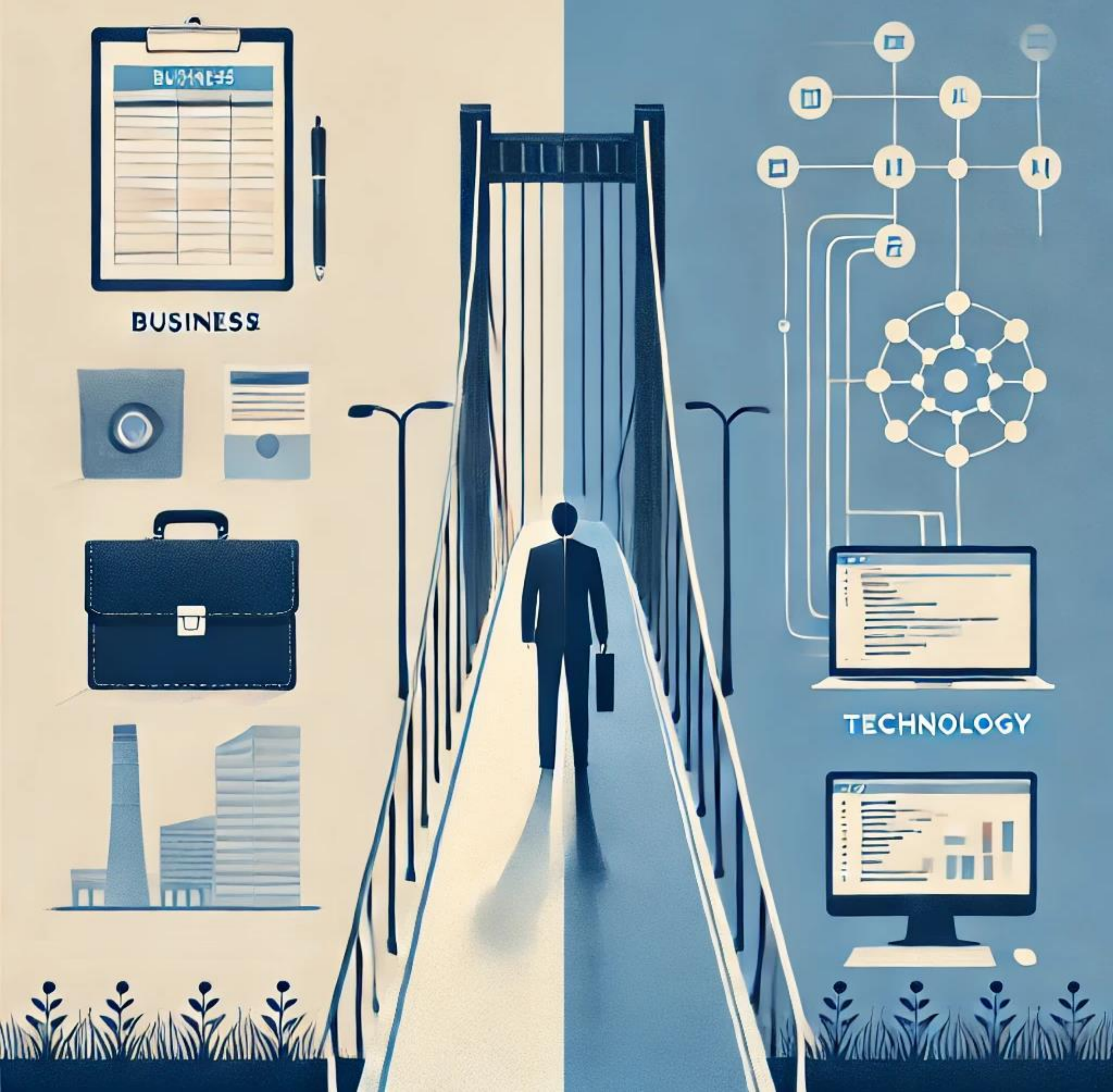
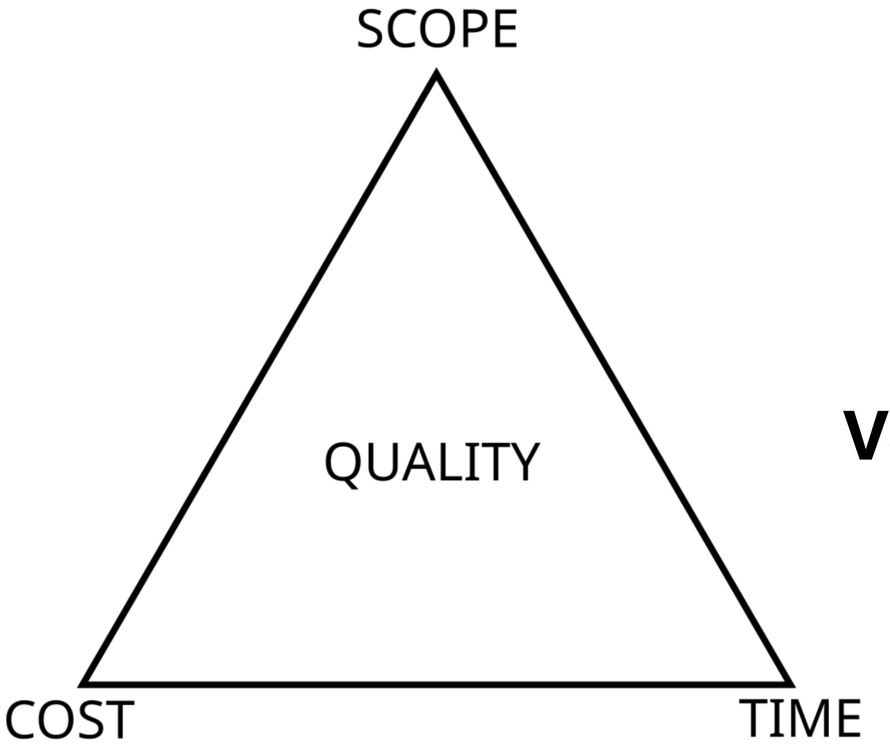


