IT's TINKOFF

Improving software flow

Alexander Polomodov
Tinkoff

IT's TINKOFF

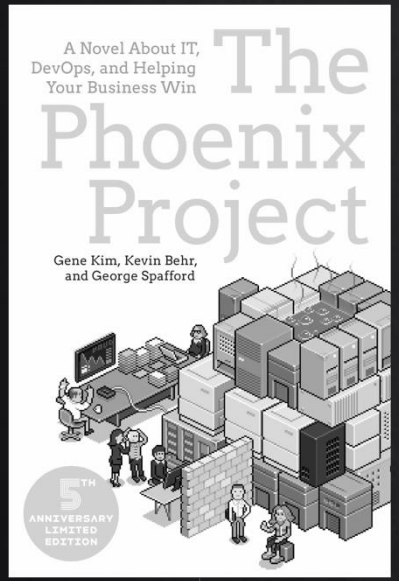
Hello!

My name is Alexander

- Technical Director
- Responsible for architecture and delivery management in the company

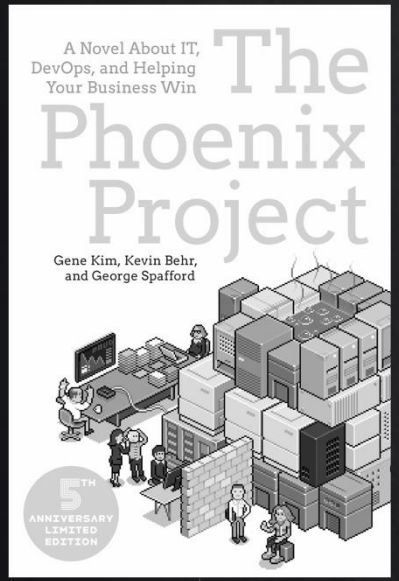
Classical books from IT Revolution

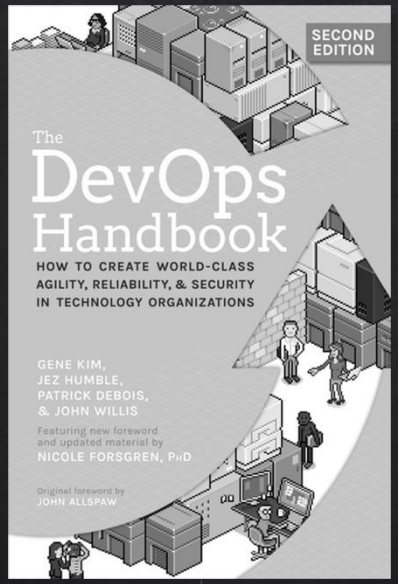
IT's TINKOFF



2013

Classical books from IT Revolution



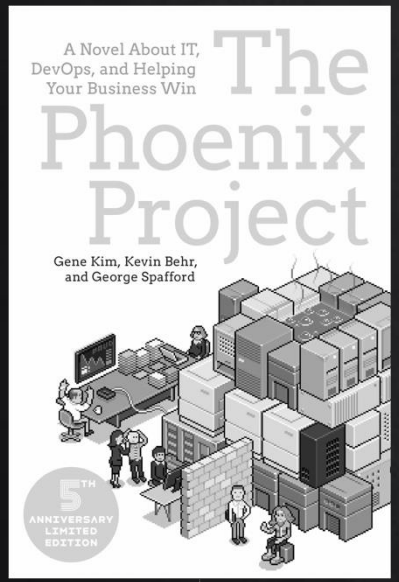


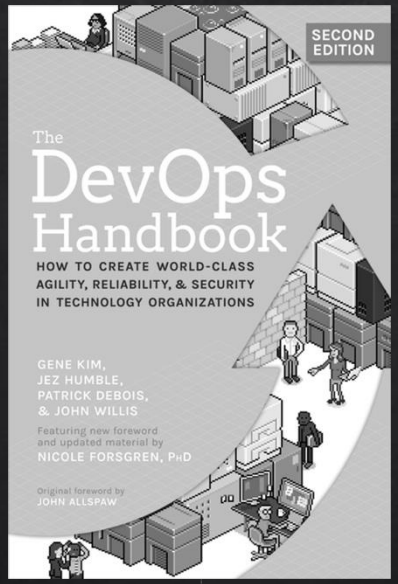
2013

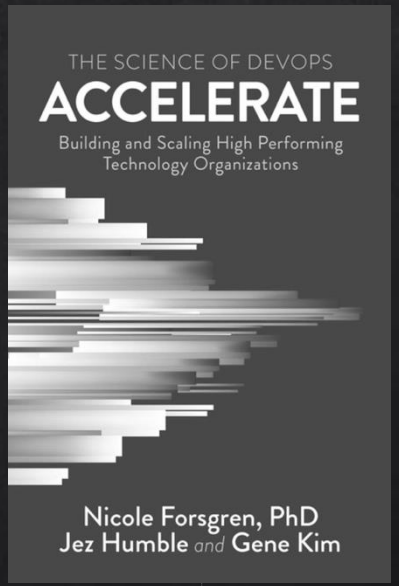
2016

Classical books from IT Revolution

IT's TINKOFF



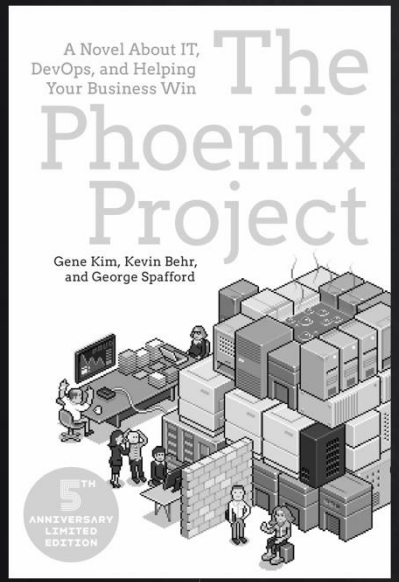


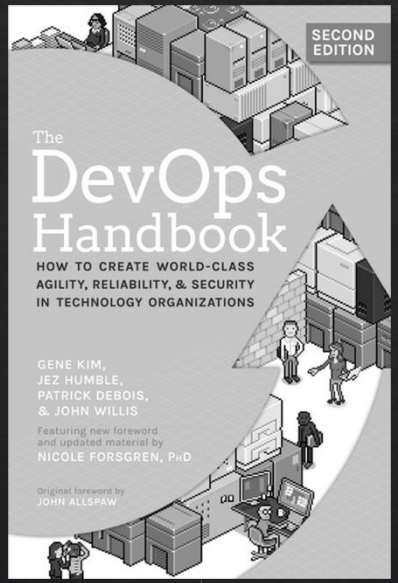


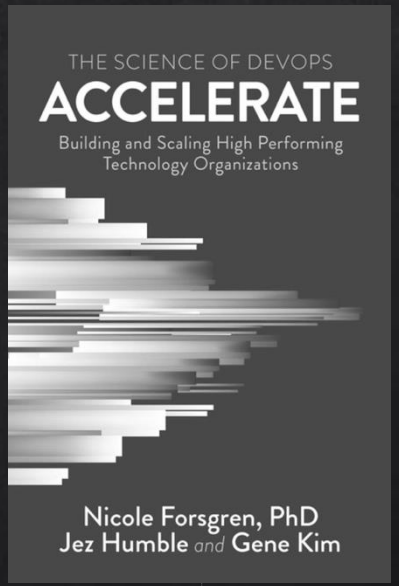
2013 2016 2018

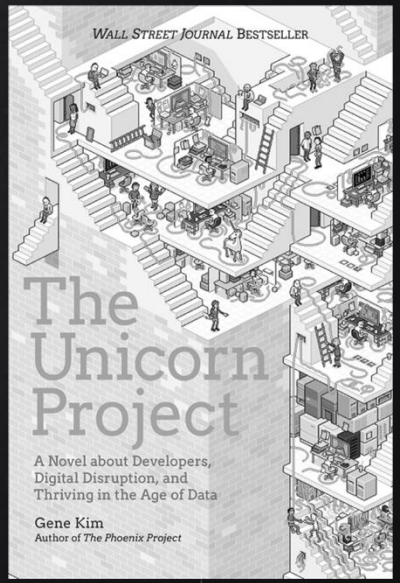
Classical books from IT Revolution

It's TINKOFF





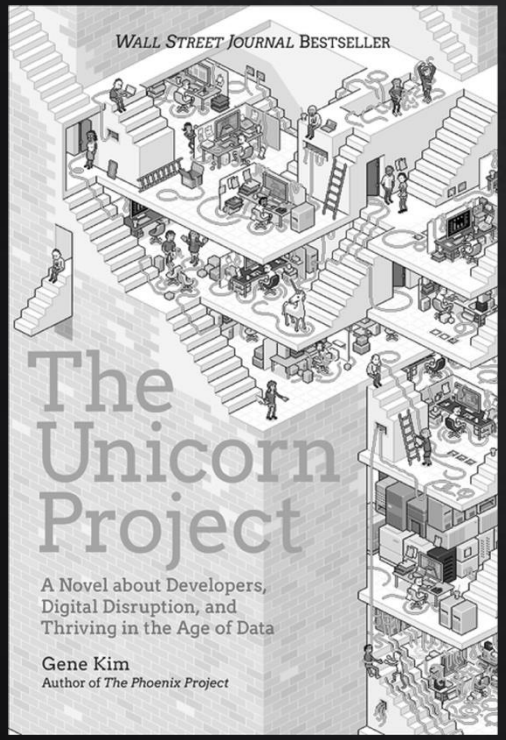




2013 2016 2018 2019

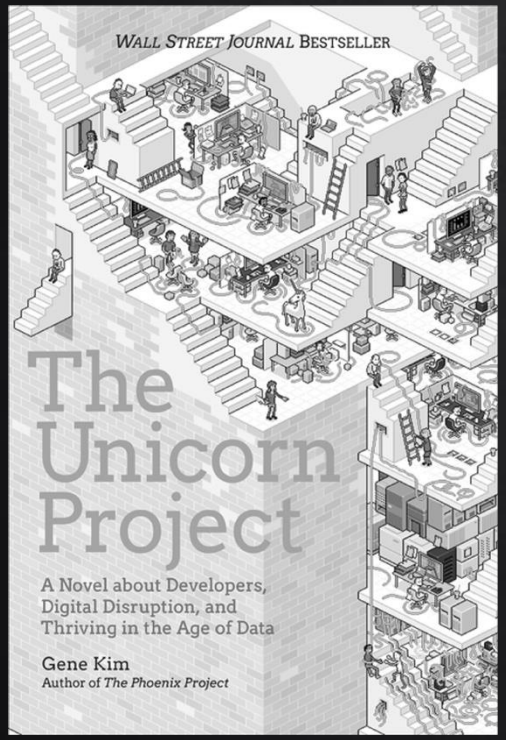
Five ideals from the “Unicorn Project”

IT's TINKOFF



Five ideals from the “Unicorn Project”

