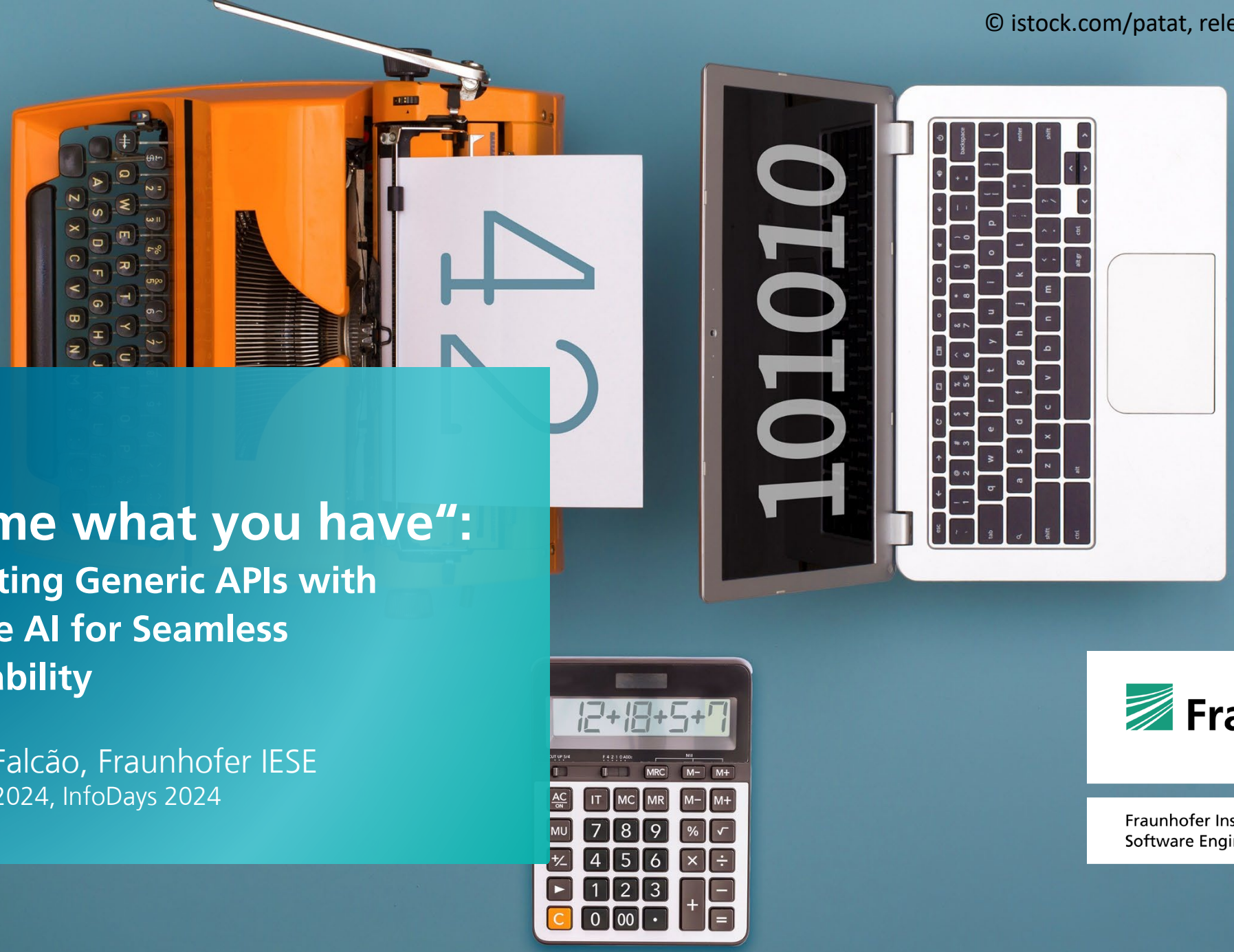


“Send me what you have“:
Implementing Generic APIs with Generative AI for Seamless Interoperability

Dr. Rodrigo Falcão, Fraunhofer IESE

September 26, 2024, InfoDays 2024

Fraunhofer IESE

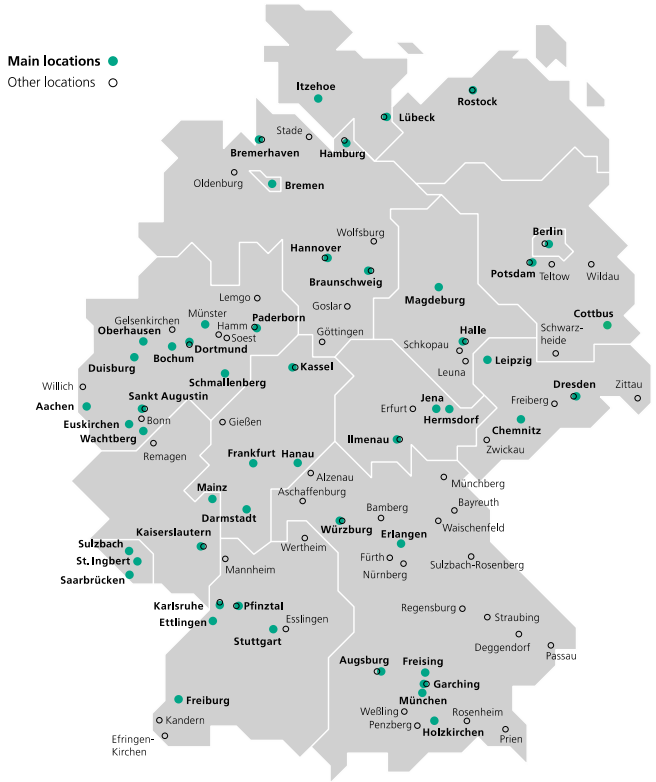
Fraunhofer Institute for Experimental Software Engineering IESE

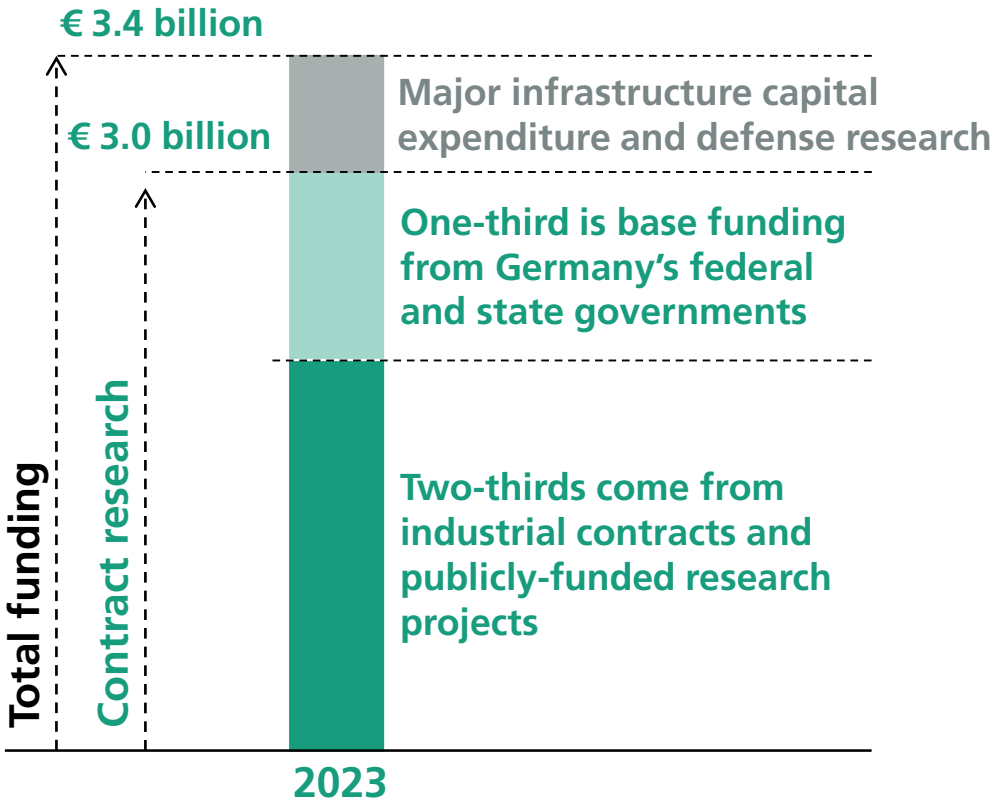
The Fraunhofer-Gesellschaft at a glance

Applied research of direct utility to private and public enterprise and of wide benefit to society