# Bridging the Communication Gap Between Tech and Business

Ivan Kusalic @ TechHive

# Which one do you prefer?



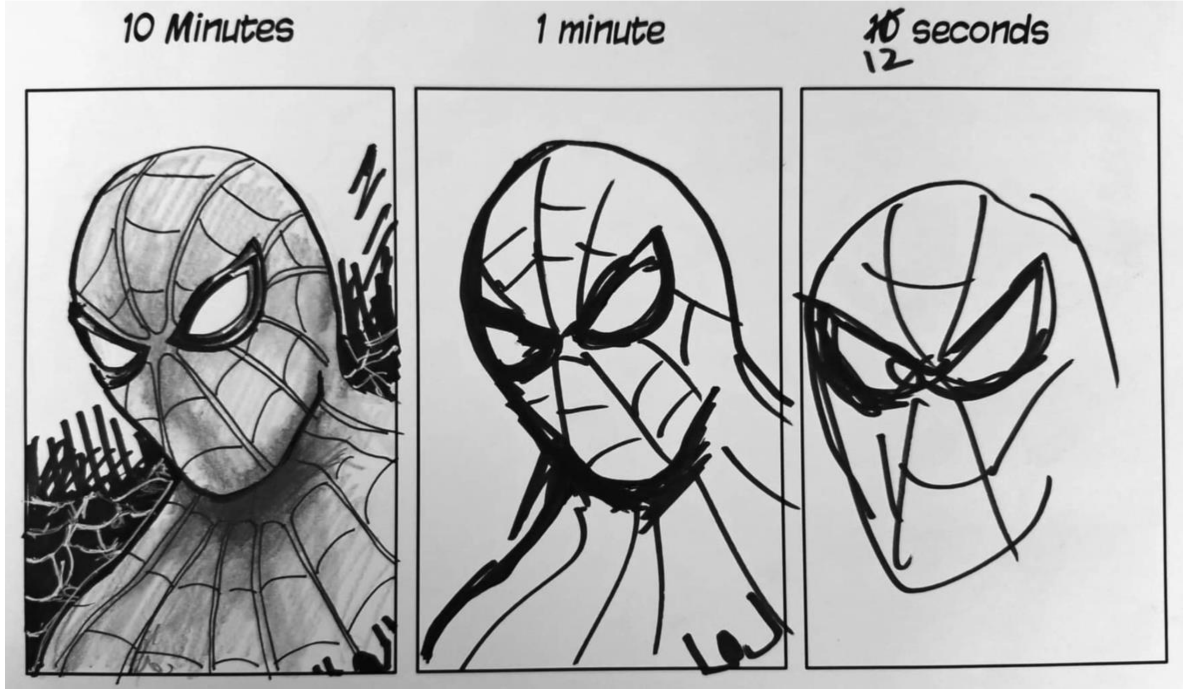
SCOPE

QUALITY

COST

TIME

VS



# Communication is hard

We express ourselves poorly, others may not be engaged, and distractions or conflicting priorities get in the way