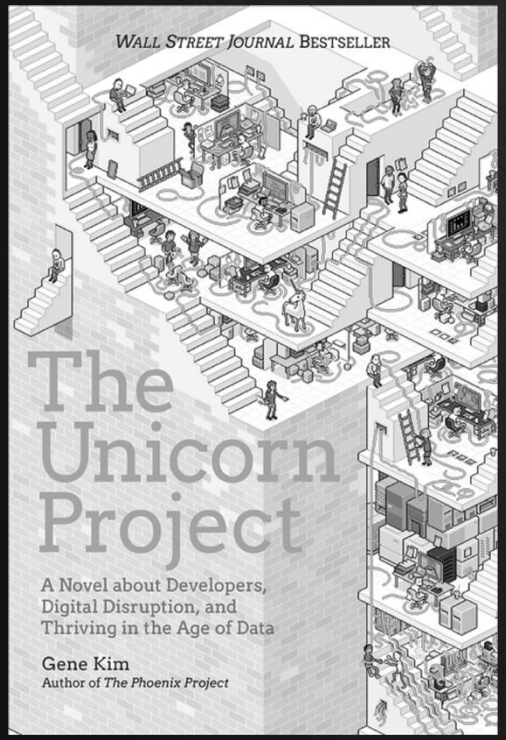
IT's TINKOFF



- Locality and simplicity

Five ideals from the “Unicorn Project”

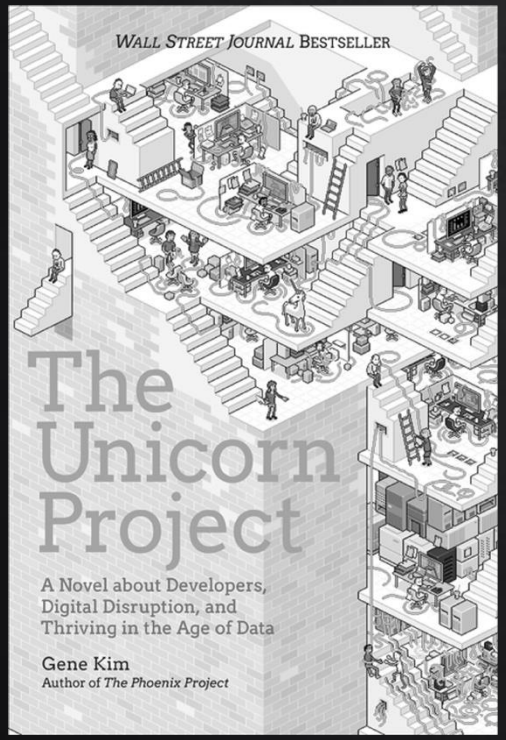
IT's TINKOFF



- Locality and simplicity
- Focus, flow, and joy

Five ideals from the “Unicorn Project”

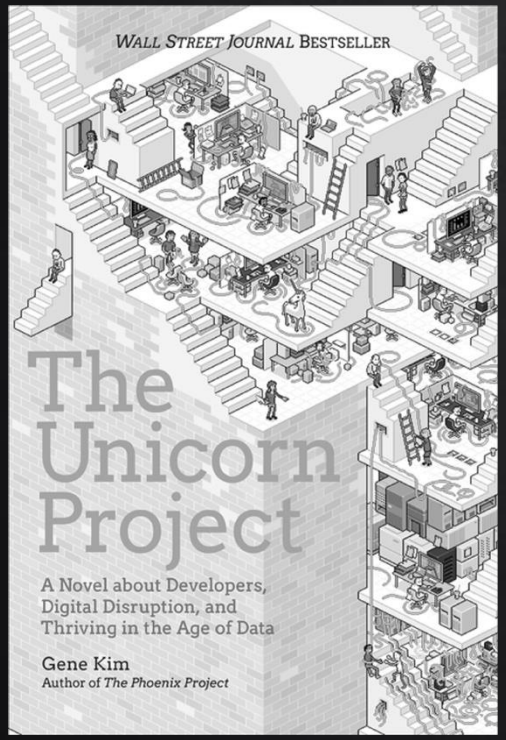
IT's TINKOFF



- Locality and simplicity
- Focus, flow, and joy
- Improvement of daily work

Five ideals from the “Unicorn Project”

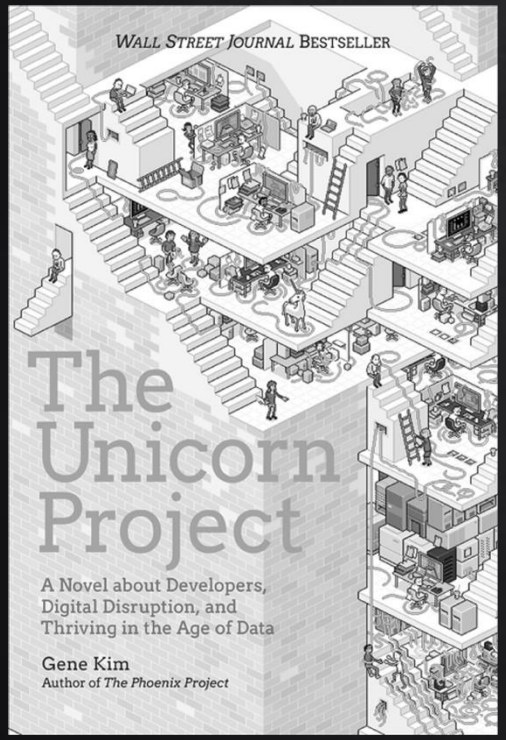
It's TINKOFF



- Locality and simplicity
- Focus, flow, and joy
- Improvement of daily work
- Psychological safety

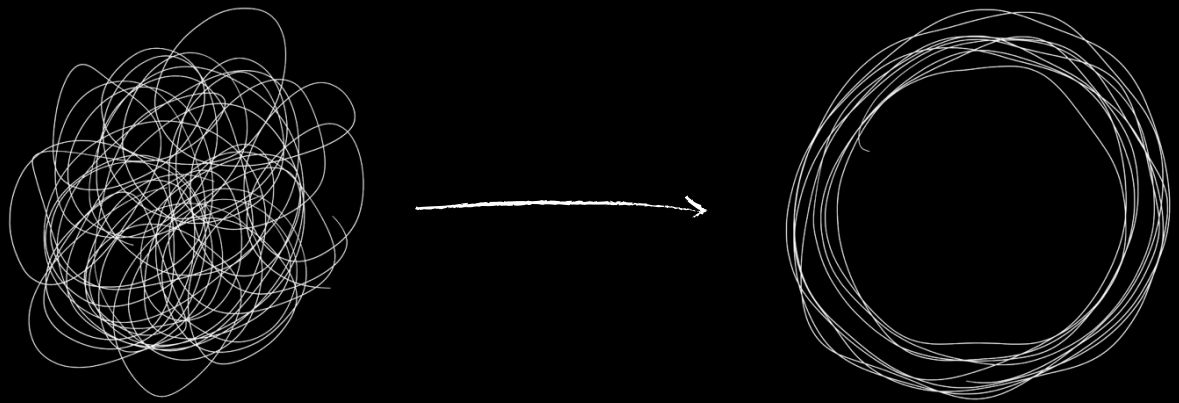
Five ideals from the “Unicorn Project”

IT's TINKOFF



- Locality and simplicity
- Focus, flow, and joy
- Improvement of daily work
- Psychological safety
- **Customer focus**

IT's TINKOFF



Locality and Simplicity (Architecture)

Conway's law

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

Conway's law & "Inverse Conway Maneuver"

IT's TINKOFF

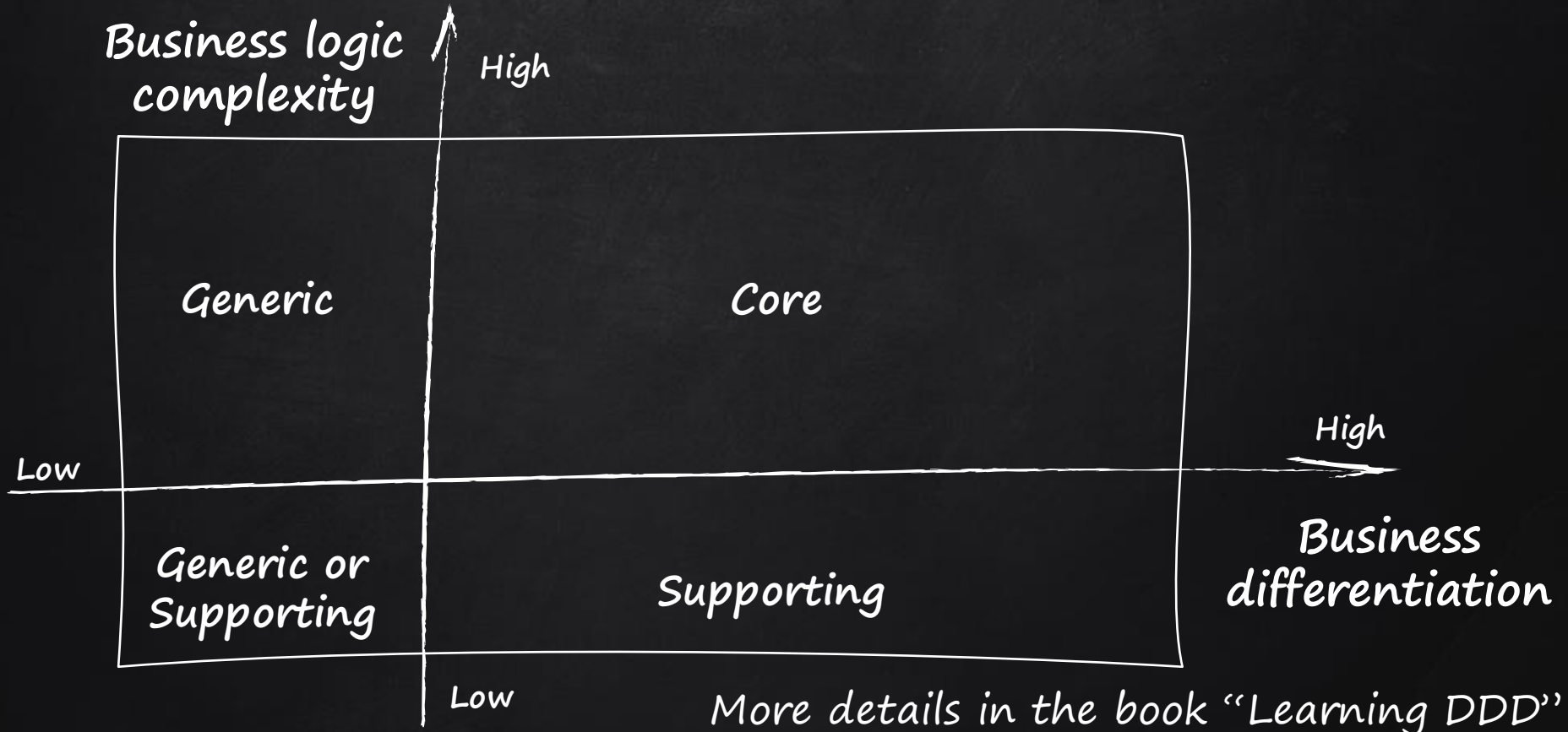
Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