76 institutes and research units

~32,000 staff



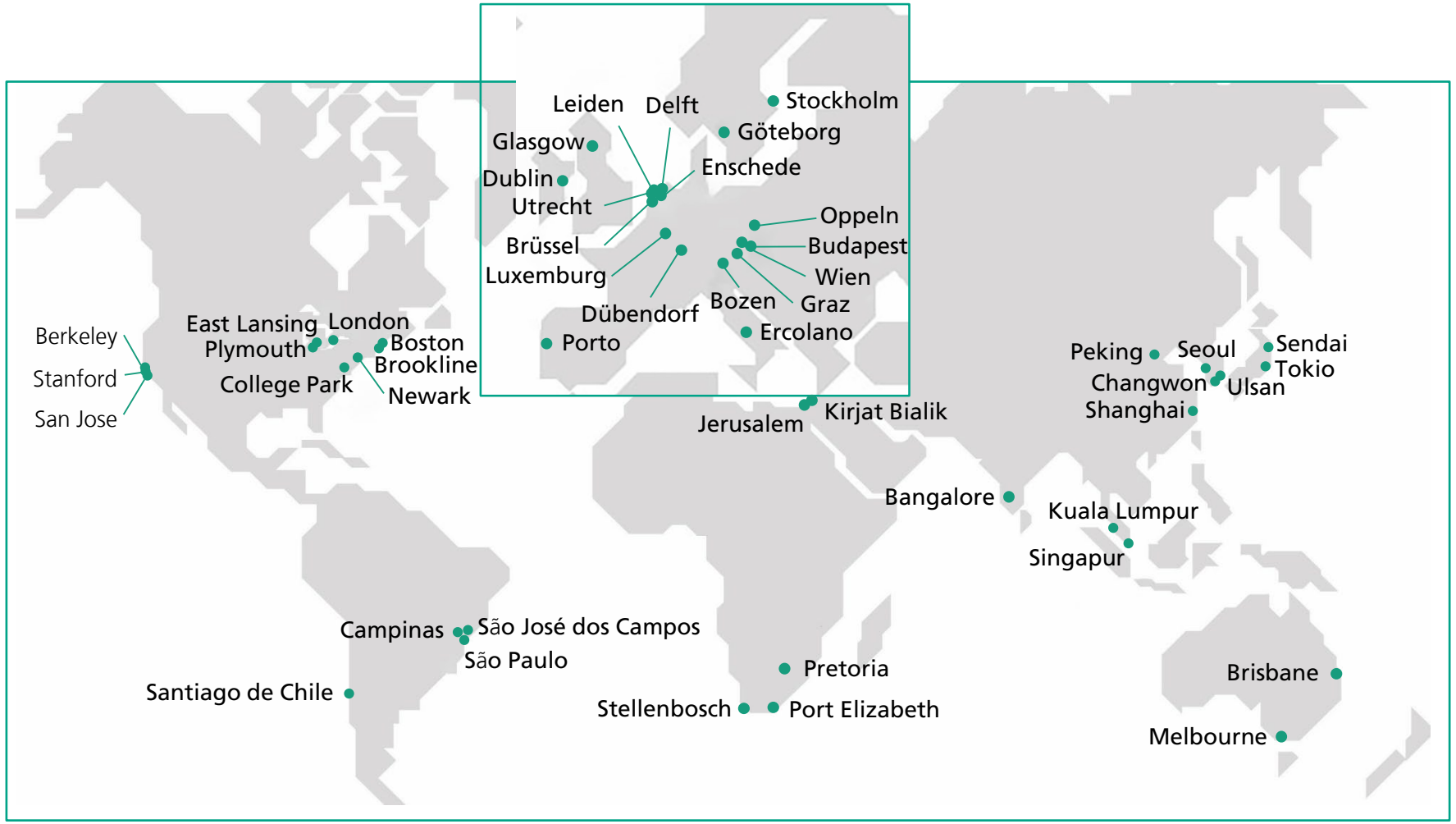


Fraunhofer

Fraunhofer IESE

International network

- Eight independent Fraunhofer affiliates
- Active with partners in approximately 80 countries
- Representative Offices and Senior Advisors worldwide leverage networks abroad



Fraunhofer IESE

The institute for software engineering, systems engineering, and innovation engineering

Founded **1996**

headquartered in
Kaiserslautern

2 locations:

Kaiserslautern and
Berlin Liaison Office

3 strategic project centers:

Potsdam (Germany), Riverdale Park (USA)
and Vancouver (Canada)

Over **270** staff

from more than **15** nations

over

2.000 projects

About me…

- **Dr.-Ing. Rodrigo Falcão, PMP**
- Researcher and Project Manager at Fraunhofer IESE, Kaiserslautern, Germany
- Software architecture
- Lecturer of Software Architecture at Mannheim University of Applied Sciences, Germany
- ~15 years of industry experience prior to stepping into research
- **Lead researcher for “Generative AI in Software Architecture”**

Agenda

- Interoperability 101
- An example scenario
- The problem
- Generative AI
- What if…
- Experiences
- Consequences
- What's next?