Poor communication leads to disconnect, mistrust, and friction

Consequences: failed projects, wasted investments, and job losses

# “Communication is what the listener does”

Effective communication gets the message across to the listener

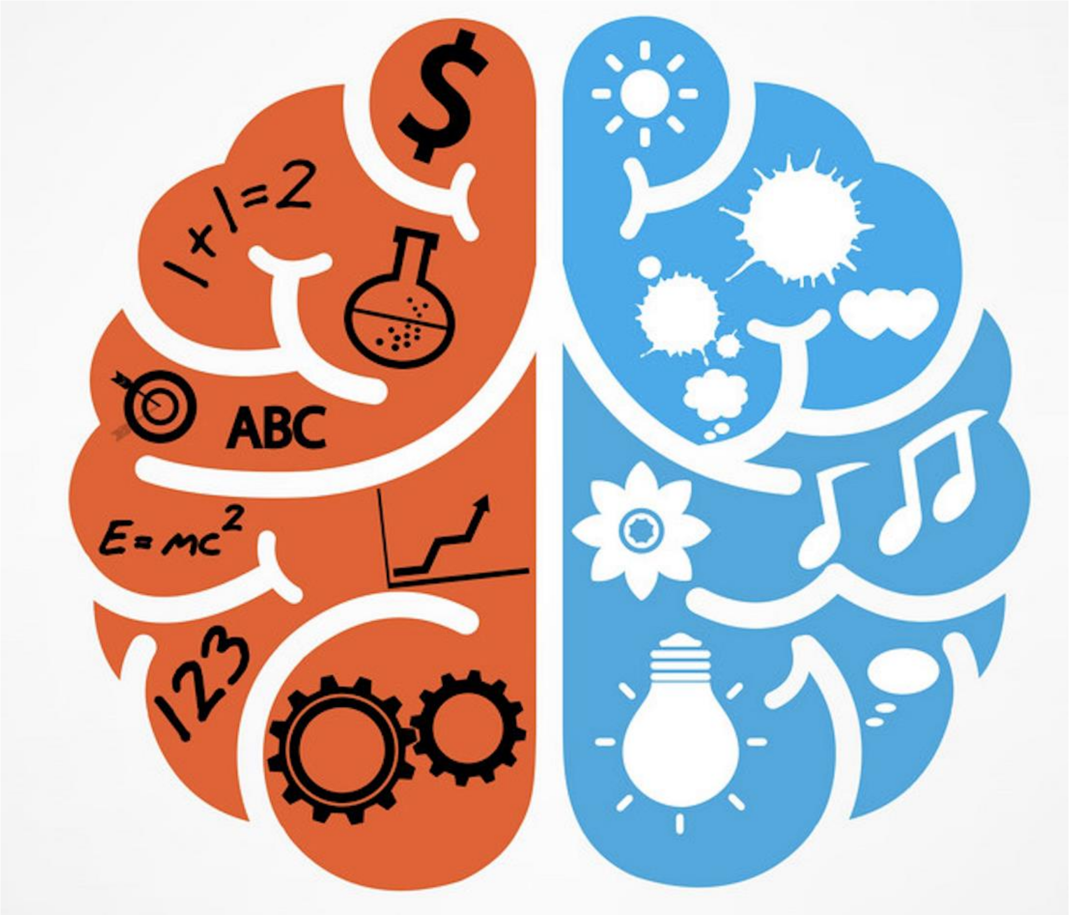
The speaker is responsible for communicating so the listener understands

Communication principles:

