

# IN5140 – DevOps, systems thinking and dealing with complexity

---



**Martin Løvteit Brox**

Enterprise Agile Coach

# Who am i?

- **Bachelor of Science – Computer Science NTNU**
- **Have been a consultant since 2007 and at Sopra Steria since 2013**
- **Experienced DevOps practitioner and Coach**

<https://www.linkedin.com/in/martin-løvteit-brox-30955027>

# Systems thinking

And ability to change

- Every system changes
  - > Some more than others
- Every system has resistance to change
  - > Some more than others
- A systems ability to change is a product of qualities from the systems culture, leadership and structure (processes)
  - > Processes that someone has dedicated their energy into creating and optimizing
- Optimized processes can also be the very things limiting the ability to change

Is the process optimized based on what we have known to be true in the past?

Or around what we do not yet know about the future?

# How do we deal with Complexity?

- Adaptable challenges call for adaptable problem solving
- While technical problems can be solved with technical solutions

# Or explained with Cynefin



Framework by Dave Snowden

# The world is changing faster

- More complexity
- More chaos

-> VUCA ( Volatility, Uncertainty, Complexity and Ambiguity )

# Consequences

- When we underestimate complexity and trying to solve adaptive challenges with technical solutions, we risk adding complexity and the creation of new complex problems.
  - > Technical/social dept
- All too often, we treat adaptive and complex challenges as if they are technical problems
- All too often, we try to simplify the world and our surroundings, and we apply short term fixes for short term gain

# Reflection

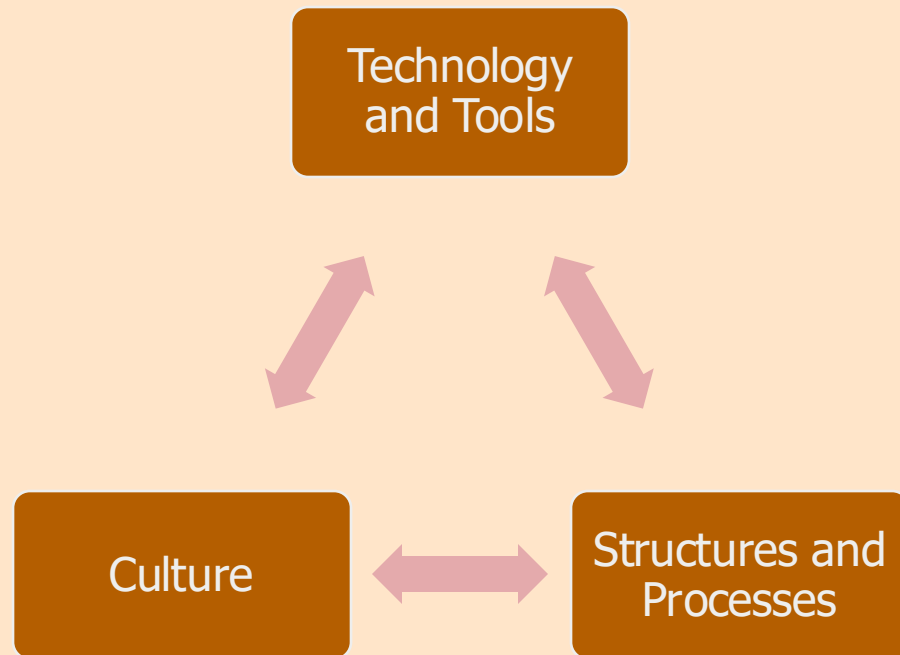
- Can you think of a time when you or someone else tried to solve a problem, believing it to be in a different problem domain? What happened?
- Discuss with the person sitting next to you (2 minutes each)

# Conways law

- Conways law is an observation that a system will mirror the communication patterns of the organization building the system
- The human and technical factors of system design are intertwined and treated as interdependent parts of a complex system
  - > Sociotechnical architecture
- When there is a gap between the desired architecture and how the organization actually functions, there will be tension. And the organization always wins.
  - > Peter Druckers famous "Culture eats strategy for breakfast" also applies to architecture