The best way to improve the quality and efficiency of your organization's software is to focus on team design

More details in the book "Team Topologies"

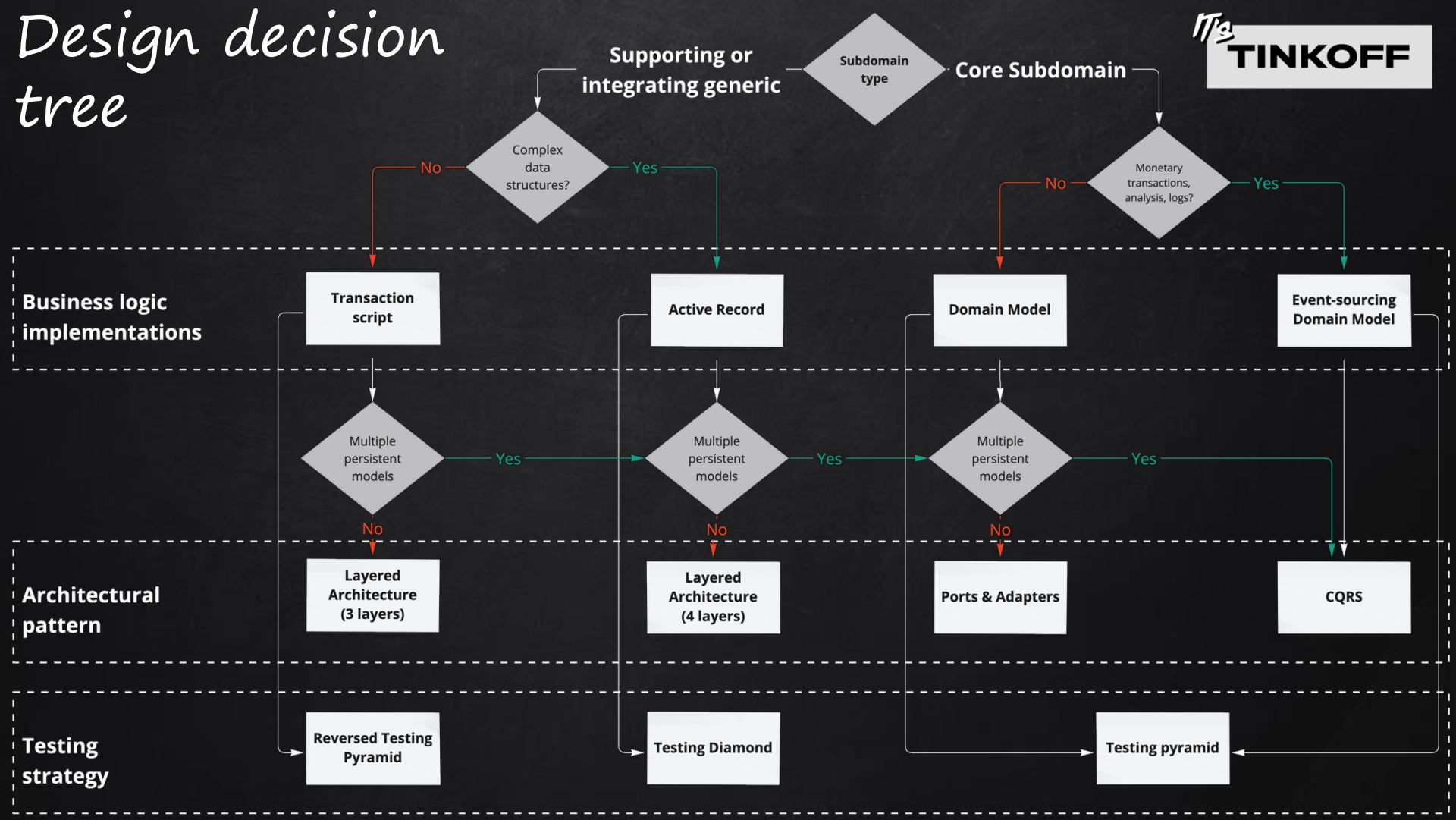
Strategic design from Domain Driven Design

It's TINKOFF



More details in the book "Learning DDD"

Design decision tree



What is microservice?

IT's TINKOFF

- Simpler to understand the function

What is microservice?

IT's TINKOFF

- Simpler to understand the function
- Limit reasons to change

What is microservice?

IT's TINKOFF

- Simpler to understand the function
- Limit reasons to change
- More autonomous for dev, management, scale

What is microservice?

IT's TINKOFF

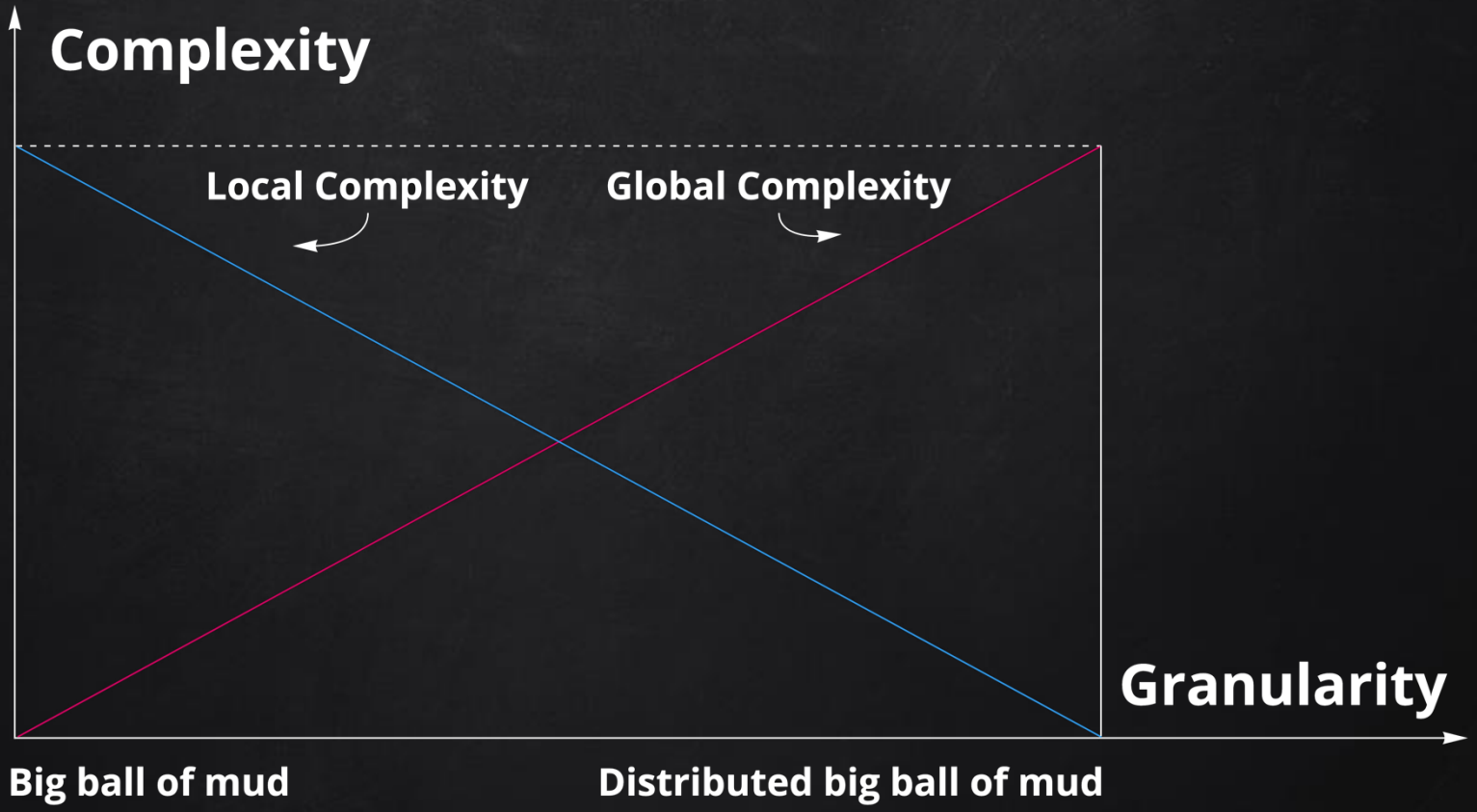
- Simpler to understand the function
- Limit reasons to change
- More autonomous for dev, management, scale
- Simpler to integrate with other components

What is microservice?

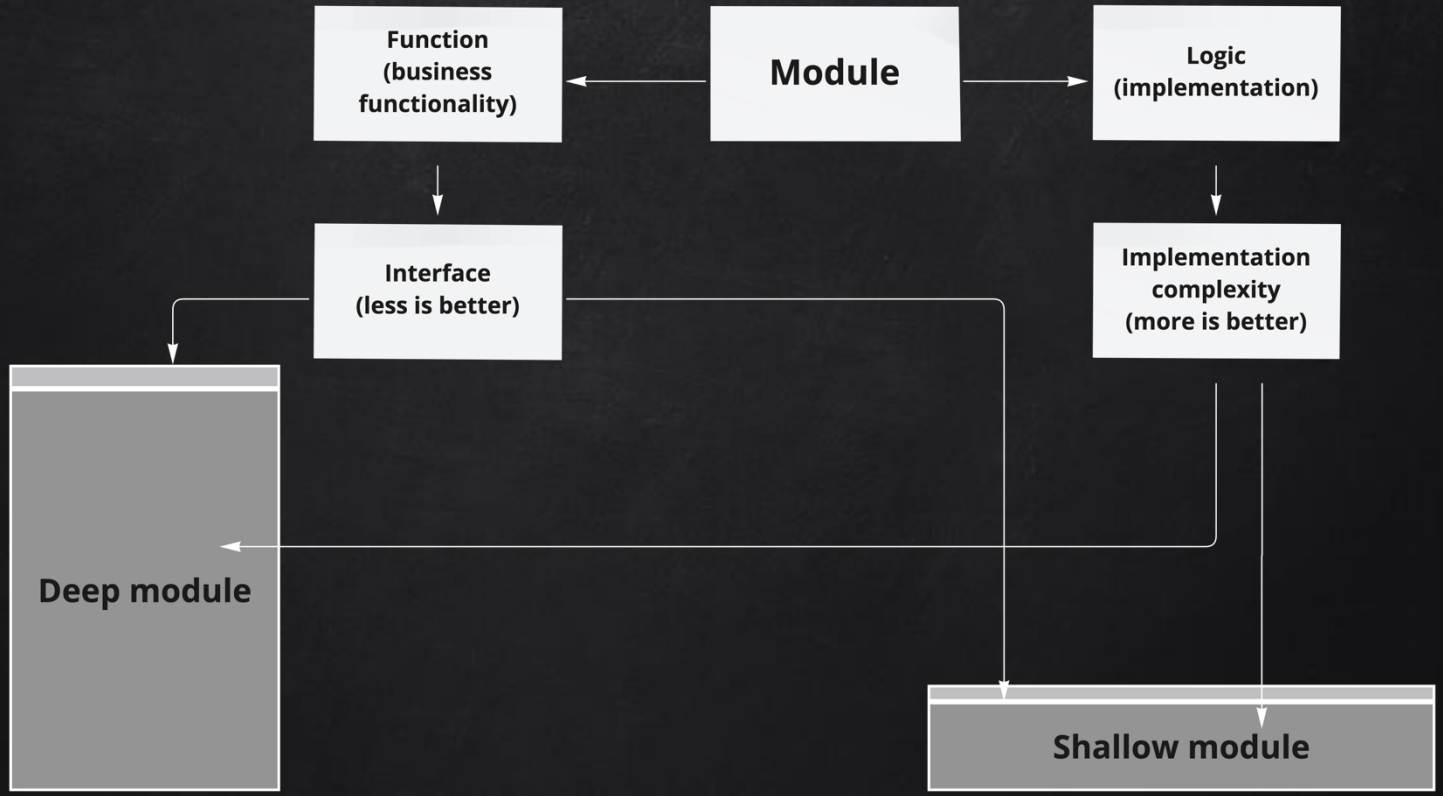
IT's TINKOFF

- Simpler to understand the function
- Limit reasons to change
- More autonomous for dev, management, scale
- Simpler to integrate with other components
- Not exposing whole database, because public database makes interface huge

