rather than being treated like workers on an assembly line.

Highlight (Yellow) | Location 3286

Periods of technical and product discovery typically require a highly collaborative environment (with fading team boundaries) to succeed. But keeping the same structures when discovery is done (established technologies and product) can lead to wasted effort and misunderstandings.

Highlight (Yellow) | Location 3298

people feel empowered and safe to speak out about problems, and the organization expects to learn continuously.

Highlight (Yellow) | Location 3314

clarity of purpose and vision in an organization helps to set the context for everyone in the organization to act in ways that address that purpose.