Interoperability 101

Quality is a core concern for stakeholders in a software project

Stakeholders‘ concerns

Functionality

- “WHAT“
- Key features of my system
- Makes my system unique

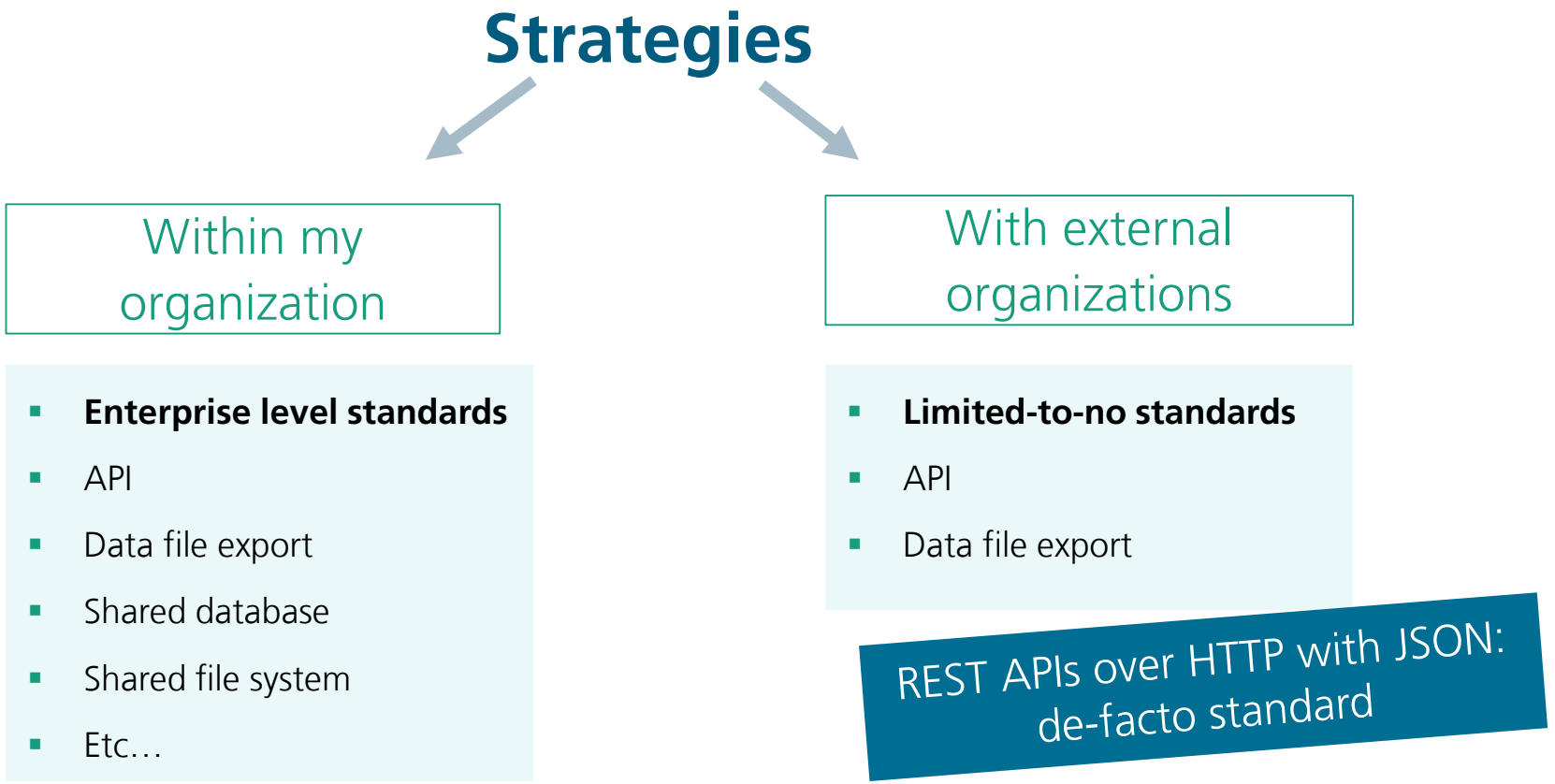
Quality

- “HOW GOOD“
- Maintainability
- Reliability
- Correctness
- …
- **Interoperability**

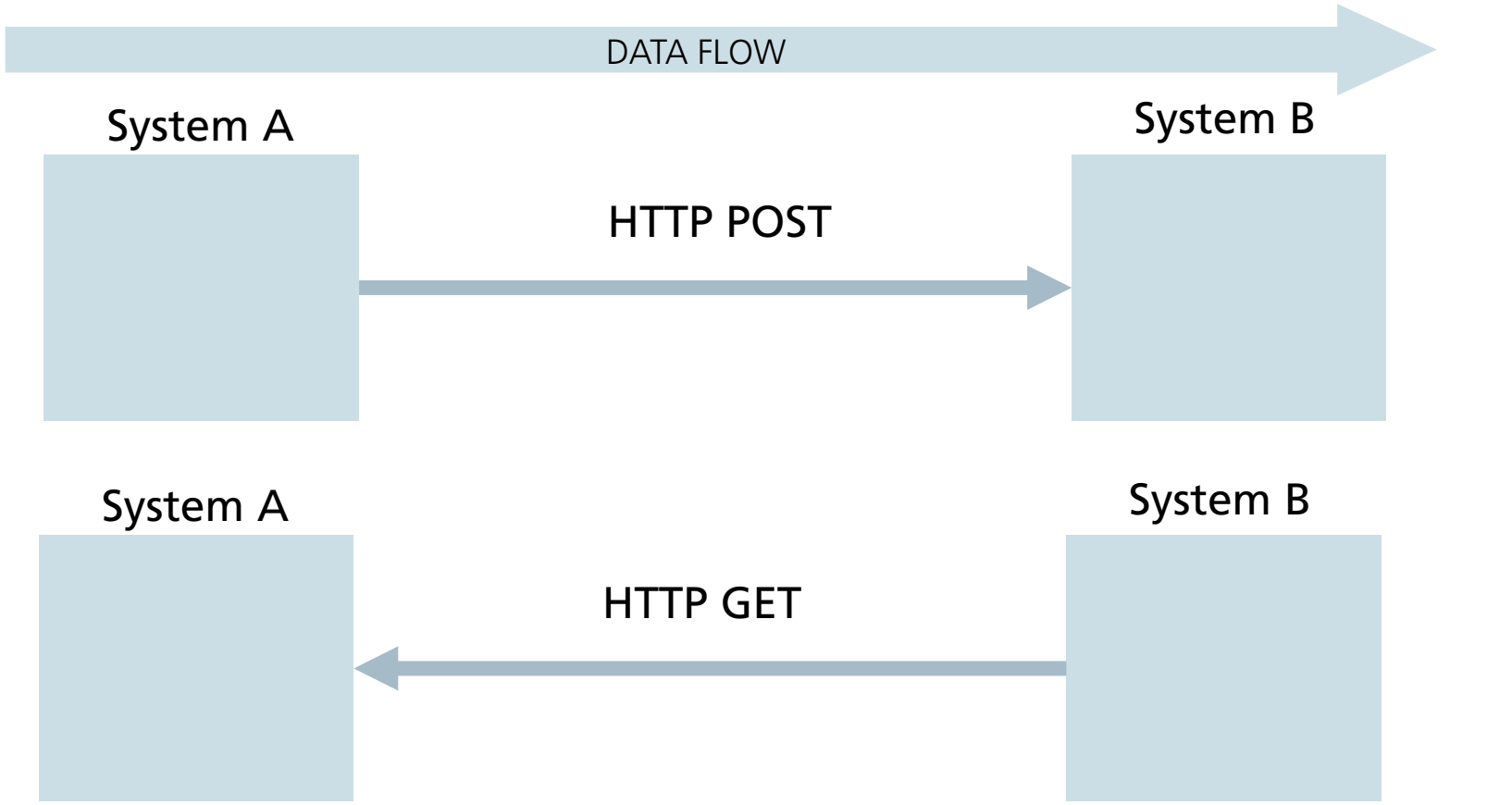
Interoperability refers to the ability of two systems to exchange and use data.