# Now DevOps

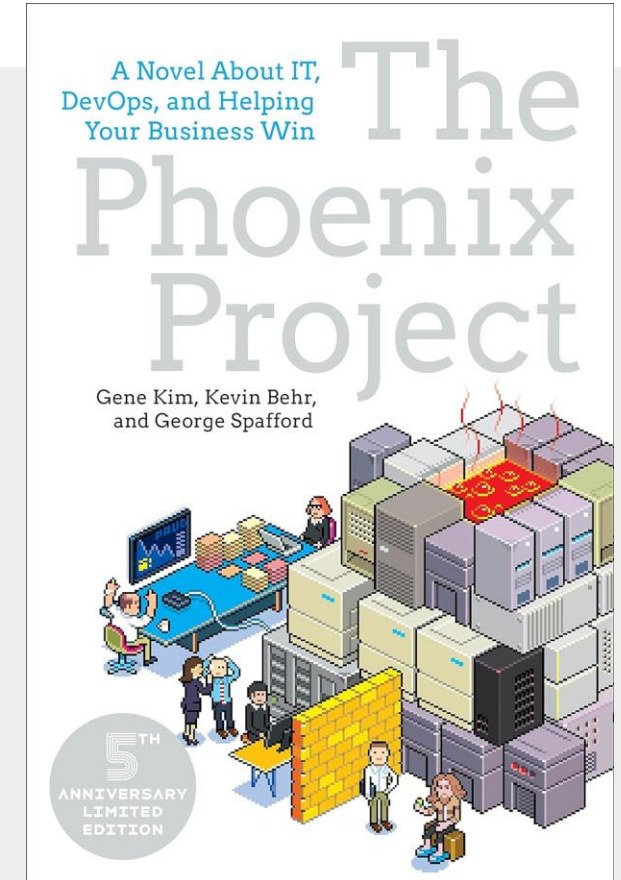
- DevOps is a software development approach well suited for dealing with complexity
- Consisting of several dimensions. Its more than just the tooling.
- Like:



# The Three Ways of DevOps

As made famous by «The DevOps Handbook» og «The Phoenix Project»

- The First Way: Flow/Systems Thinking
- The Second Way: Amplify Feedback Loops
- The Third Way: Culture of Continual Experimentation and Learning



# How DevOps has evolved

- The term DevOps became a thing in 2008
- But practices and principles associated with DevOps has existed for longer
- The term plays on the combination of Development and Operations
  - And the breaking down of silos
  - Other variations:
    - FinOps
    - DevSecOps
    - Product Teams
- Every handover from team to team is a transaction with a cost, i.e. waiting time and loss of information
- And the different organizational units often has different ways of measuring success. In example:
  - Dev -> Lead time and output
  - Ops -> Reliability