Relationship between complexity and granularity of services



Deep and shallow microservice



More details in the book "A philosophy of software design"

Domain driven design and boundaries

IT's TINKOFF

- Bounded context is the boundary of a model

Domain driven design and boundaries

IT's TINKOFF

- Bounded context is the boundary of a model
- Subdomains bounds a business capability

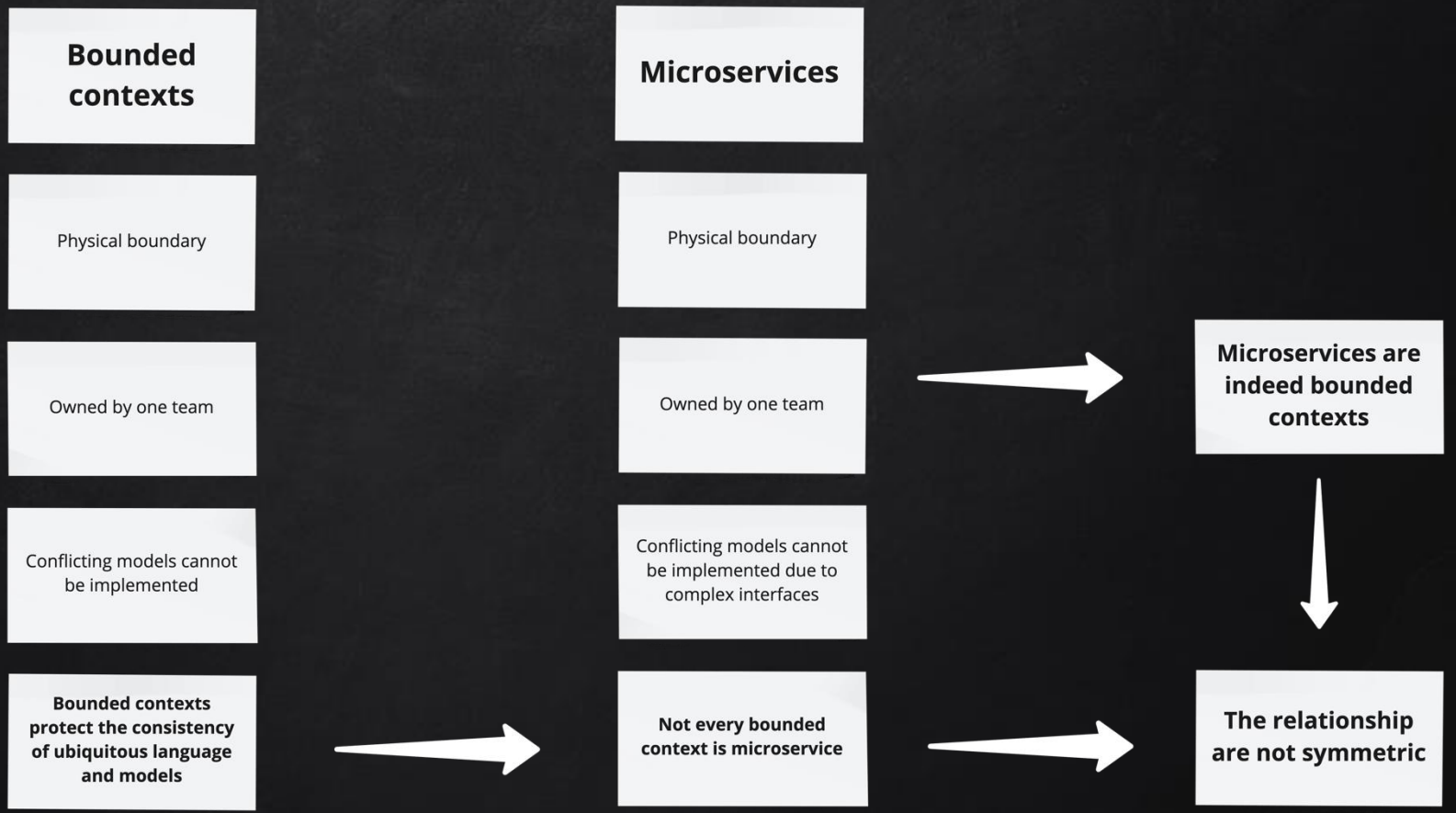
Domain driven design and boundaries

IT's TINKOFF

- Bounded context is the boundary of a model
- Subdomains bounds a business capability
- Aggregates and value objects are transactional boundaries

Bounded contexts and microservices

IT's TINKOFF



Event-Driven Architecture

IT's TINKOFF

Event-driven architecture (EDA) is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events.

Event-Driven Architecture

IT's TINKOFF

Event-driven architecture (EDA) is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events.

- **Components publish events to notify other system elements**

Event-Driven Architecture

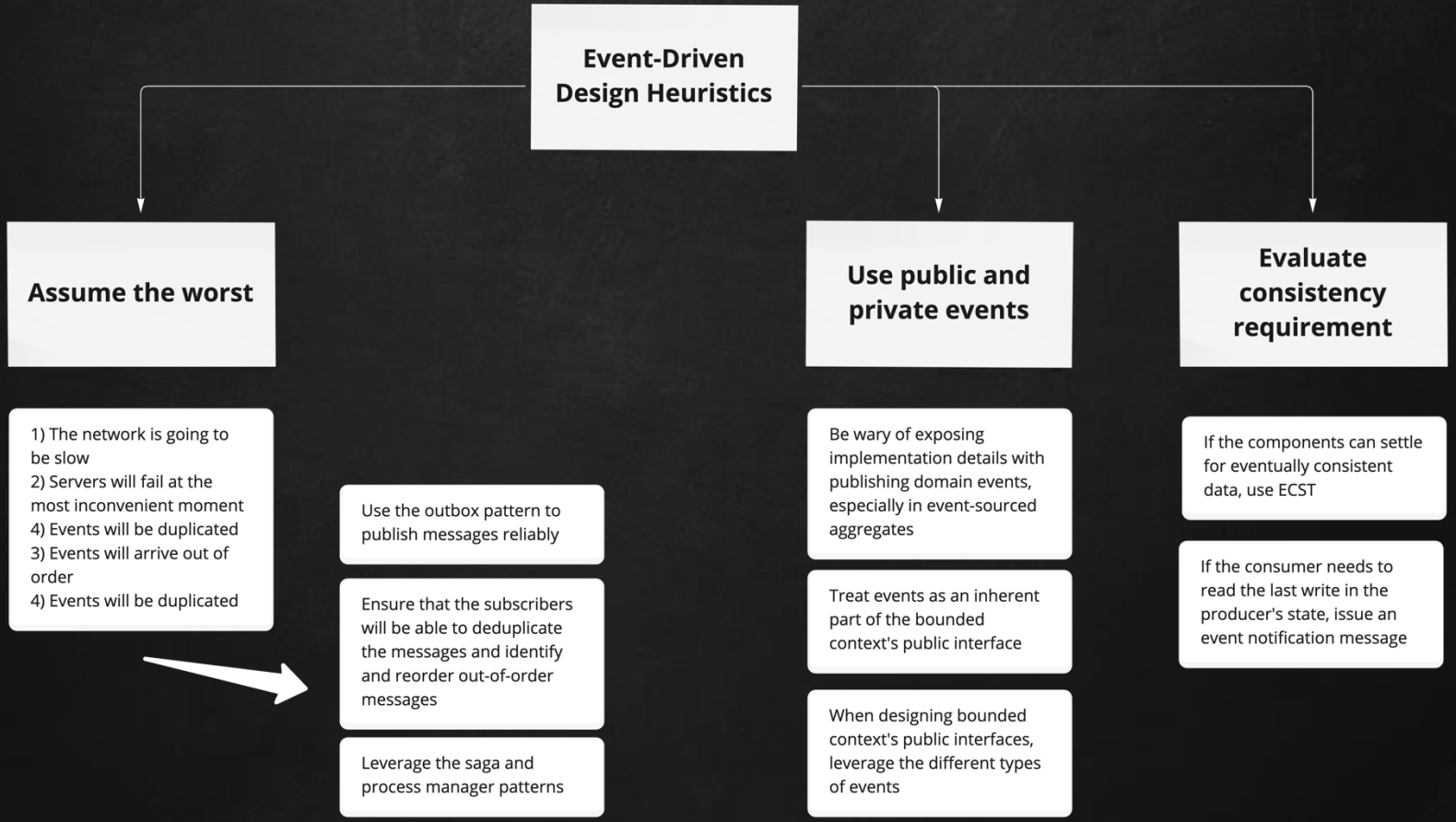
IT's TINKOFF

Event-driven architecture (EDA) is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events.