-- (loosely based on ISO 25010/2023)

APIs are among the most traditional strategies for data exchange with external organizations



When a system needs another system's data, it can either expose or consume an API



Other designs are possible, but you got the idea

© iStock.com/Scharfsinn86

An example

(Agricultural Domain)

Agriculture is a huge domain with several interoperability challenges

Agricultural domain

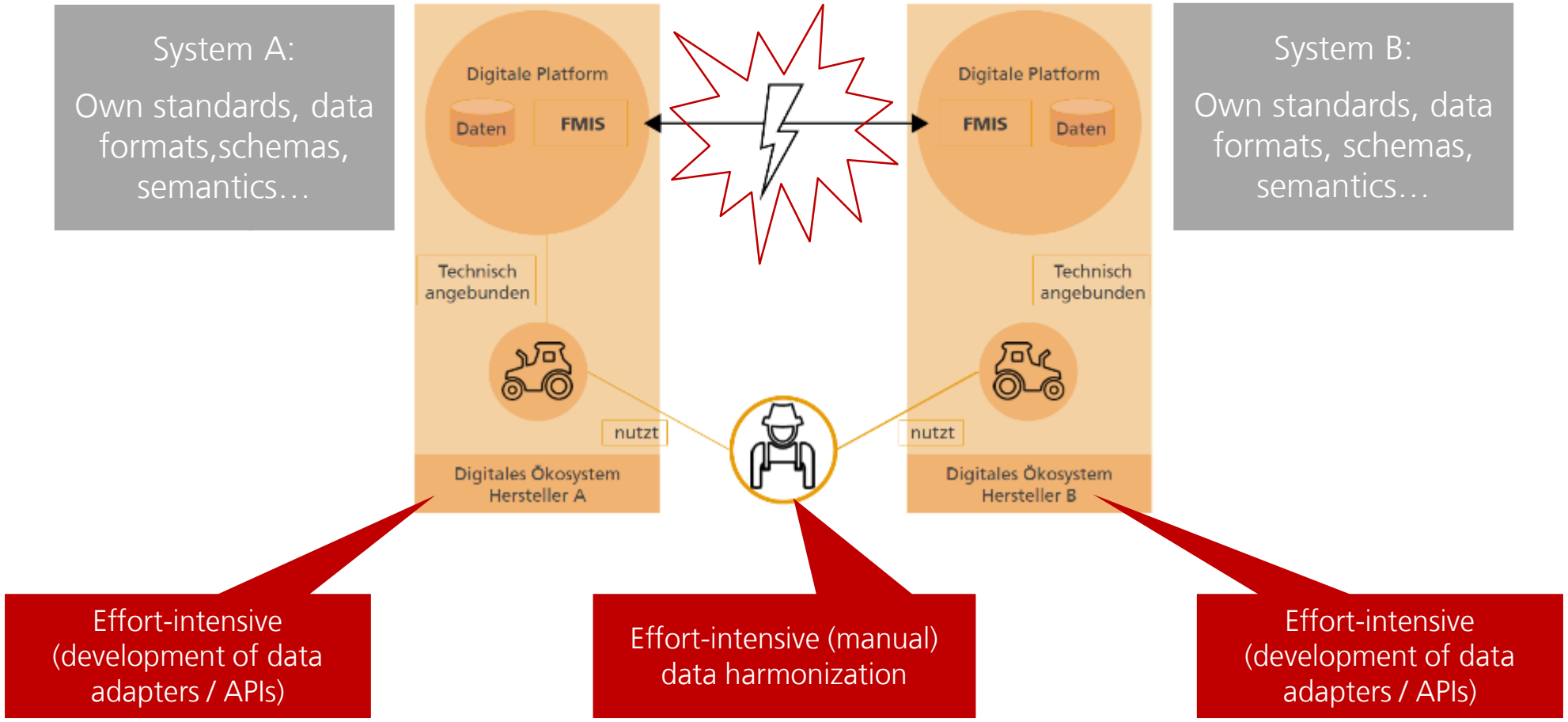
Huge

- Several subdomains
- Several processes
- Several formats
- Projects are silos
- …

Interoperability challenges

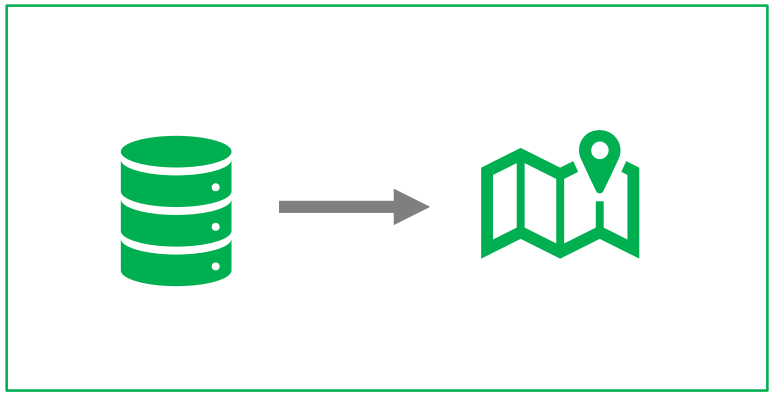
- Lack of standards
- Organizational challenges
- Technology heterogeneity
- Data quality

The agricultural sector faces interoperability challenges



Two solution providers implement Farm Management Information Systems to support farming activities