# The four key metrics

For measuring software delivery performance

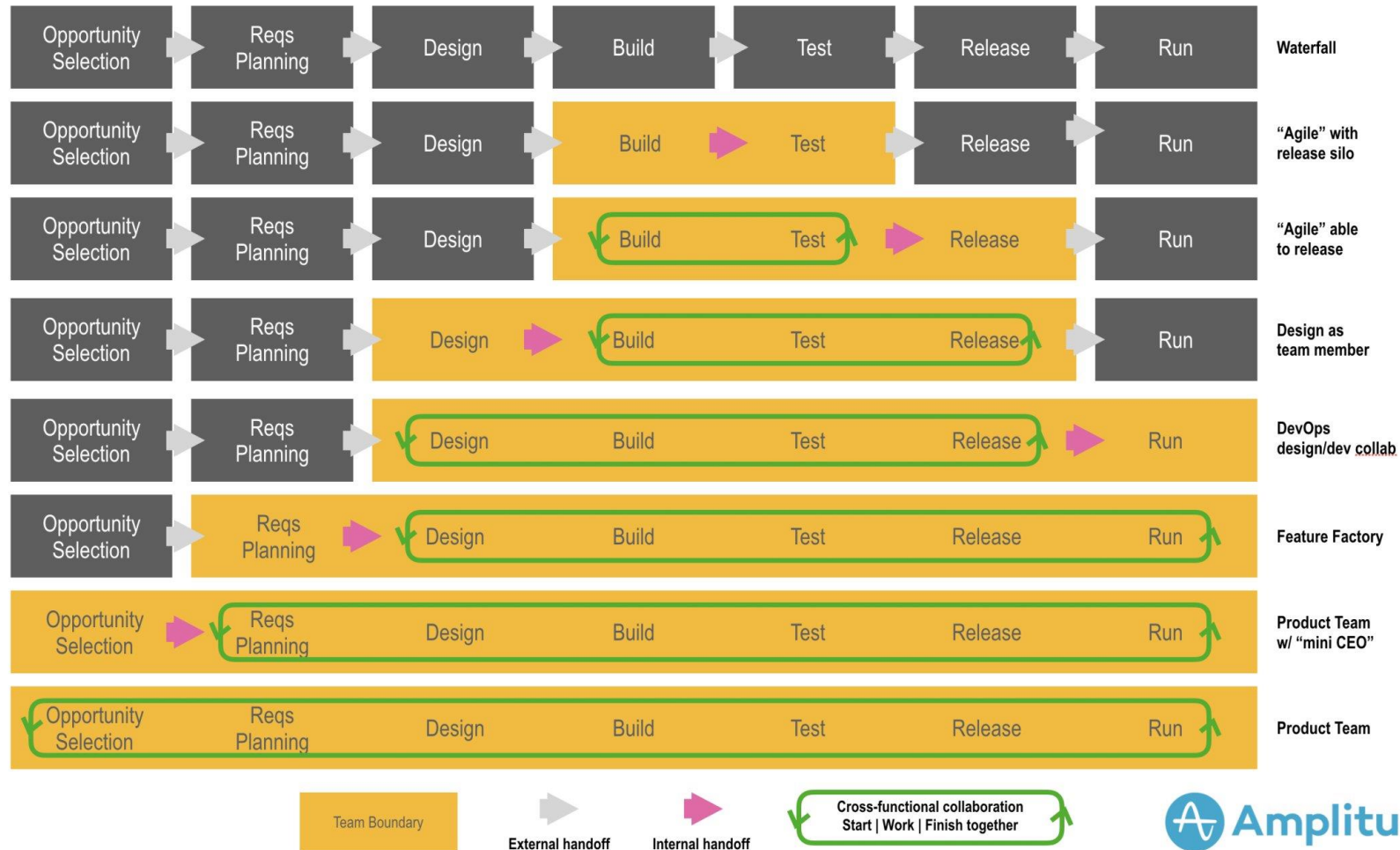
| Software delivery performance metric                                                                                                                                                                                                                                                                                         | Low                                            | Medium                                   | High                                 |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|------------------------------------------|--------------------------------------|
| <b>Deployment frequency</b><br>For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?                                                                                                                                                    | Between once per month and once every 6 months | Between once per week and once per month | On-demand (multiple deploys per day) |
| <b>Lead time for changes</b><br>For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?                                                                                                   | Between one month and six months               | Between one week and one month           | Between one day and one week         |
| <b>Time to restore service</b><br>For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?                                                                       | Between one week and one month                 | Between one day and one week             | Less than one day                    |
| <b>Change failure rate</b><br>For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)? | 46%-60%                                        | 16%-30%                                  | 0%-15%                               |

- Notice the fact that there are 2 measurements for flow, and 2 for reliability
- Try it out for yourself!
  - <https://dora.dev/quickcheck/>