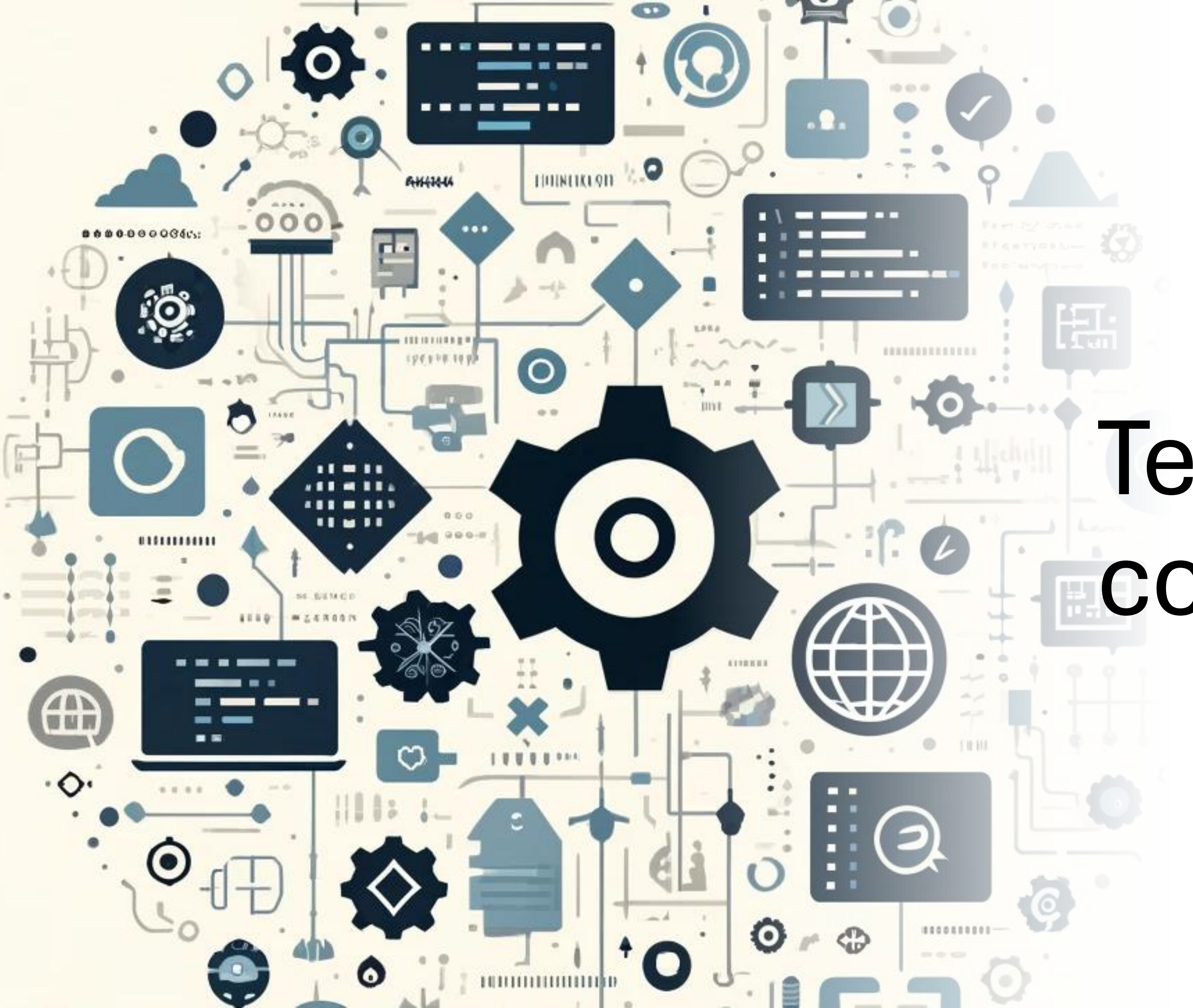
- Tailor communication to the listener
- Consider the emotional context
- Keep it simple
- Establish common vocabulary
- Overcommunicate
- The medium is secondary

# Stories, analogies, and metaphors

Use analytical tools whenever possible…



… but don’t forget stories, analogies, and metaphors!



# Tech is complex

# Tech is engineering

Software engineering is a true engineering discipline
Rooted in mathematics and fueled by continuous scientific progress

## A highly inaccurate and simplified view:

**Hardware:**

- **Transistors**: *Pure electronics, quantum physics rules, no software yet.*
- **Logic Gates**: *Basic operations, e.g. NOT, AND, OR, using electrical current.*
- **Integrated Circuits**: *More complex operations, e.g. adding numbers; hardware and software are still the same.*
- **Processor**: *Software (code) is stored in memory and is executed by hardware.*

**Code:**

- **Assembler**: *Low-level, hardware-specific, e.g. "move" or "add" operations.*
- **Abstract Syntax Tree**: *Abstracted from hardware, introduces conditions (loops, ifs).*
- **Low-level code**: *General-purpose but still for specific hardware, e.g., C, Bytecode.*
- **High-level code**: *Modern programming languages like Scala, C#, JavaScript, abstracting hardware further.*

**Execution Environment:**

- **Kernel**: *Core of the operating system with full control over hardware.*
- **Operating System**: *Managed access to hardware and environment for applications.*
- **Container**: *Isolated environment for applications, simplifying resource management.*
- **Process**: *Running environment of an application's code.*
- **Application**: *The end-user interface, running in a process.*

# Tech is also… unusual?

Mechanical or hardware engineering have tangiable constraints

Software solutions are less grounded in physical reality

There are many different ways to solve the same problem in Tech, without clear criteria for what makes one approach better