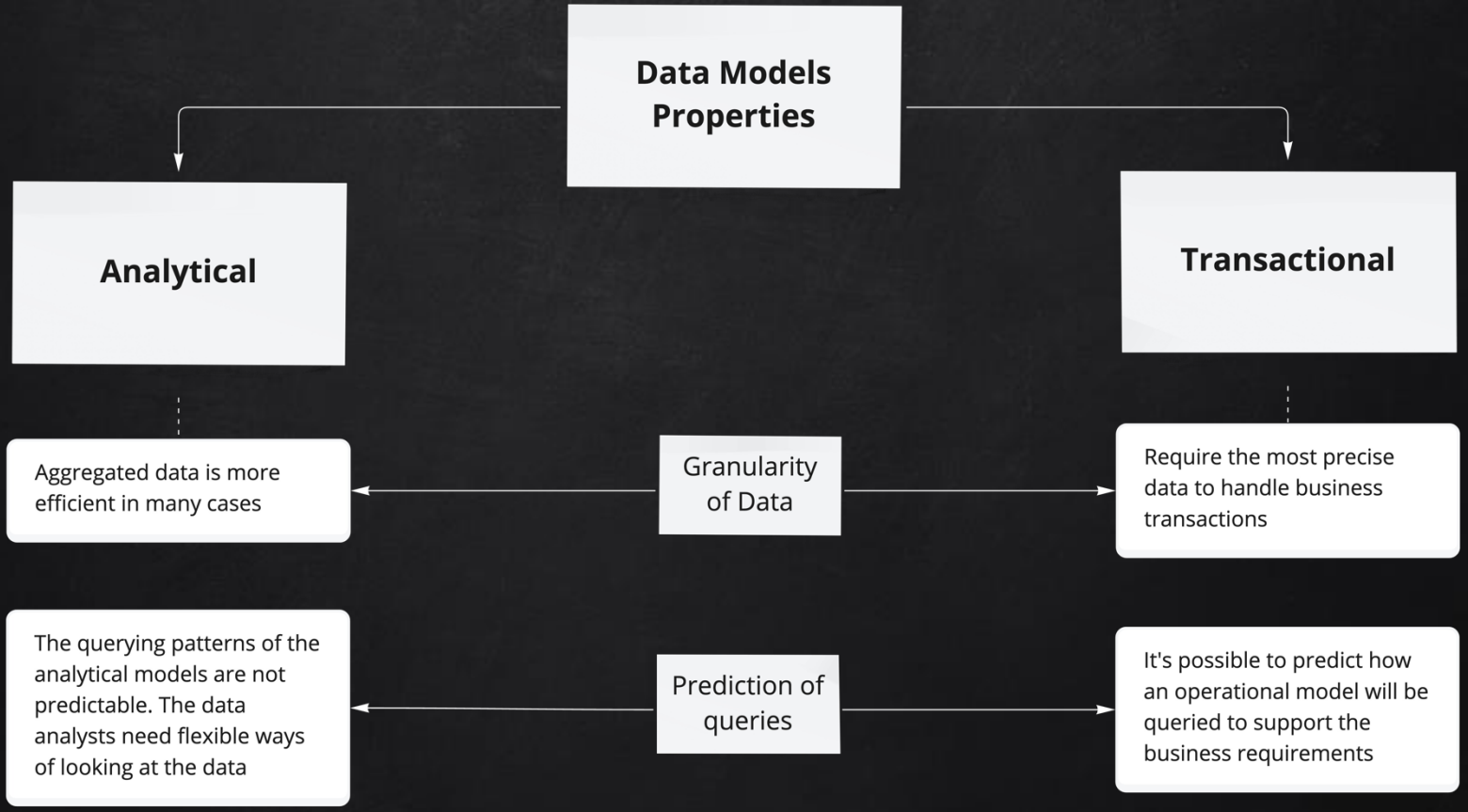
- Components publish events to notify other system elements
- **Components can subscribe to events raised in the system**

Event-Driven Design Heuristics

It's TINKOFF

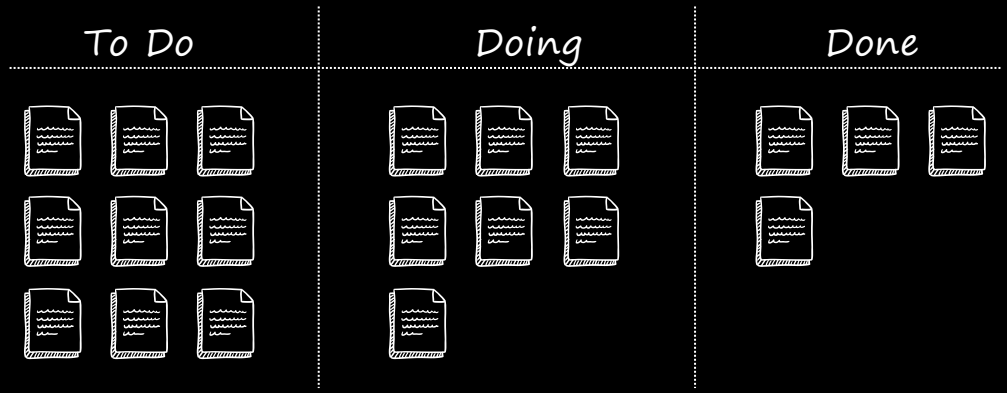


Data models and their properties



DWH and Data Mesh

It's TINKOFF



Focus, Flow, and Joy (Processes)

The Five Thieves of Time

- Too Much Work-in-Progress (WIP)

More details in the book "Making Work Visible"

The Five Thieves of Time

It's TINKOFF

- Too Much Work-in-Progress (WIP)
- Unknown Dependencies

More details in the book "Making Work Visible"

The Five Thieves of Time

It's TINKOFF

- Too Much Work-in-Progress (WIP)
- Unknown Dependencies
- Unplanned Work

More details in the book "Making Work Visible"

The Five Thieves of Time

- Too Much Work-in-Progress (WIP)
- Unknown Dependencies
- Unplanned Work
- Conflicting Priorities

More details in the book "Making Work Visible"

The Five Thieves of Time

- Too Much Work-in-Progress (WIP)
- Unknown Dependencies
- Unplanned Work
- Conflicting Priorities
- Neglected Work

More details in the book "Making Work Visible"

Metrics for operation review

- Throughput

More details in the book “Making Work Visible”

Metrics for operation review

- Throughput
- Flow time

More details in the book “Making Work Visible”

Metrics for operation review

- Throughput
- Flow time
- Issues and blocked work items

More details in the book "Making Work Visible"

Metrics for operation review

- Throughput
- Flow time
- Issues and blocked work items
- The Time Thief O'Gram

More details in the book "Making Work Visible"

Шаблоны T-Meter

Дашборд команды Base

Дашборд команды начального уровня с базовыми метриками для анализа процессов

Демодашборд

Дашборд команды Pro

Дашборд команды продвинутого уровня с расширенным набором метрик для анализа процессов

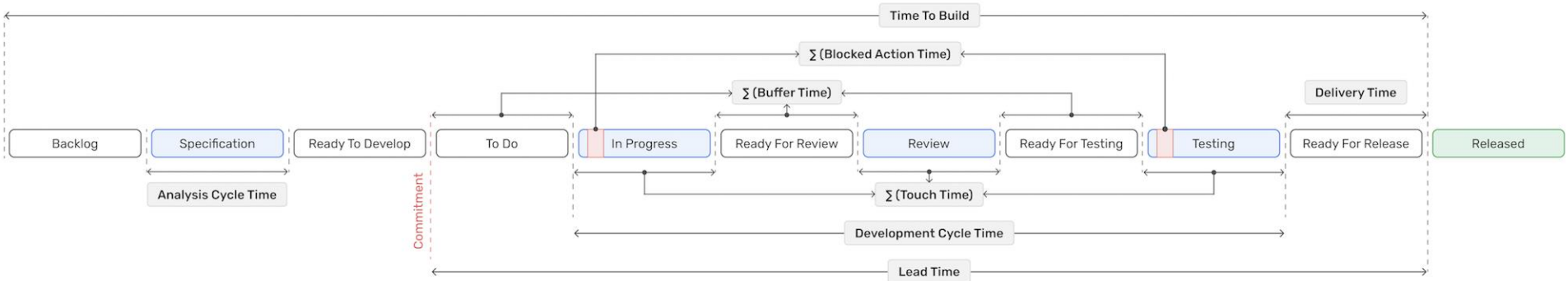
Демодашборд

Дашборд руководителя

Дашборд для мониторинга и анализа метрик команд одного направления (отдела)

Демодашборд

Hint for setting up dashboard



Time In:

Blocked State | Buffer State: Buffer

Action State: Action | Resolved State: Done

Считается **блокировкой**,
если **установлен флаг** или есть **блокирующее соединение**

Flagged = Impediment

Blocked Issue — Is blocked by → Blocking Task

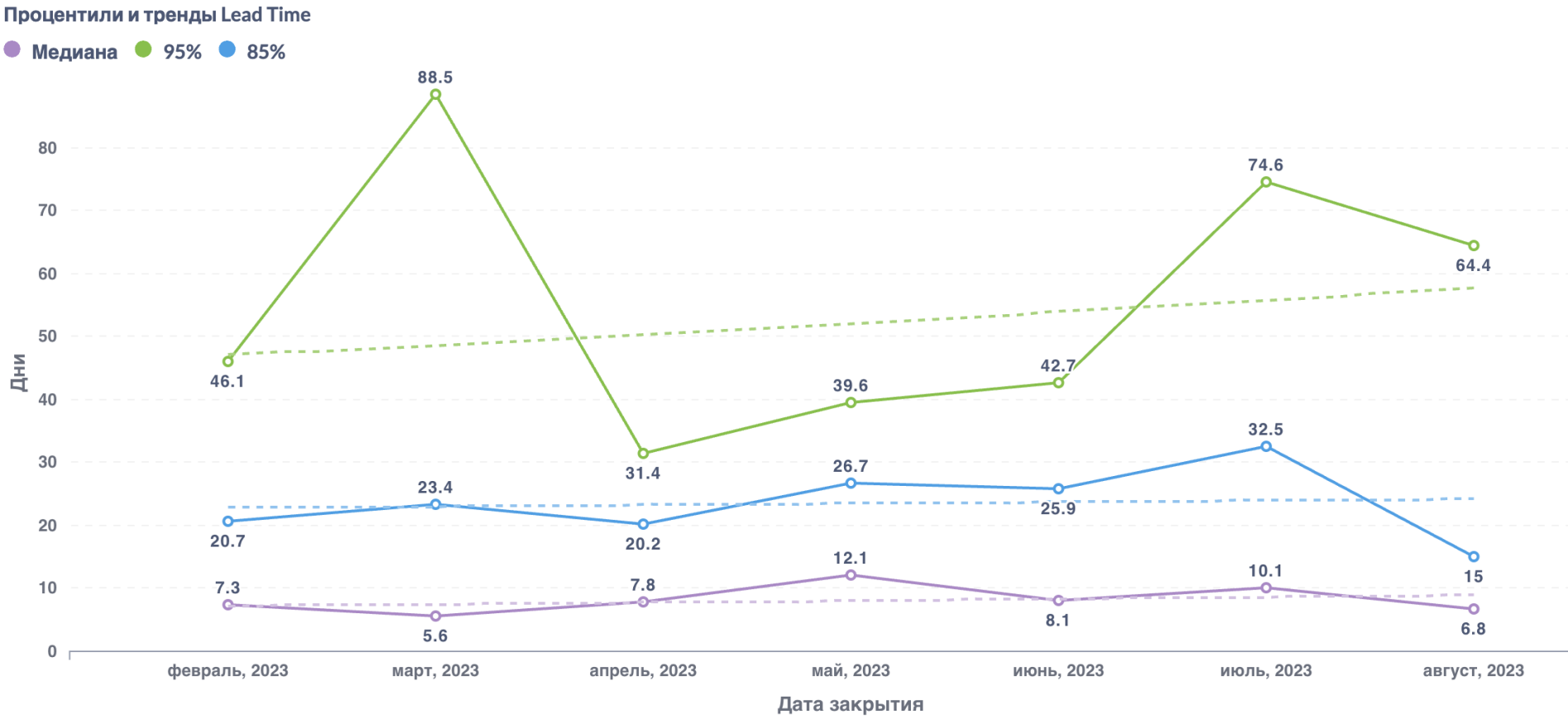
Trash Time - время от момента попадания задачи в один из **статусов**
Lead Time до момента, когда она была отклонена (если была отклонена)