Source: Accelerate | State of DevOps 2022

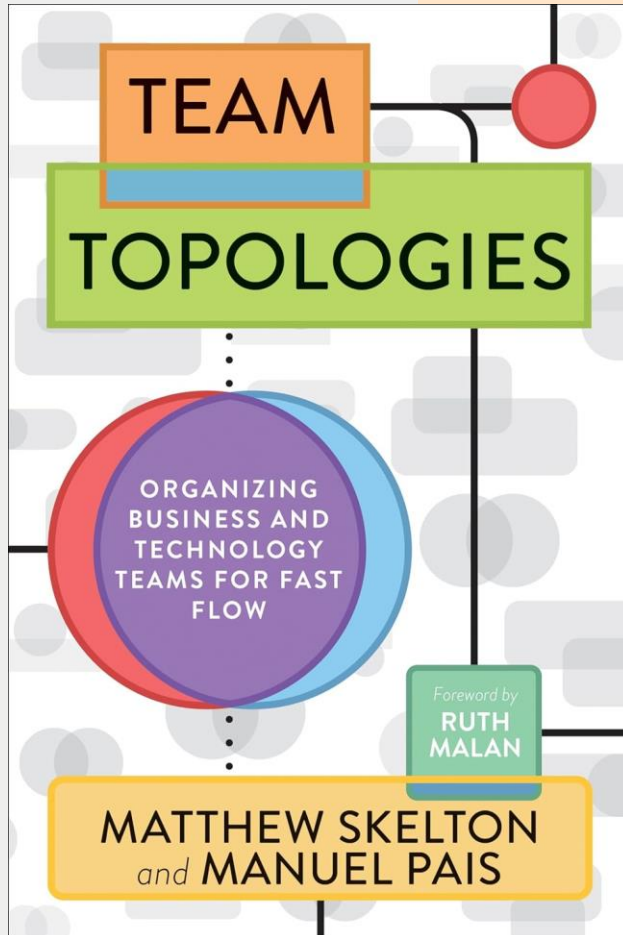
# Team boundaries

And cross functionality on different levels



# Team Topologies

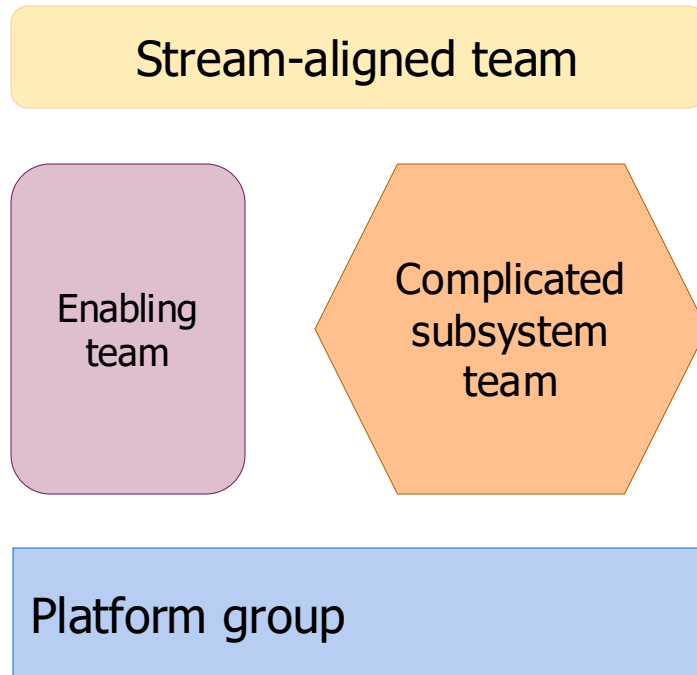
Matthew Skelton & Manuel Pais, 2019



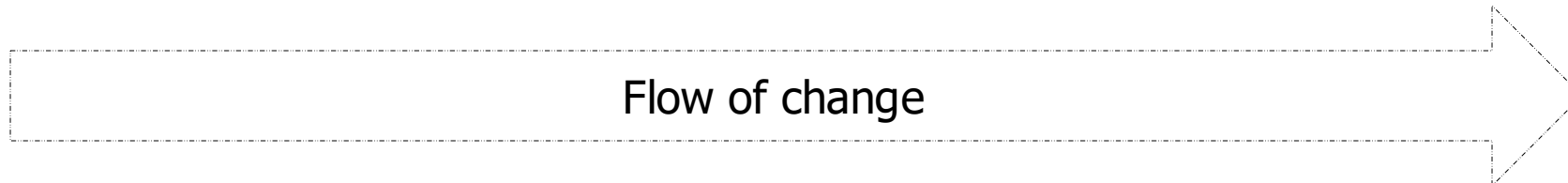
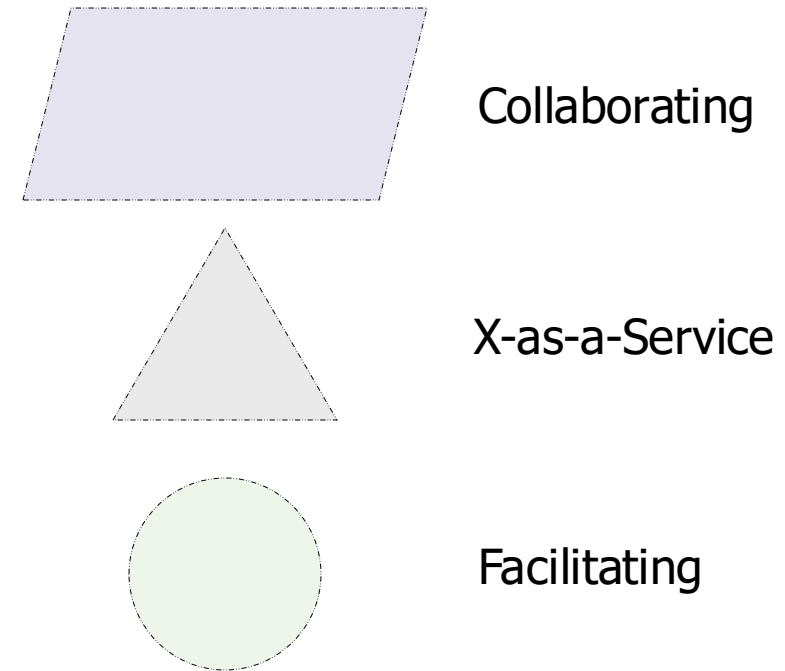
- To organize for flow, they have defined
  - 4 types of teams
  - 3 interaction modes
  - Principles and practices for fast and sustainable stream of flow
  - Reduce cognitive load for developers