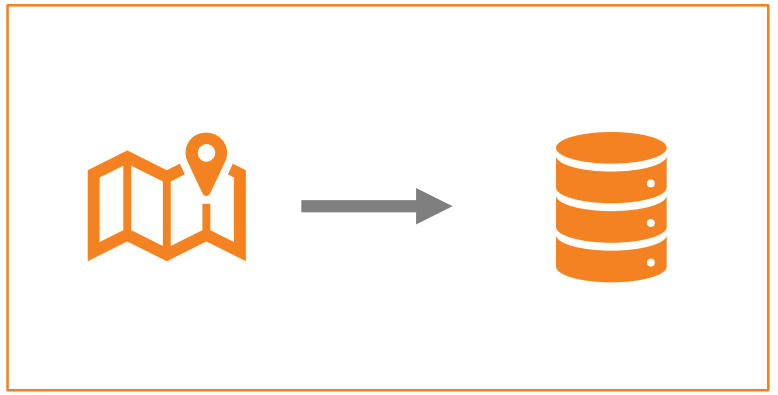
FMIS 1



has field data

using its own schema

FMIS 2



has its own representation of field data

Field boundaries are not the most trivial data type

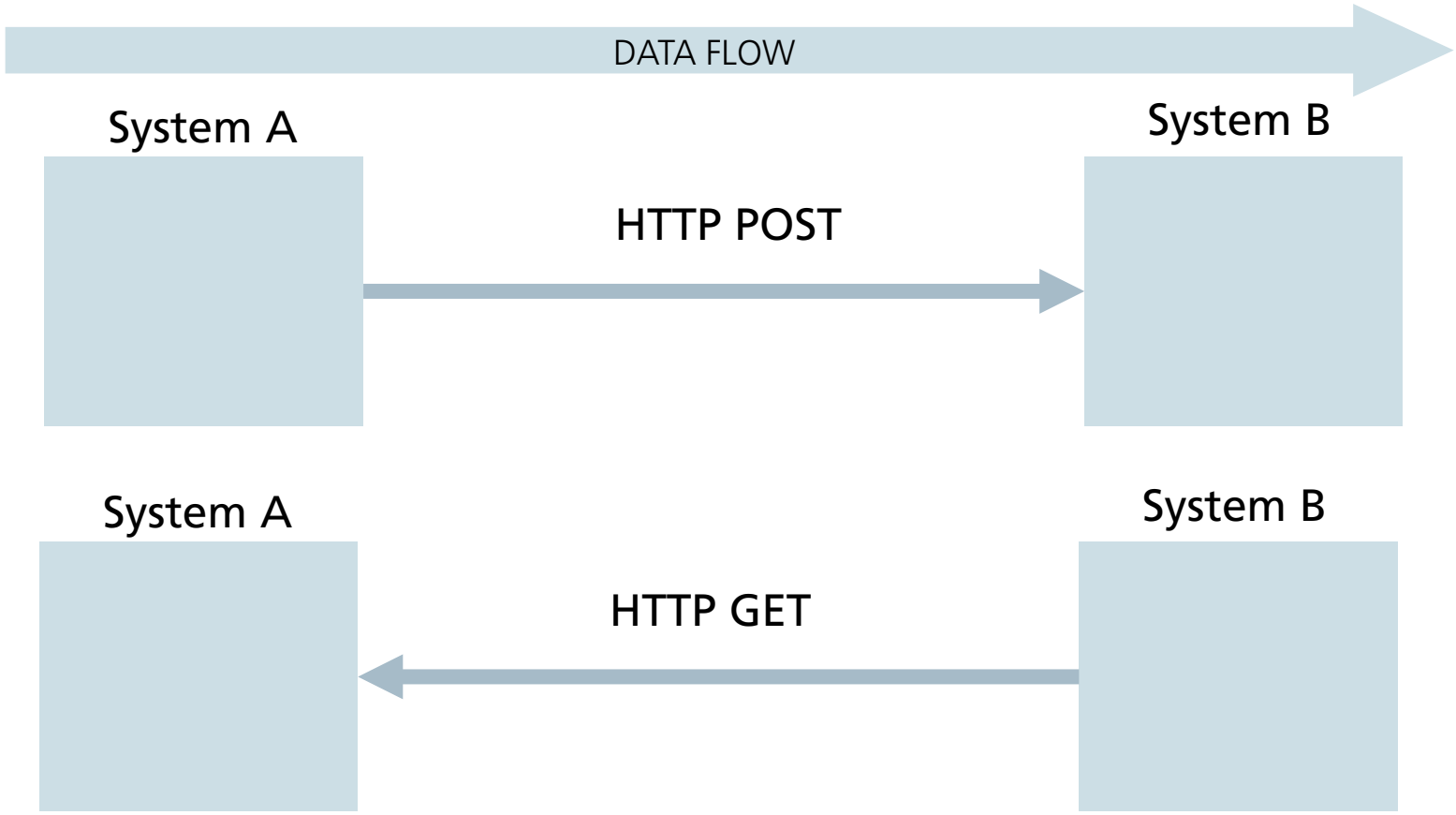
Trivial

- Customer record
- Product inventory item
- Address book entry
- Student record
- …

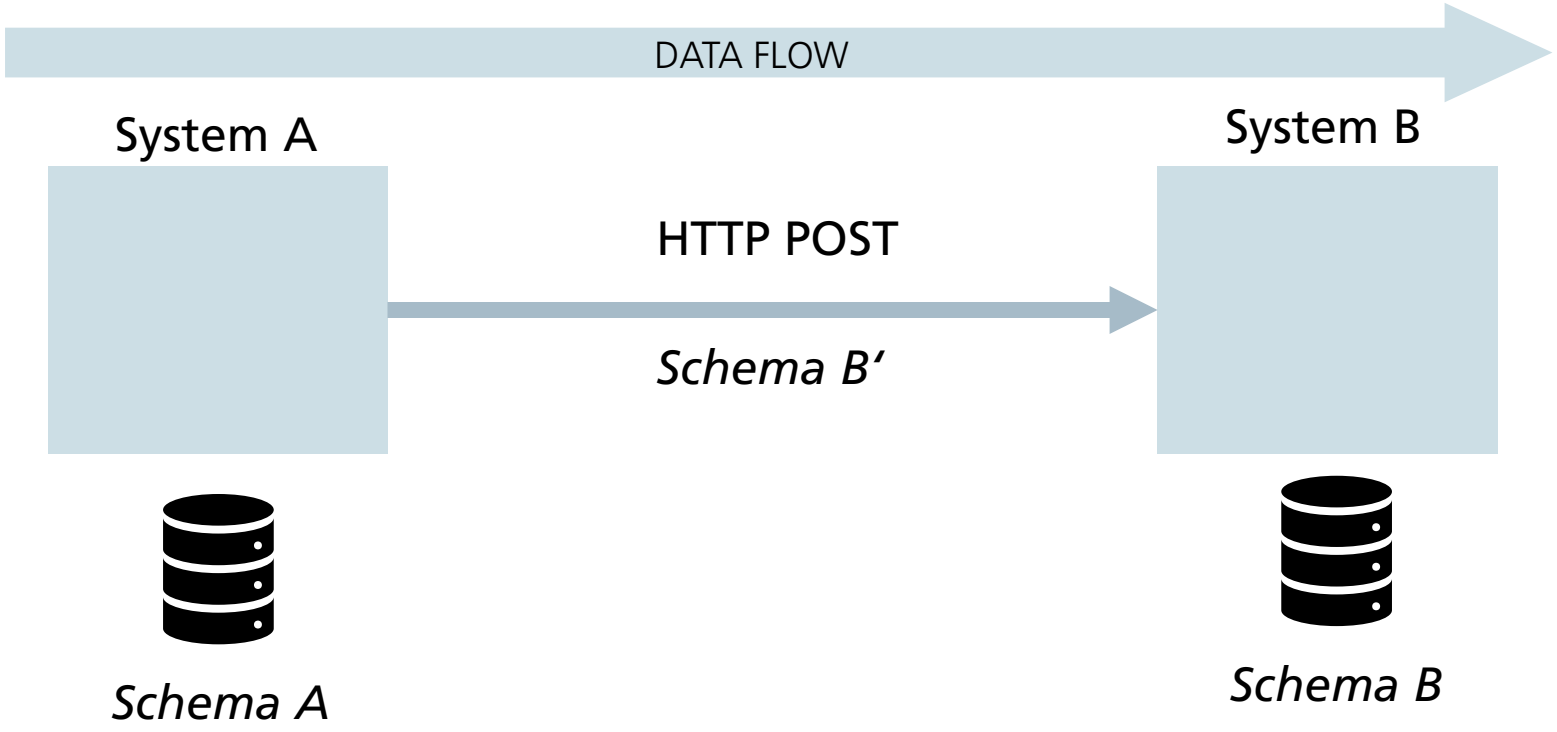
Non-trivial

- Geographic boundaries representation
- Multi-dimentional array of image data
- Graph representation of field network
- …