Trashed = ∑ (Trash time)

In Progress, Review, Testing → ∑ (Trash Time)

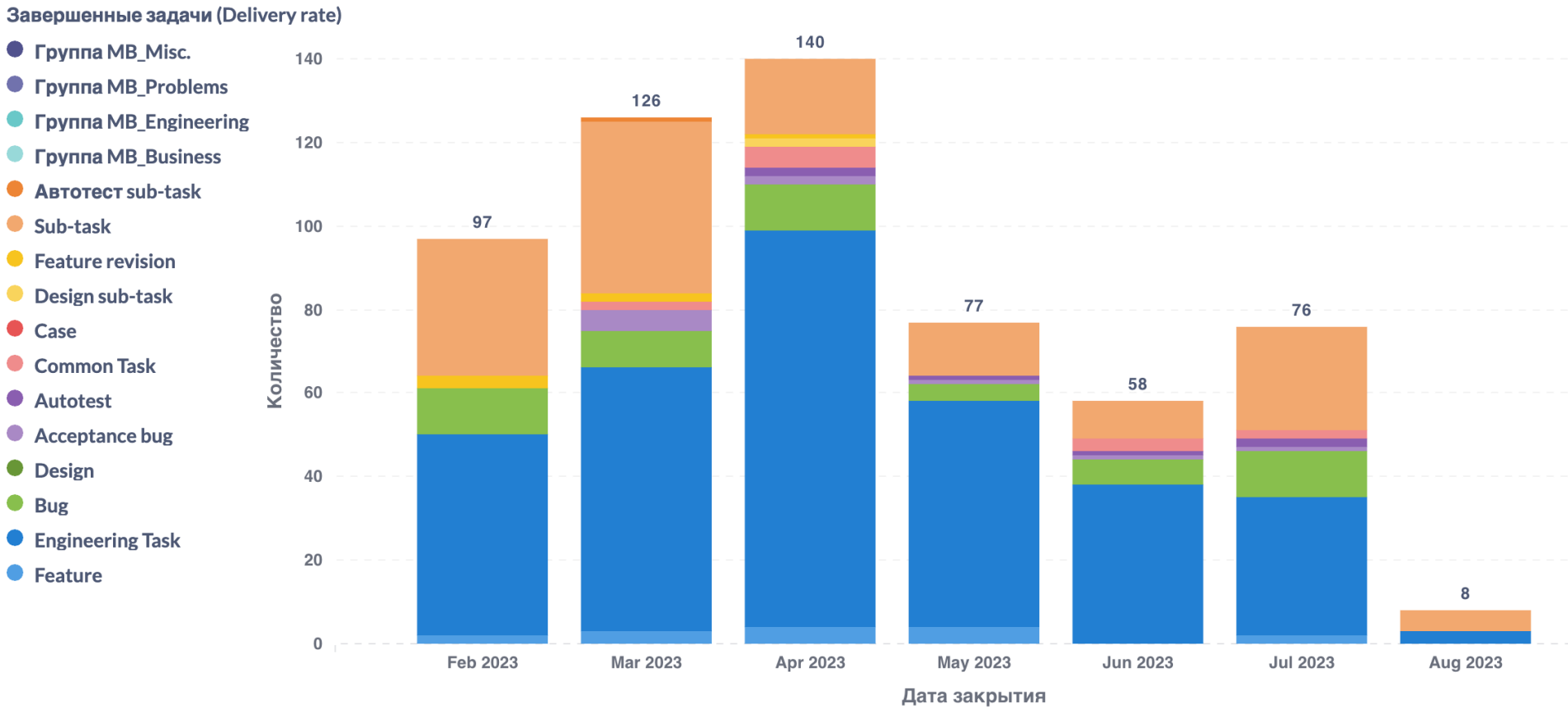
Dashboard example

TINKOFF



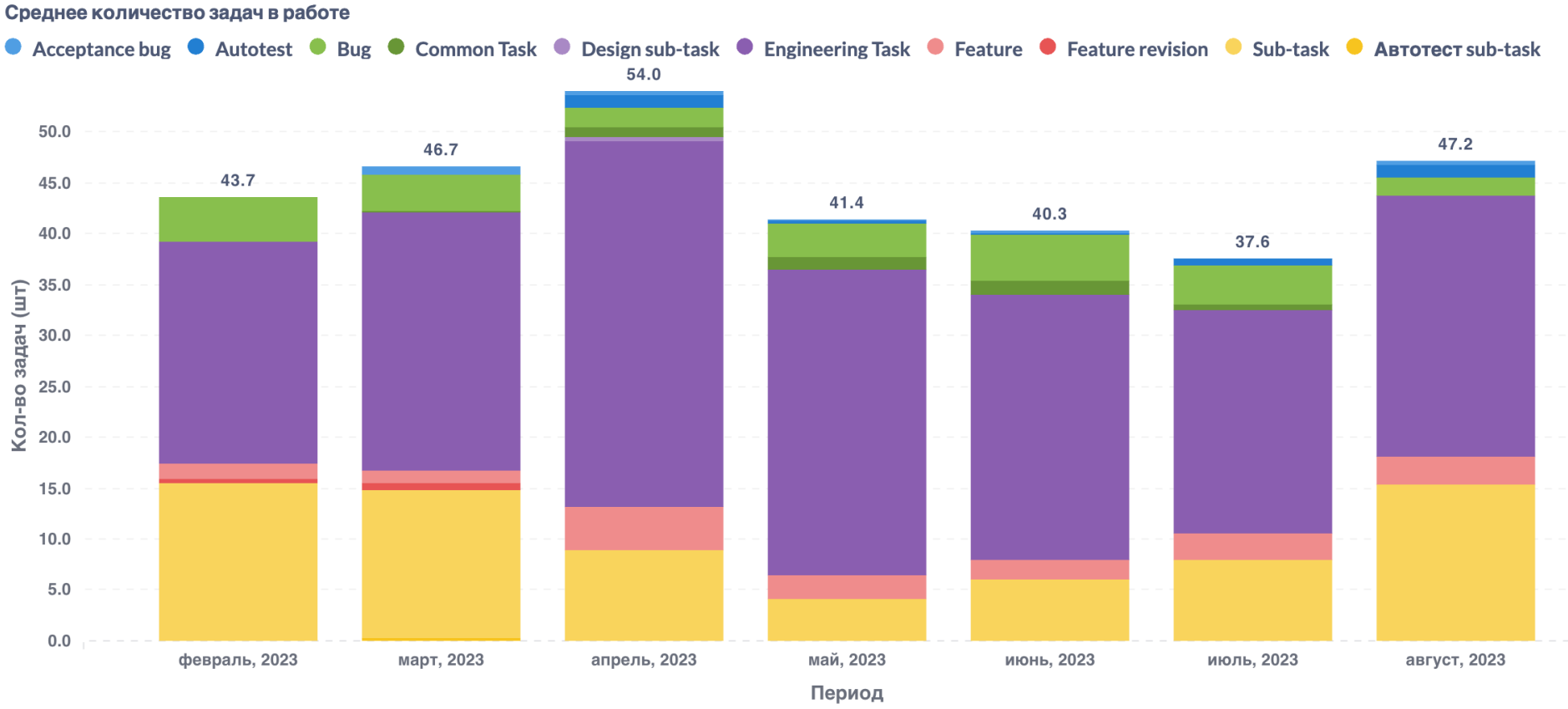
Dashboard example

TINKOFF



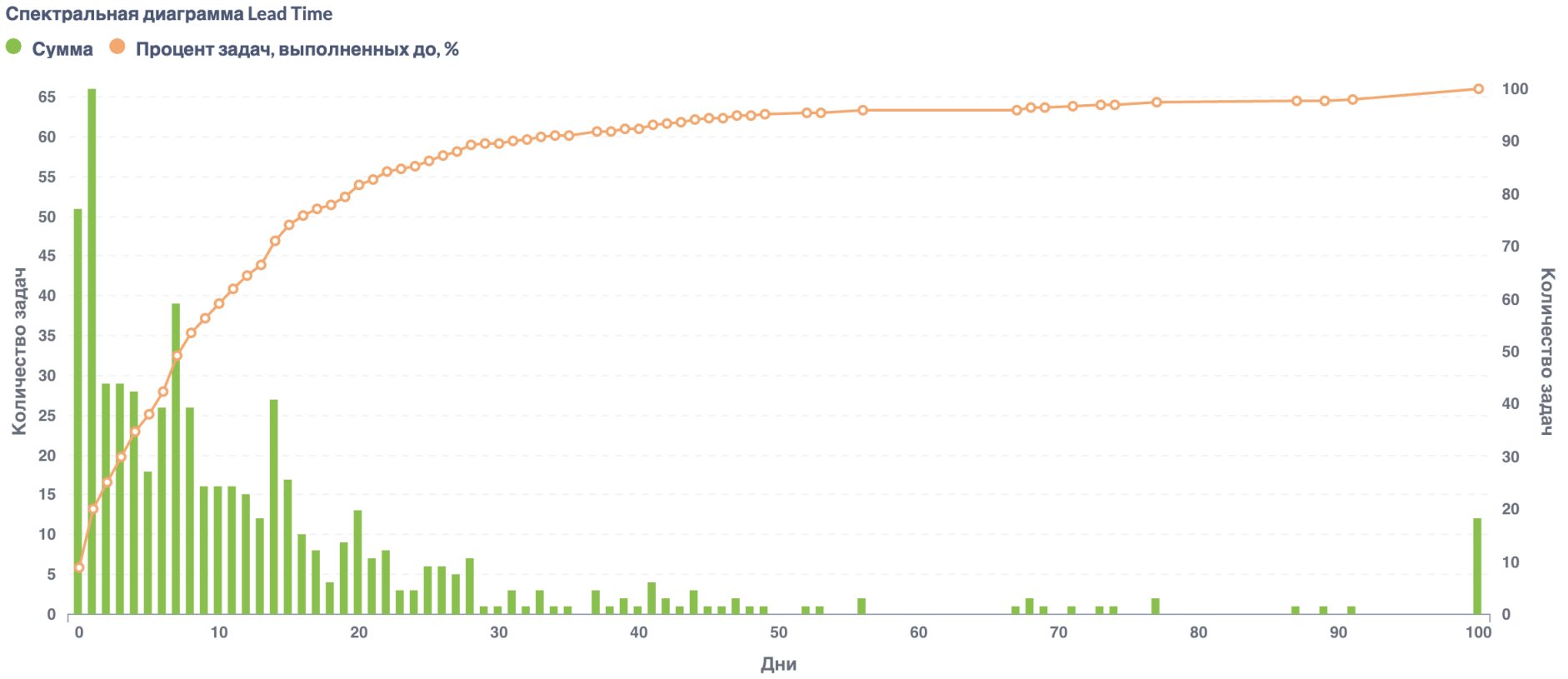
Dashboard example

TINKOFF



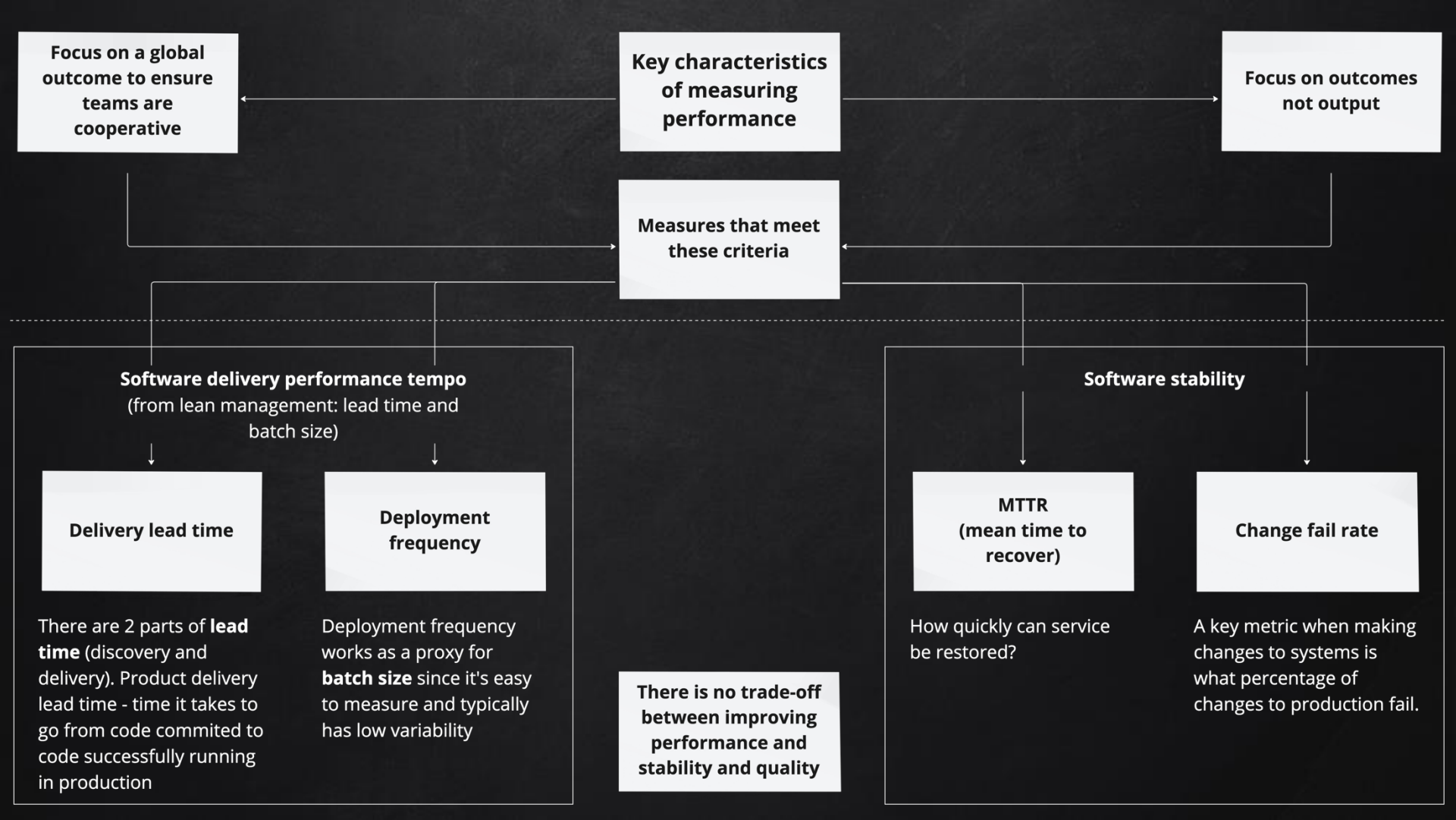
Dashboard example

TINKOFF



Improvement of Daily Work (Engineering practices)

More details in the book "Accelerate"


Focus on a global outcome to ensure teams are cooperative
Key characteristics of measuring performance
Focus on outcomes not output
Measures that meet these criteria
Software delivery performance tempo
(from lean management: lead time and batch size)
Delivery lead time
Deployment frequency
There are 2 parts of lead time (discovery and delivery). Product delivery lead time - time it takes to go from code commited to code successfully running in production
Deployment frequency works as a proxy for batch size since it's easy to measure and typically has low variability
There is no trade-off between improving performance and stability and quality
Software stability
MTTR (mean time to recover)
Change fail rate
How quickly can service be restored?
A key metric when making changes to systems is what percentage of changes to production fail.

Key principles of continuous delivery

Build quality in

In continuous deliraigvery, we invest in building a culture supported by tools and people where we can detect any issues quickly, so that they can be fixed straight away when they are cheap to detect and resolve

Work in small batches

Even though working in small chunks adds some overhead, it reaps enormous rewards by allowing us to avoid work that delivers zero or negative value for organization. A key goal of CD is changing the economics of the software delivery process so the cost of pushing out individual changes is very low