# Team Topologies

## Fundamental Team Types



## Team Interaction Modes



# Platform Engineering

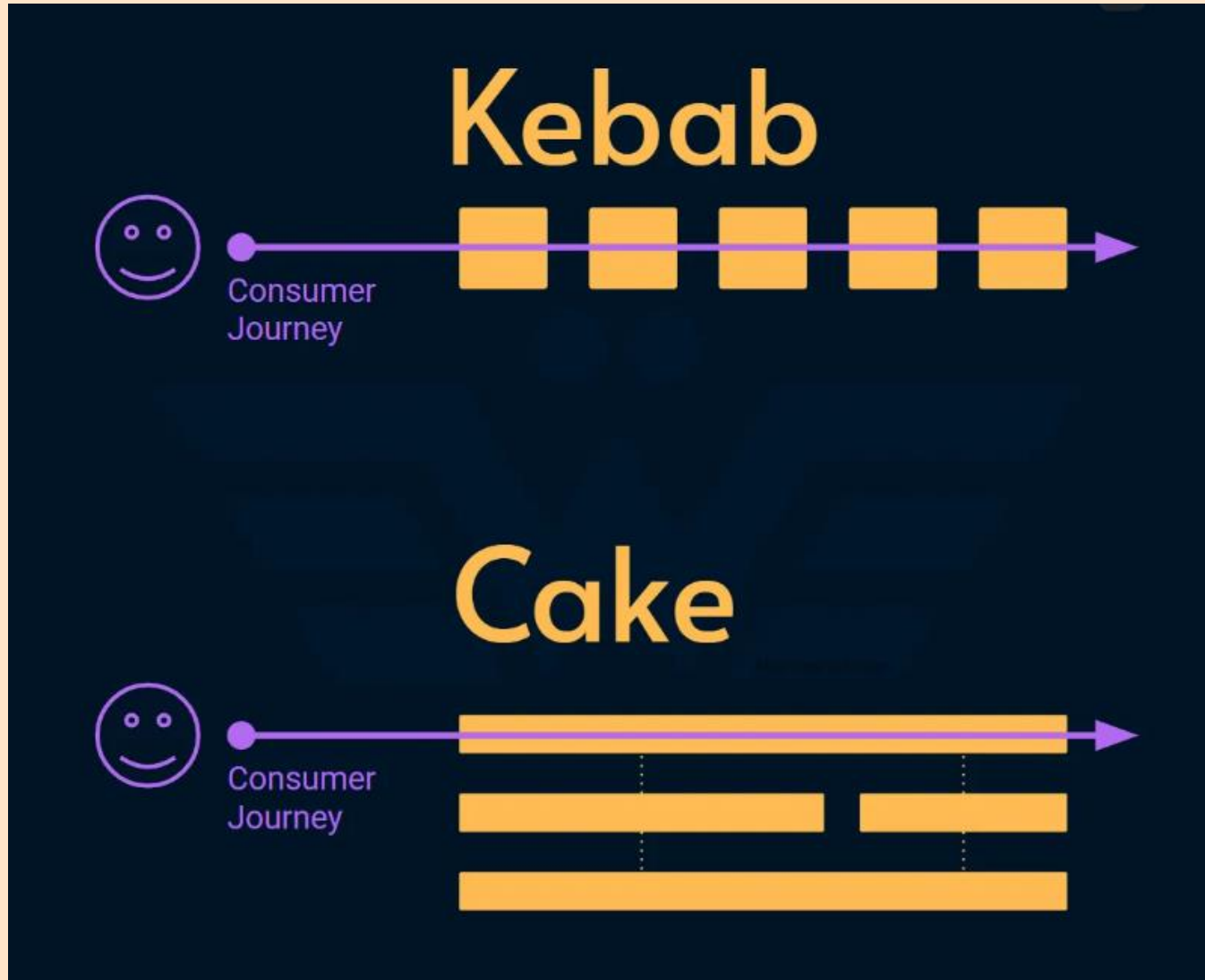
*"Platform engineering is the discipline of designing and building self-service capabilities to minimize cognitive load for developers and to enable fast flow software delivery. Platform teams deliver shared infrastructure platforms to internal users"*