# Tech is both

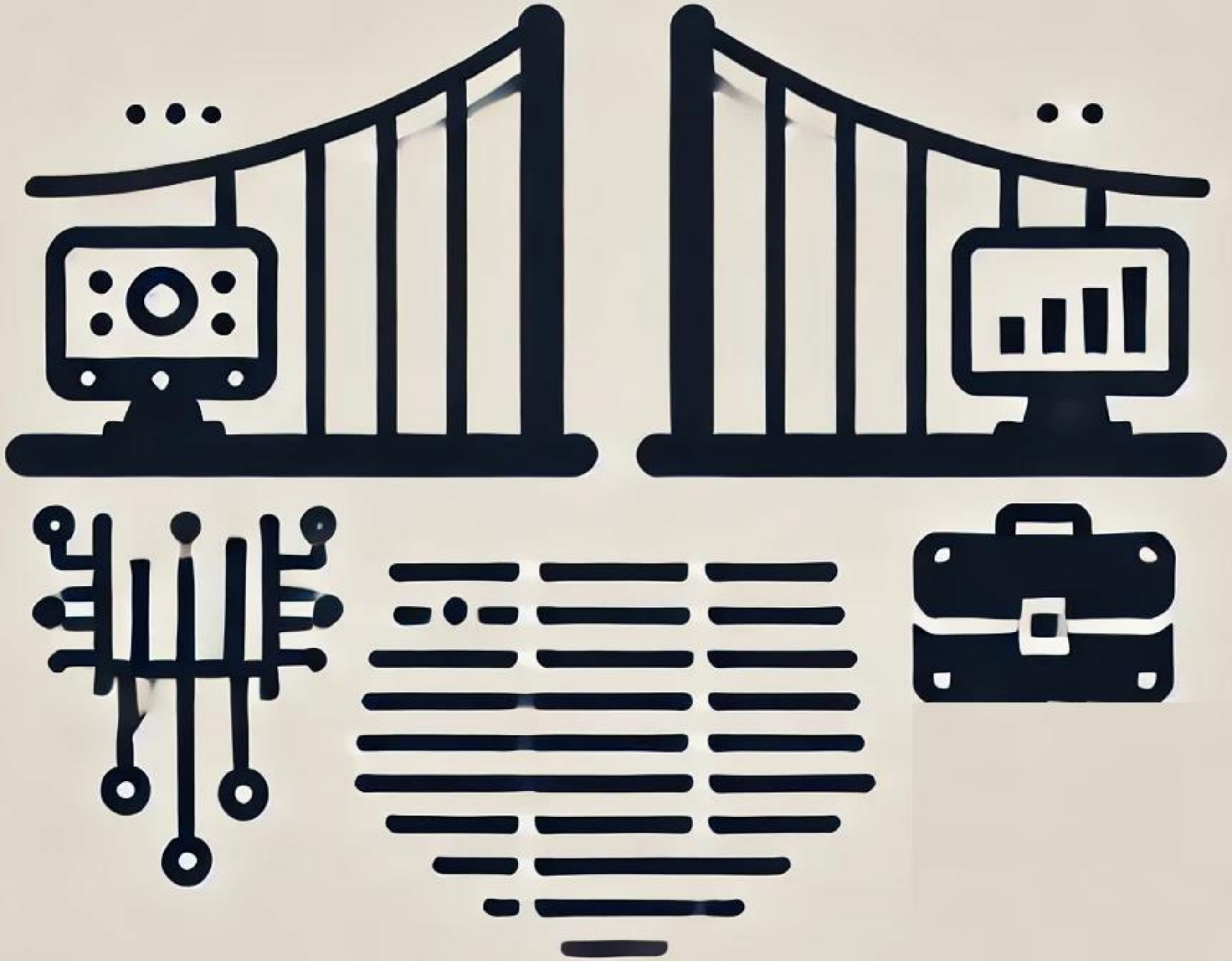
**Engineering**

- Complex, multi-layered, with deep historical roots
- Requires significant expertise and precision
- Focuses on solving problems with structured approaches

**and**

**Art**

- Expressive, with infinite choices and creative freedom
- Has few hard rules, relying more on intuition and vague guidelines
- Explores new problems and builds novel solutions never created before



# Tech
# <>
# Business

# Empathy for the Business

Imagine you are a CEO with an operations background

- You are detail-oriented and understand the business inside out
- When you need more information, you can always dig deeper

Then there is Tech

- You can’t verify quality of software you get
- Tech leaders avoid committing to timelines and struggle to estimate projects
- Jargon is everywhere, and it doesn’t make sense
- Engineers are highly paid, but it’s hard to assess their performance
- Productivity seems next to impossible to measure
- Even tech leaders seem to routinely disagree on key details
- When trying to understand, you only get told that ”Tech is different”

…

**Would you like Tech?**

# Taking a step towards Business

Role of tech leadership?

- Achieve business results by using technology
- Build trust and help the Business to understand Tech better

Communication is key:

- Translate complex tech concepts into accessible language
- Use data, graphs, and models where needed
- Leverage stories, metaphors, and analogies to simplify and connect

# Let’s make it practical

with analogies and metaphors

# Tech systems compared to a building

Common topics:

- Stages of scale: house, building, tower, skyscraper
- Foundations different for 1, 2, 5, 20, or 100 floors
- Quality of materials and construction work
- Building modifications
- Repairs, cleaning, and maintenance

More personable: Home

# Engineering quality compared to an Excel model

Comparison with Excel:

- Multiple sheets – multiple services
- Input and config sheets – deteriorating separation of concerns
- Editing by multiple people – race conditions
- Analyst working on a model alone – onboarding, bus factor
- Editing someone else's excel – readability and maintenance
- Hardcoding into existing formula – code smells
- Formulas vs constants - copy/pasting