Computers perform repetitive tasks; people solve problems

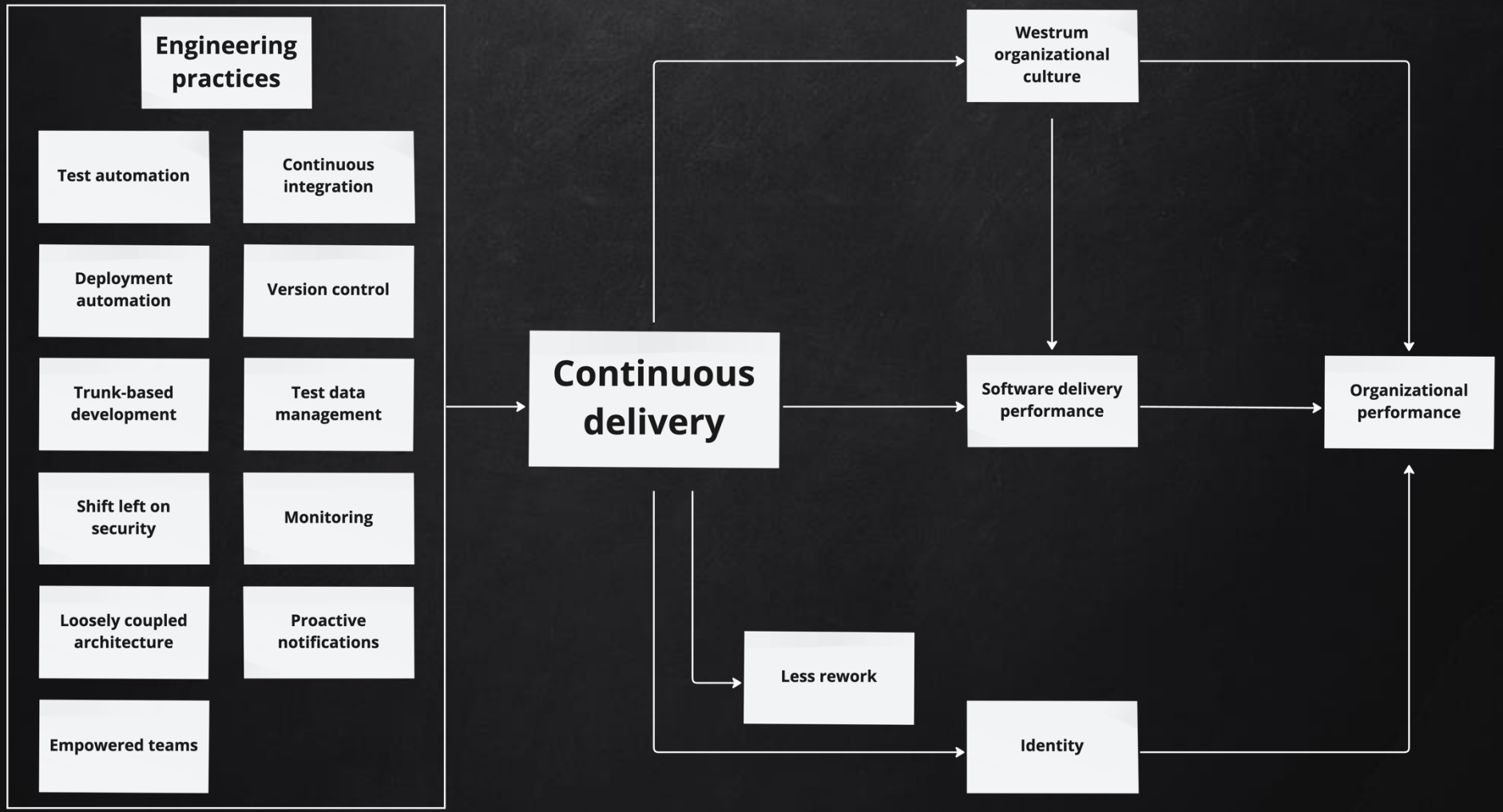
Invest in simplifying and automating repetitive work that takes a long time, such as regression testing and software deployments

Relentlessly pursue continuous impovement

High performing teams are never satisfied, they make improvement part of everybody's daily work

Everyone is responsible

A key objective for management is making the state of system-level outcome (throughput, quality, stability) transparent, working with the rest of organization to set measurable, achievable, time-bound goals for these outcomes, and then helping their teams work toward them


Engineering practices
Test automation
Continuous integration
Deployment automation
Version control
Trunk-based development
Test data management
Shift left on security
Monitoring
Loosely coupled architecture
Proactive notifications
Empowered teams
Continuous delivery
Westrum organizational culture
Software delivery performance
Organizational performance
Less rework
Identity

Dev ... Sec ... Data ... WTF ... Ops

It's TINKOFF

Idea

Requirements

Architecture

Data Scientist

Data Engineer

Data Analyst

Infosec

Quality assurance

Operations

Deploy

Development

Pipeline driven organization

IT's TINKOFF

Idea

Requirements

Architecture

Operations

Deploy

Development

Data Scientist

Data Engineer

Data Analyst

Infosec

Quality assurance

Strategic programming

The first step towards becoming a good software designer is to realize that working code isn't enough. It's not acceptable to introduce unnecessary complexities in order to finish your current task faster...