(Puppet, State Of Platform Engineering Report 2023)

# Some common DevOps mistakes

- Oversimplification
- Trying to automate what is fluid and unpredictable
- Overly focusing on tools, failing to change in other dimensions
- Like with process optimization in general, the absence of system thinking leads to suboptimization
- And like so often, there are tradeoffs to be made
  - What may happen when flow of value is your sole focus?

# Kebab organizations vs Cake organizations



By Alex Ewerlöf

<https://blog.alexewerlof.com/p/kebab-vs-cake>

# News from Dora State of DevOps Report 2024

- AI boost productivity on individual level.
- But actually hurts software delivery performance in total.
- Much due to AI assisted coding generating larger batches of code, and larger changesets introduce risk.
- And also highlighted the importance of Transformational Leadership  
"Transformational leadership is a model in which leaders inspire and motivate employees to achieve higher performance by appealing to their values and sense of purpose, facilitating wide-scale organizational change. "

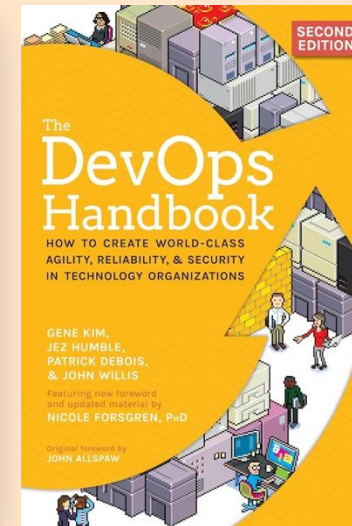
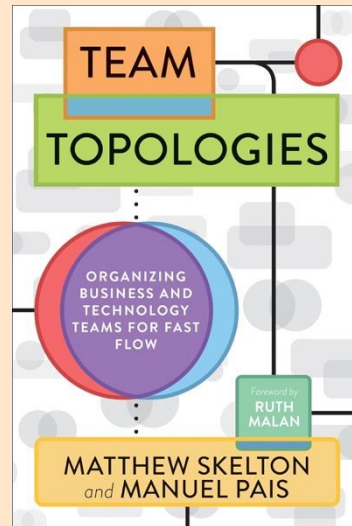
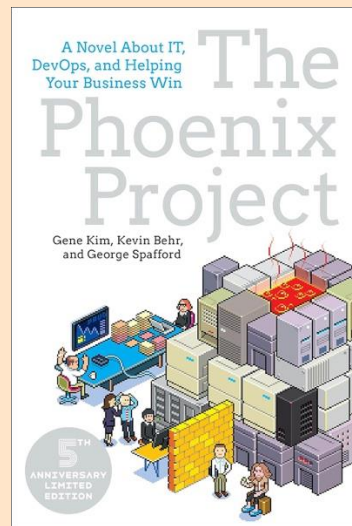
Download and read at <https://dora.dev/research/2024/dora-report/>