# Integrating systems: comparison

In civil engineering:

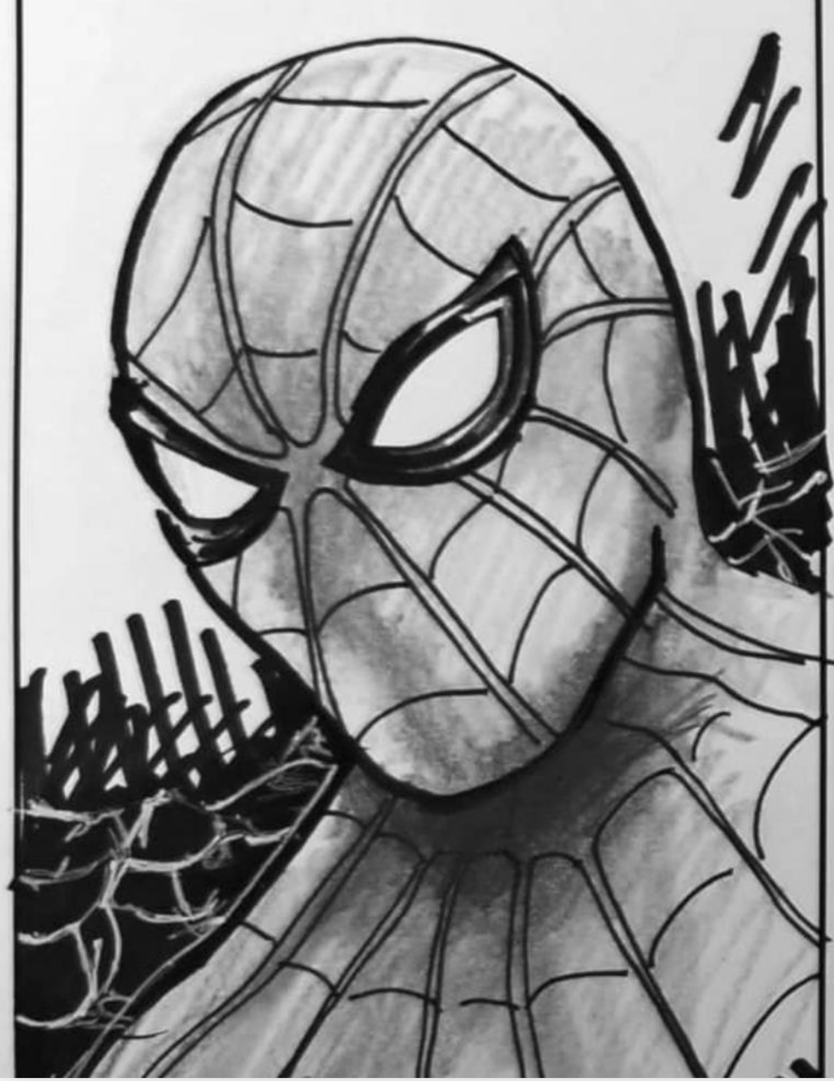
- Building a 200km road with multiple contractors has some integration overhead

In Tech:

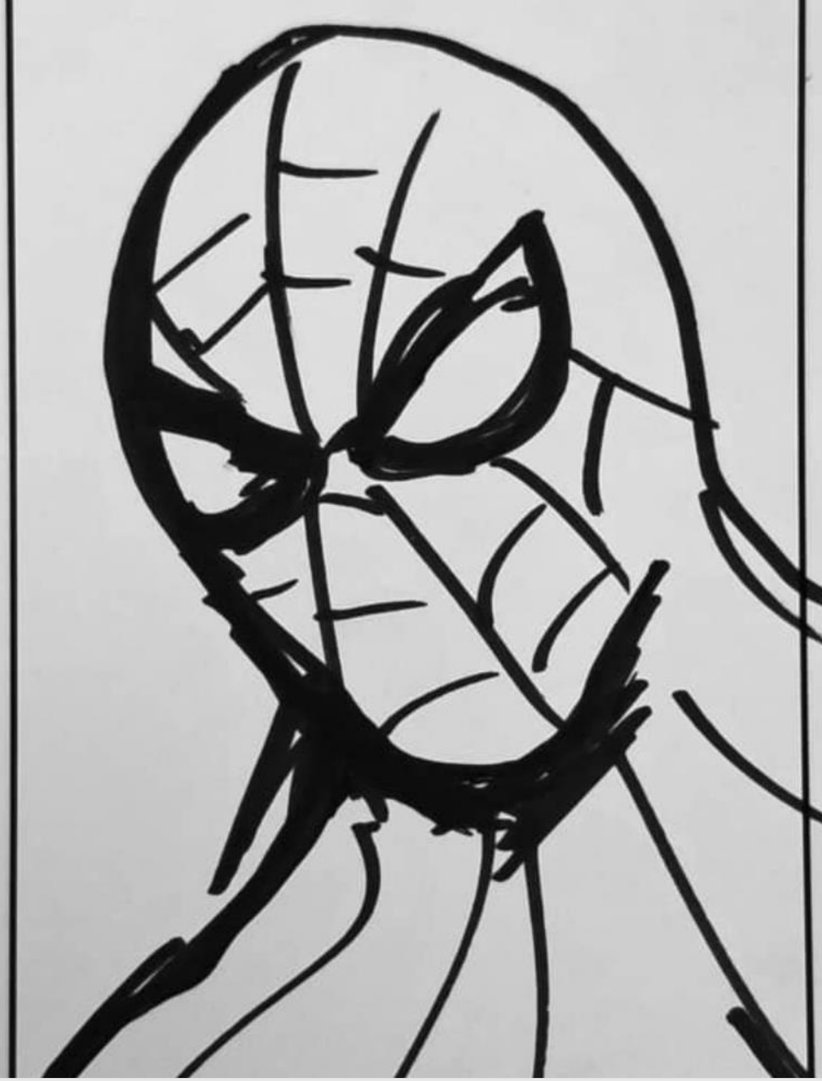
- Integration overhead is much larger
- Often simpler and cheaper to build separate systems than to integrate them
- This is counterintuitive, especially in mergers & acquisitions