The interface can be designed in different ways



Consider that system B expose an API to receive field data



© iStock.com/wakila

The problem

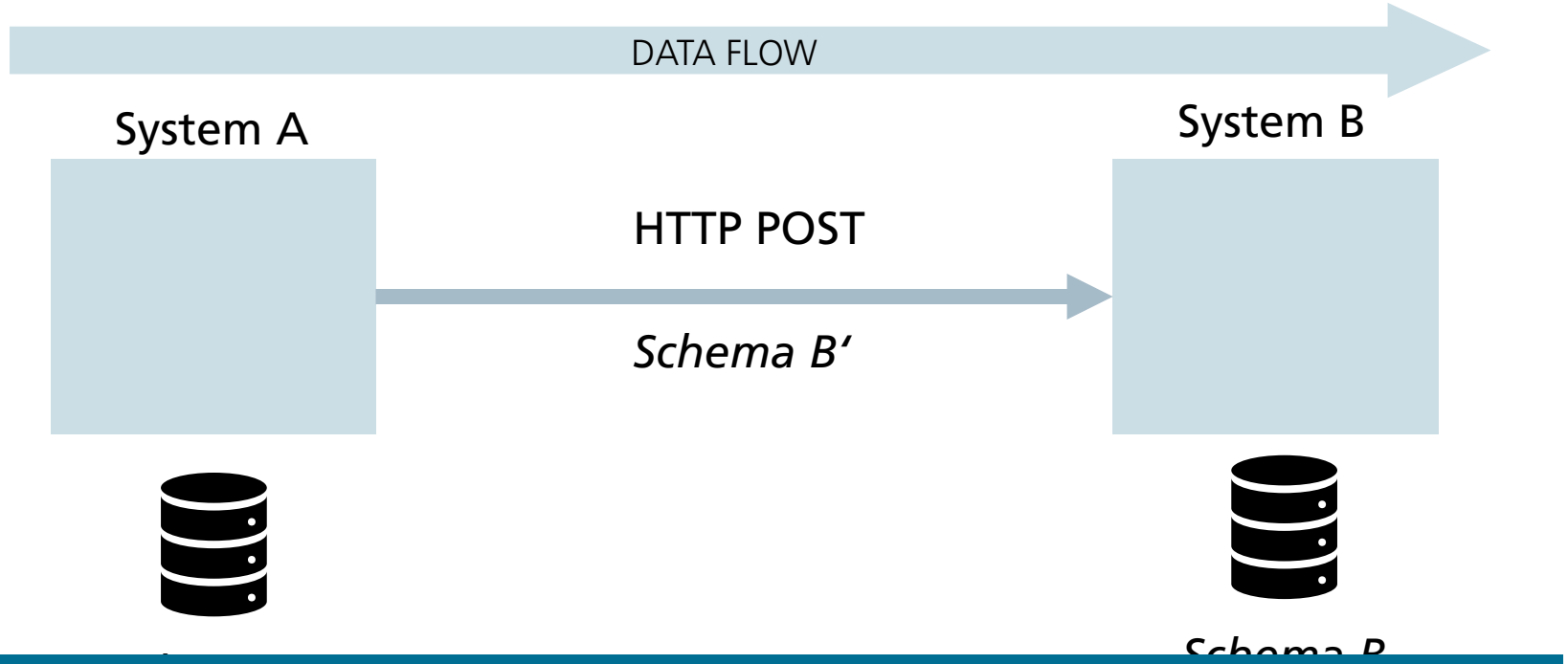
Achieving interoperability via APIs comes at a cost: implementation efforts

API consumers must invest efforts in complying with the API requirements

Efforts

	Providers	Consumers
Functional aspects	▪ Define format ▪ Define schema	▪ Comply with requirements ▪ Convert data
Quality aspects	▪ take care of reliability	▪ must consider latency, fault tolerance, etc.

Who is willing to take the effort?



Typically, it is better to expose the API.

If I want field data, the dream is that I expose my API and everyone complies with it!

(But this is everyone's else dream!)

In the aftermath, it's not about exposing or consuming APIs; it's about the effort needed to understand a foreign schema and implement adapters for data conversion.

What if…

WHAT IF…

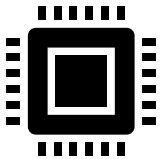
…our systems were smart to the point that new participants could join the agricultural ecosystem without development efforts to adapt interfaces?

Recently, we at IESE have explored achieving interoperability at runtime

Interoperability

DEV TIME

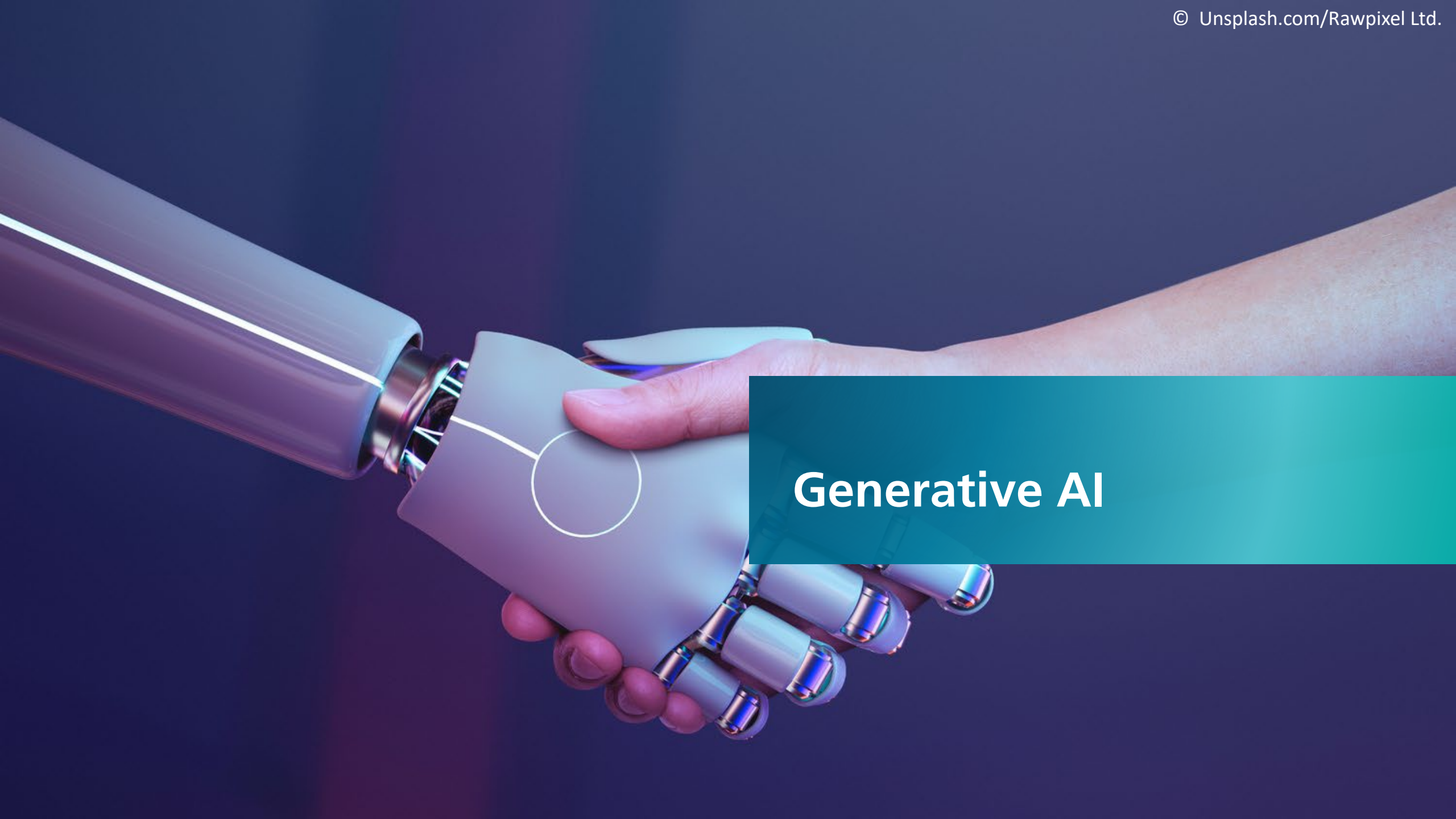
- Human effort to understand the interfaces, implement adequate data converters (a.k.a. “adapters“), test, deploy, etc.
- Deterministic solution
- Time to market ☹



RUNTIME

- What if AI can do at runtime what humans would have to do at dev time?
- Can LLMs implement effective adapters at runtime?