Your primary goal must be to produce a great design, which also happens to work. This is **strategic programming.**

John Ousterhout, Raft's co-inventor

More details in the book "A philosophy of software design"

Spirit – our internal developer platform

It's TINKOFF

Spirit – our internal developer platform

IT's TINKOFF

- VCS

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
  - Managed Kubernetes

Spirit – our internal developer platform

It's TINKOFF

- VCS
- XaaS
  - Managed Kubernetes
  - DBaaS (Postgres)

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
  - Managed Kubernetes
  - DBaaS (Postgres)
  - Kafka as a Service

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
  - Managed Kubernetes
  - DBaaS (Postgres)
  - Kafka as a Service
  - Cassandra as a Service

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
  - Managed Kubernetes
  - DBaaS (Postgres)
  - Kafka as a Service
  - Cassandra as a Service
  - S3

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
- Pipelines

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
- Pipelines
  - Build infrastructure

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
- Pipelines
  - Build infrastructure
  - Image registry

Spirit – our internal developer platform

- VCS
- XaaS
- Pipelines
  - Build infrastructure
  - Image registry
  - Test management system (Allure) + Quality gate

Spirit – our internal developer platform

- VCS
- XaaS
- Pipelines
  - Build infrastructure
  - Image registry
  - Test management system (Allure) + Quality gate
  - Performance testing (Cosmos)

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
- Pipelines
  - Build infrastructure
  - Image registry
  - Test management system (Allure) + Quality gate
  - Performance testing (Cosmos)
  - Fitness functions (Danger)

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
- Pipelines
- Operation

Spirit – our internal developer platform

IT's TINKOFF

- VCS
- XaaS
- Pipelines
- Operation
  - Observability platform (Sage) and traces

Spirit – our internal developer platform

- VCS
- XaaS
- Pipelines
- Operation
  - Observability platform (Sage) and traces
  - SLAser (SLAs for all system)

Spirit – our internal developer platform

- VCS
- XaaS
- Pipelines
- Operation
  - Observability platform (Sage) and traces
  - SLAser (SLAs for all system)
  - OMG (for incidents)

Psychological Safety (Organizational culture)

Project Aristotle by Google

IT's TINKOFF

What makes a team effective at Google?

Project Aristotle by Google

IT's TINKOFF

What makes a team effective at Google?

- Psychological safety
  - Team members feel safe to take risks and be vulnerable in front of each other

Project Aristotle by Google

What makes a team effective at Google?

- Psychological safety
- Dependability
  - Team members get things done on time and meet Google's high bar for excellence

Project Aristotle by Google

What makes a team effective at Google?

- Psychological safety
- Dependability
- Structure and clarity
  - Team members have clear roles, plans and goals

Project Aristotle by Google

IT's TINKOFF

What makes a team effective at Google?

- Psychological safety
- Dependability
- Structure and clarity
- Meaning
  - Work is personally important to team members

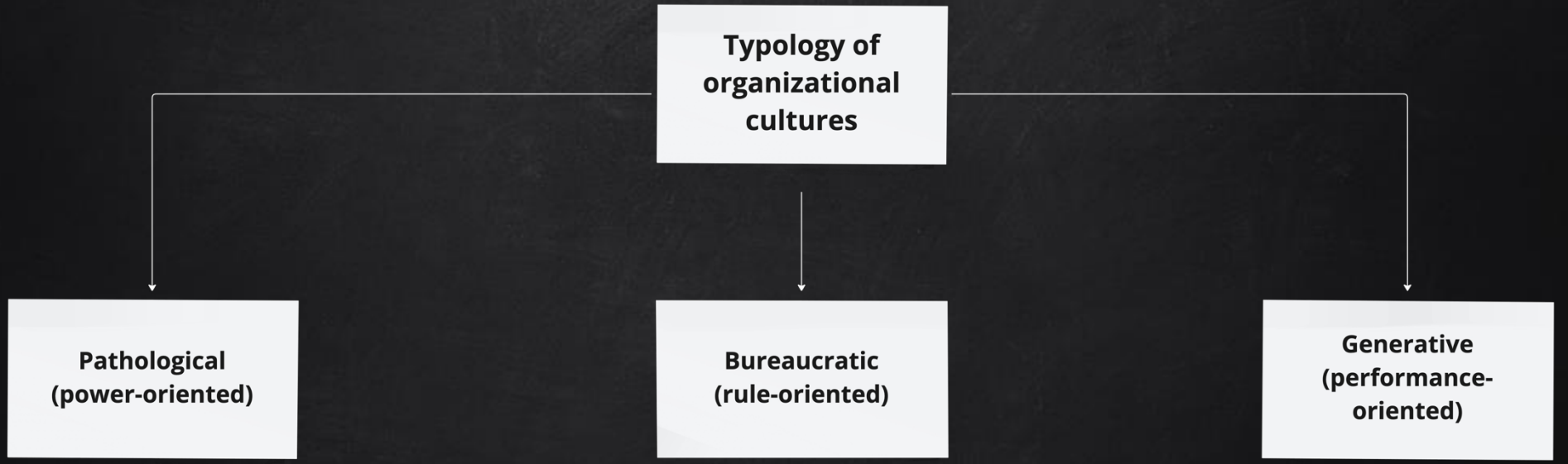
Project Aristotle by Google

What makes a team effective at Google?

- Psychological safety
- Dependability
- Structure and clarity
- Meaning
- Impact
  - Team members think their work matters and creates change

Westrum organizational cultures

IT's TINKOFF

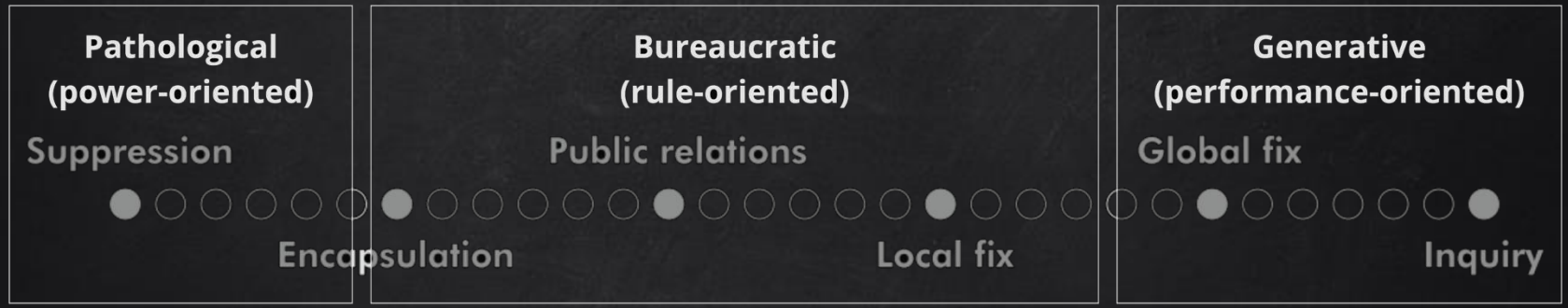


Organizations are characterized by large amounts of fear and threat. People often hoard information or withhold it for political reasons, or distort it to make themselves look better.

Organizations protect departments. Those in the department want to maintain their "turf," insist on their own rules, and generally do things by the book—*their* book.

Organizations focus on the mission. How do we accomplish our goal? Everything is subordinated to good performance, to doing what we are supposed to do.

Responses to anomalous information



Suppression—Harming or stopping the person bringing the anomaly to light; "shooting the messenger"

Encapsulation—Isolating the messenger, so that the message is not heard

Public relations—Putting the message "in context" to minimise its impact

Local fix—Responding to the presenting case, but ignoring the possibility of others elsewhere

Global fix—An attempt to respond to the problem wherever it exists. Common in aviation, when a single problem will direct attention to similar ones elsewhere

Inquiry—Attempting to get at the "root causes" of the problem

Postmortems

IT's TINKOFF

The primary goals of writing a postmortem are to ensure that the incident is documented, that all contributing root cause(s) are well understood, and, especially, that effective preventive actions are put in place to reduce the likelihood and/or impact of recurrence.

From Google SRE Book

Postmortems

IT's TINKOFF

The primary goals of writing a postmortem are to ensure that the incident is documented, that all contributing root cause(s) are well understood, and, especially, that effective preventive actions are put in place to reduce the likelihood and/or impact of recurrence.

From Google SRE Book

I have two public stories about postmortems

Postmortems

The primary goals of writing a postmortem are to ensure that the incident is documented, that all contributing root cause(s) are well understood, and, especially, that effective preventive actions are put in place to reduce the likelihood and/or impact of recurrence.

From Google SRE Book

I have two public stories about postmortems

- Culture of postmortems or how we learn from ~~our~~ fuckups

Postmortems

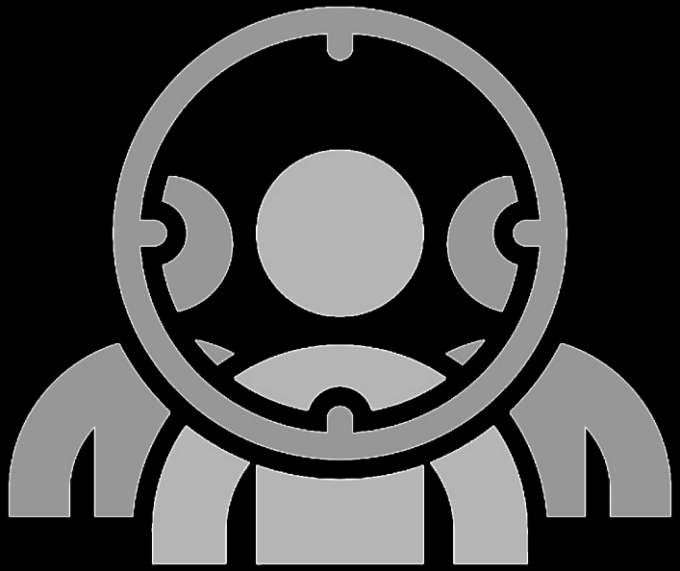
IT's TINKOFF

The primary goals of writing a postmortem are to ensure that the incident is documented, that all contributing root cause(s) are well understood, and, especially, that effective preventive actions are put in place to reduce the likelihood and/or impact of recurrence.

From Google SRE Book

I have two public stories about postmortems

- Culture of postmortems or how we learn from ~~our~~ fuckups
- From monolith to microservices and back again at Fuckup Nights



Customer Focus
(Outcome not output)

Business outcomes

IT's TINKOFF

- Clear business metrics

Business outcomes

IT's TINKOFF

- Clear business metrics
  - Revenue

Business outcomes

IT's TINKOFF

- Clear business metrics
  - Revenue
  - Profit

Business outcomes

IT's TINKOFF

- Clear business metrics
  - Revenue
  - Profit
  - Market share

Business outcomes

IT's TINKOFF

- Clear business metrics
- Clear customer metrics

Business outcomes

- Clear business metrics
- Clear customer metrics
  - Performance of key customer flow

Business outcomes

IT's TINKOFF

- Clear business metrics
- Clear customer metrics
  - Performance of key customer flow
  - Failed customer interactions

Business outcomes

IT's TINKOFF

- Clear business metrics
- Clear customer metrics
  - Performance of key customer flow
  - Failed customer interactions
  - Availability to business

The biggest cause of failure

The biggest cause of failure in software-intensive systems is not technical failure; it's building the wrong thing.

Mary Poppendieck, co-author
"Lean Software Development"

Поделись своим впечатлением о докладе