Tech challenges:

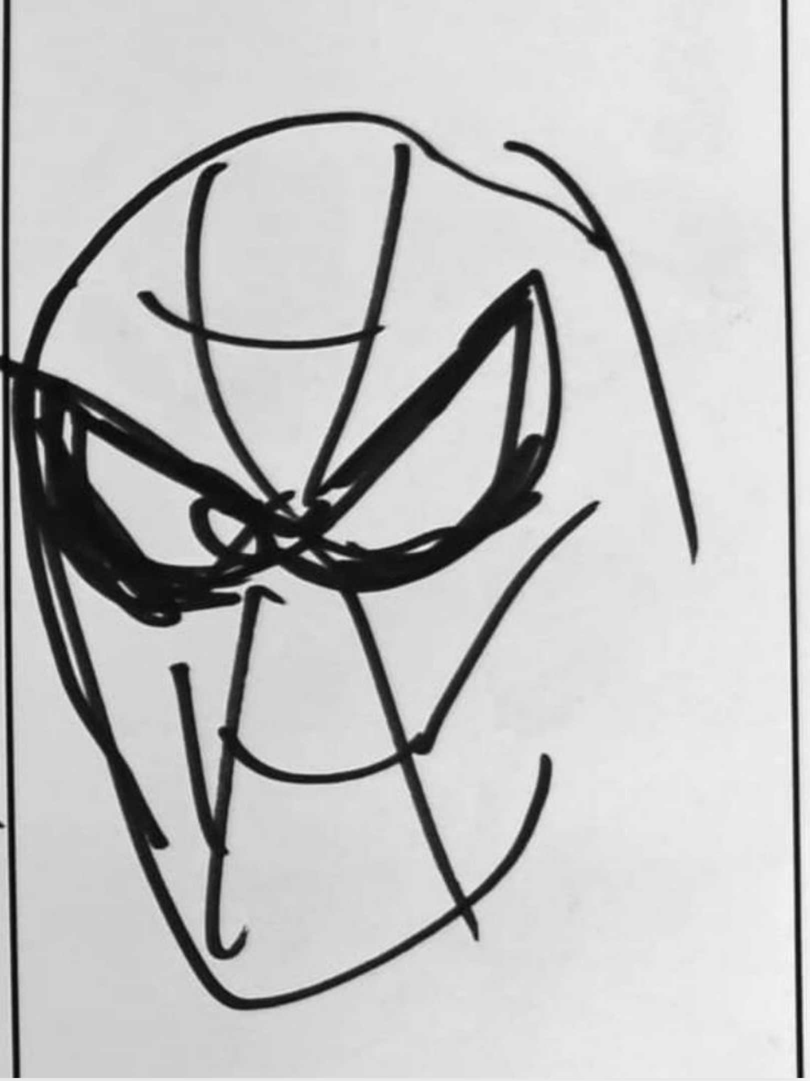
- Few standards or common building blocks
- Systems with the same scope and tech stack often don’t integrate well
- We rely on adapters and lengthy migrations



10 minutes



1 minute



12 seconds

# Quality vs Timeline

# Project variety

Tech projects are rarely similar

- Limited transfer of knowledge between projects

If Tech was civil engineering… projects done by the same team:

- Highway from Munich to Berlin
- Seven houses in Brandenburg next to a lake
- Skyscraper with 79 floors
- New airport
- Warehouse 10000m$^2$
- Mall

# Planning challenges

Seven houses in Brandenburg next to a lake

- All houses identical?
- How big?
- Which lake?
- Access to electricity, water, gas, and waste removal?
- Sand or rock ground?
- Architectural style?