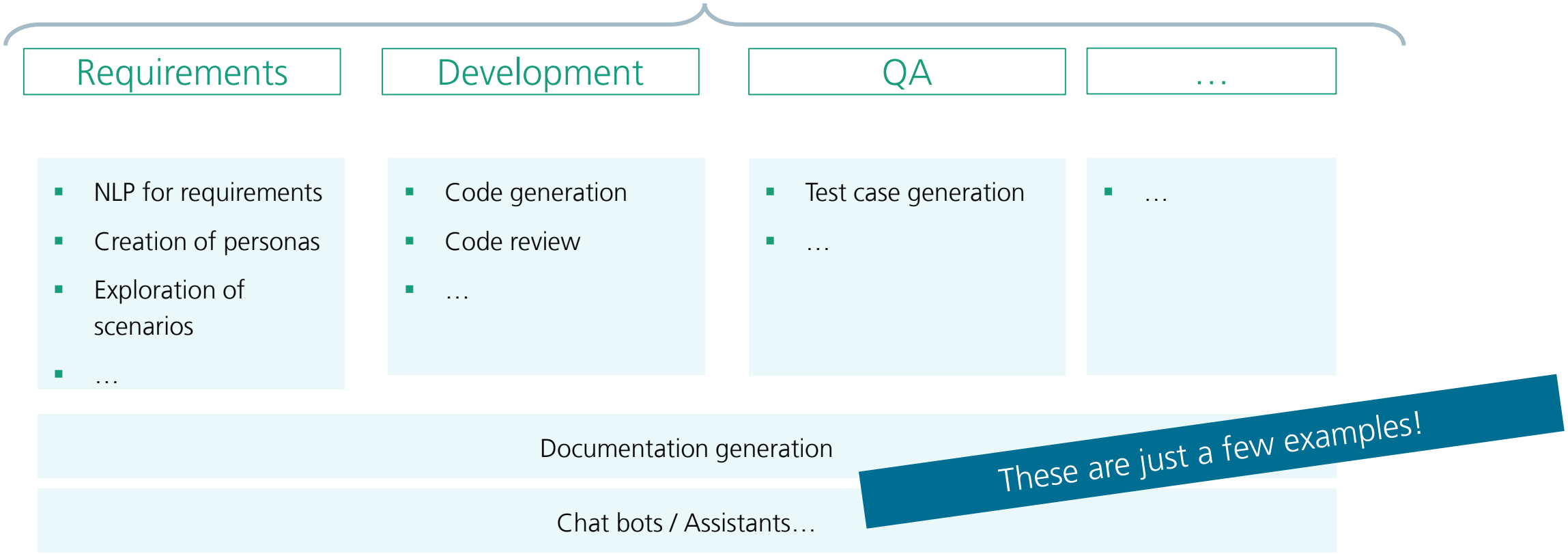
We have tested this idea in the agricultural domain



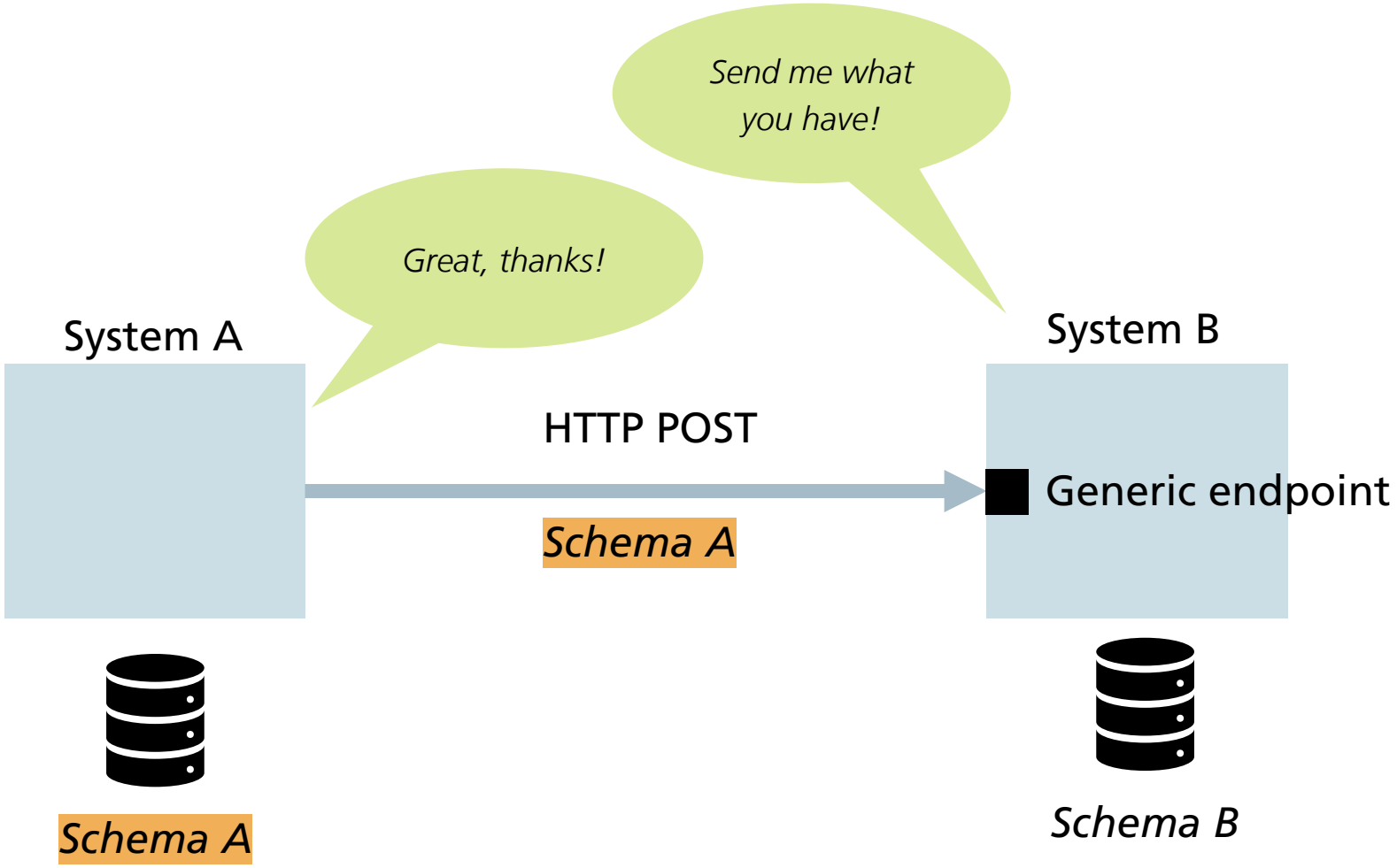
Generative AI

An LLM is a probabilistic model trained on extensive data to generate meaningful word sequences.

There are several application fields for LLMs in SE



Imagine that System A must no longer concerned about B's schema



With the rise of LLMs, there is an open door for the design of “creative systems”— systems that can perform human-like tasks to achieve qualities such as interoperability at runtime.

How does it look like for a customer record?