# What we have learned



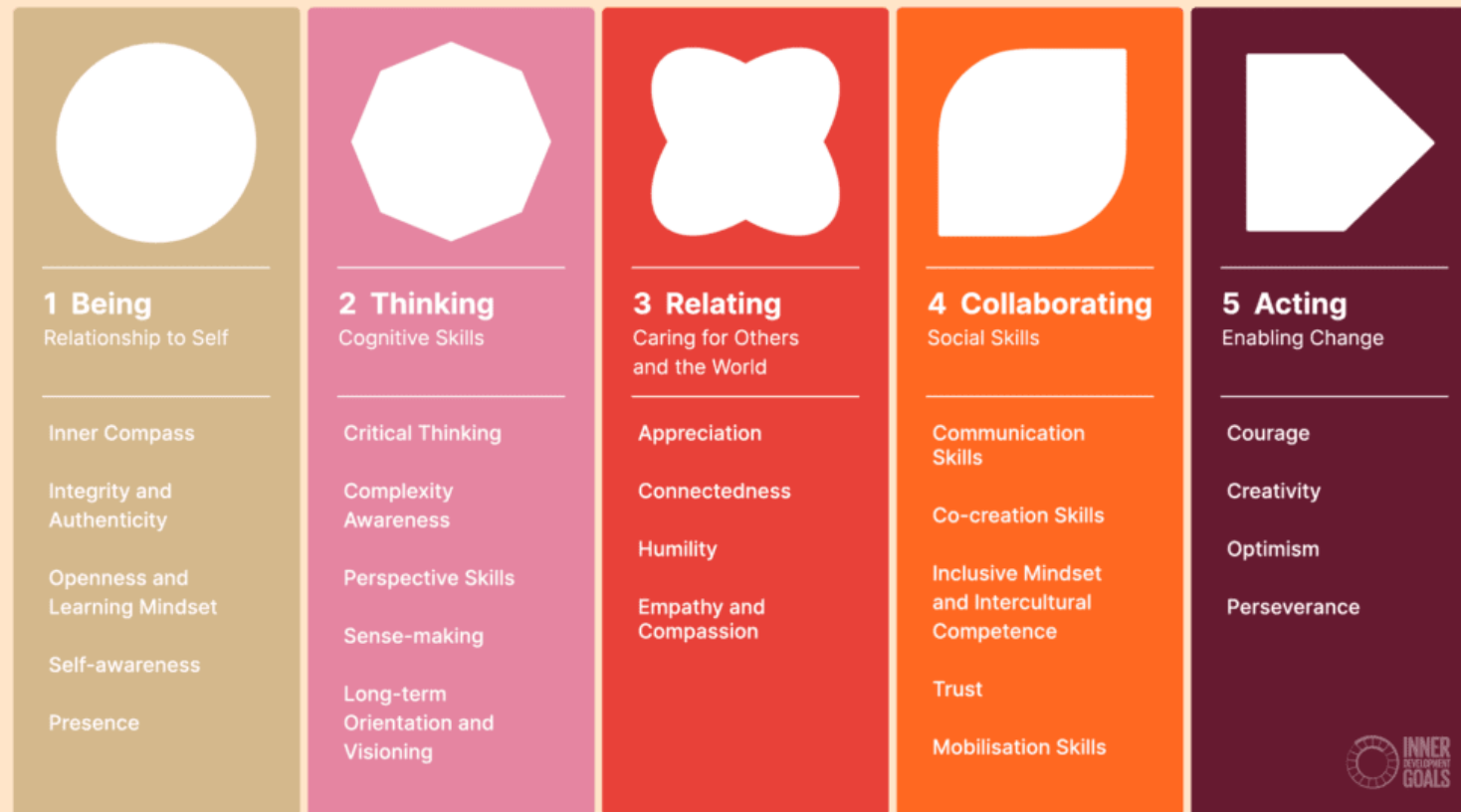
- Find your problem domain with Cynefin
- Deliver value with DevOps
- Build your practices with «The three ways of DevOps» as guiding principles
- Organize for flow with Team Topologies
- Keep Conways Law in mind
- Expand team boundaries with cross functionality
- Measure your delivery performance with the 4 key metrics
- Build platforms for enterprise scalability
- Grow transformational leadership
  - Embrace uncertainty and inspire change

# Recommended reading



# Other ways to deal with adaptable challenges

Change starts from within. Transform and grow with inner development



<https://innerdevelopmentgoals.org/>

Transformative Skills for Sustainable Development

# Questions?

# Thank you for having me!