Estimation needs precise requirements

- Surprises can still happen, like discovering an aquifer while digging
- Construction workers solve small problems, software crashes

# Conclusion

# Create your own analogies

More examples:

- Crowded dance floor – adding people decreases productivity
- Messy bookshop – assessment or due diligence
- Mechanics' workshop – standardisation, reuse, and duplication
- Jenga tower – deteriorating system and decreasing velocity

Create your own stories, analogies, and metaphors!

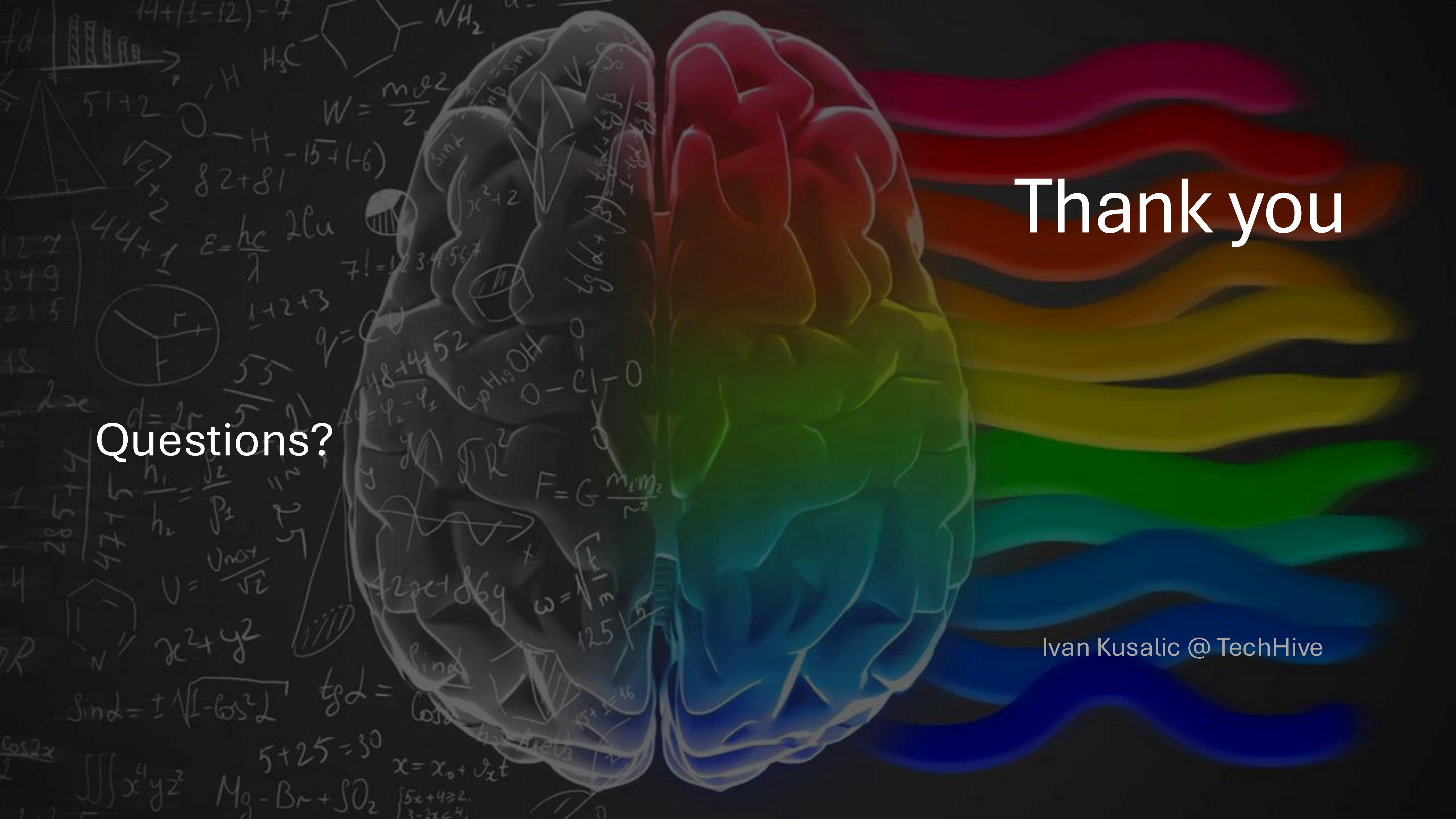
- Repeat and expand until they are part of the common vocabulary

# Bridge the gap

Back up your message with analysis and facts…

… but don’t forget to:

- Connect through storytelling
- Engage the intuition
- Communicate with empathy
- Tap into emotions and personal experiences



# Thank you

Questions?

Ivan Kusalic @ TechHive