Schema X


```
{
  "customerData": {
    "customerId": 67890,
    "name": {
      "first": "Jane",
      "last": "Smith"
    },
    "contactInfo": {
      "emailAddress": "janesmith@domain.com",
      "phone": "+1-555-9876"
    },
    "address": "200 Elm St, Apt 12A, Metropolis, NY, 10001, USA",
    "birthDate": "1990-09-23",
    "loyaltyTier": "Platinum",
    "signupDate": "2021-05-22"
  }
}
```


Schema Y


```
{
  "customer": {
    "id": 12345,
    "firstName": "John",
    "lastName": "Doe",
    "email": "johndoe@example.com",
    "phoneNumber": "+1-555-1234",
    "address": {
      "streetName": "Main Street",
      "streetNumber": "100",
      "apartmentNumber": "5B",
      "city": "Springfield",
      "state": "IL",
      "zipCode": "62704",
      "country": "USA"
    },
    "dateOfBirth": "1985-06-15",
    "membershipLevel": "Gold",
    "registrationDate": "2022-01-10"
  }
}
```


How does it look like for a customer record?

Schema X

```
{
  "customerData": {
    "customerId": 67890,
    "name": {
      "first": "Jane",
      "last": "Smith"
    },
    "contactInfo": {
      "emailAddress": "janesmith@domain.com",
      "phone": "+1-555-9876"
    },
    "address": "200 Elm St, Apt 12A, Metropolis, NY, 10001, USA",
    "birthDate": "1990-09-23",
    "loyaltyTier": "Platinum",
    "signupDate": "2021-05-22"
  }
}
```

LLM

@runtime

Schema Y

```
{
  "customer": {
    "id": 67890,
    "firstName": "Jane",
    "lastName": "Smith",
    "email": "janesmith@domain.com",
    "phoneNumber": "+1-555-9876",
    "address": {
      "streetName": "Elm St",
      "streetNumber": "200",
      "apartmentNumber": "12A",
      "city": "Metropolis",
      "state": "NY",
      "zipCode": "10001",
      "country": "USA"
    },
    "dateOfBirth": "1990-09-23",
    "membershipLevel": "Platinum",
    "registrationDate": "2021-05-22"
```

Prompt: “Convert data from format: <input> to format: <output example>“

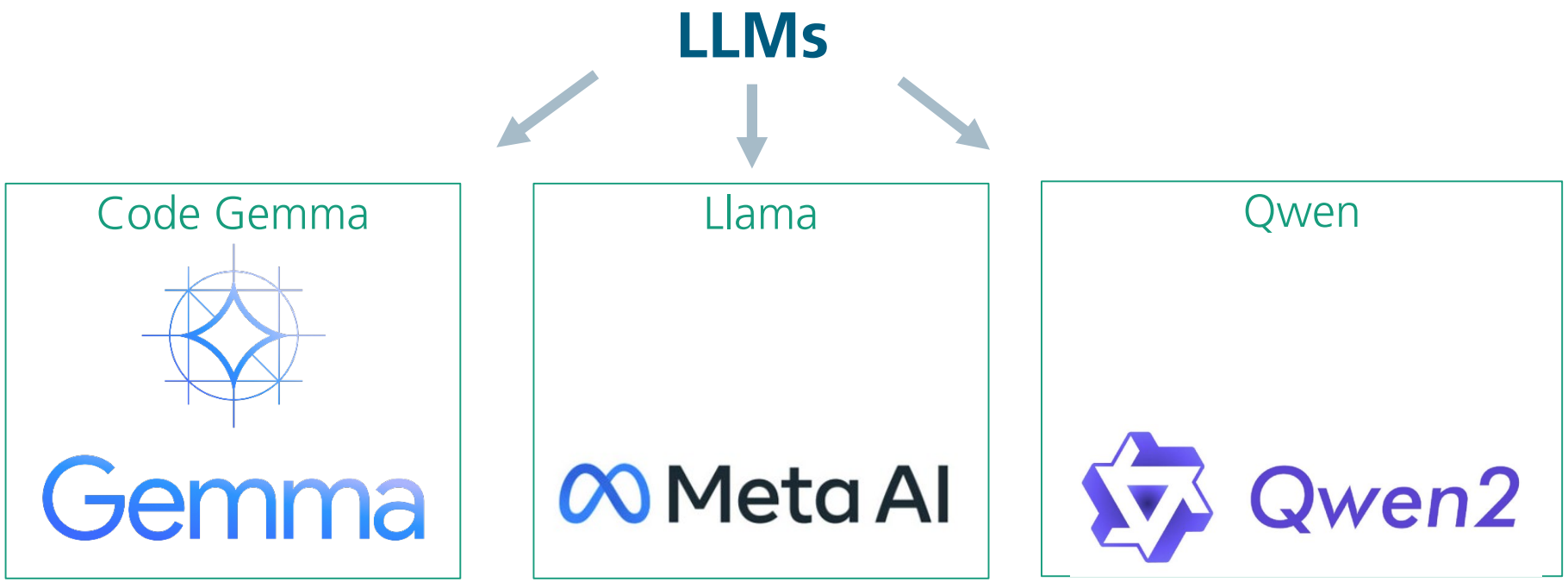

©iStock.com/Ekkasit919
Experiences

We tested two strategies to convert real field data from a proprietary schema

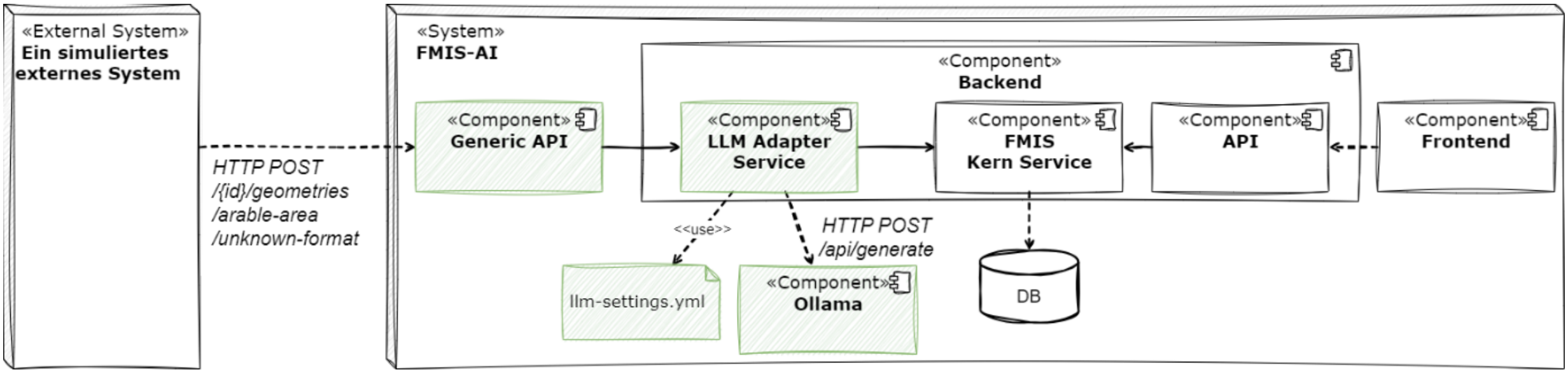
Tests

- Strategies →
  - **Direct Conversion**
  - **Conversion Module Generation**
- Real data →
  - 123 real field boundaries of agricultural fields
  - Proprietary data format of major company in the domain

We have tried out three LLMs to implement the adapters at runtime



We created a demonstrator to illustrate the strategy “direct conversion“



We used zero-shot prompting in our experiment

Prompting

- **Zero-shot** → Just prompt the models once to get the response — Our approach
- **Few-shot** → Provide the models with a few examples before you go — Not applicable
- **Fine-tuning*** → Use datasets to refine base models — Effort intensive

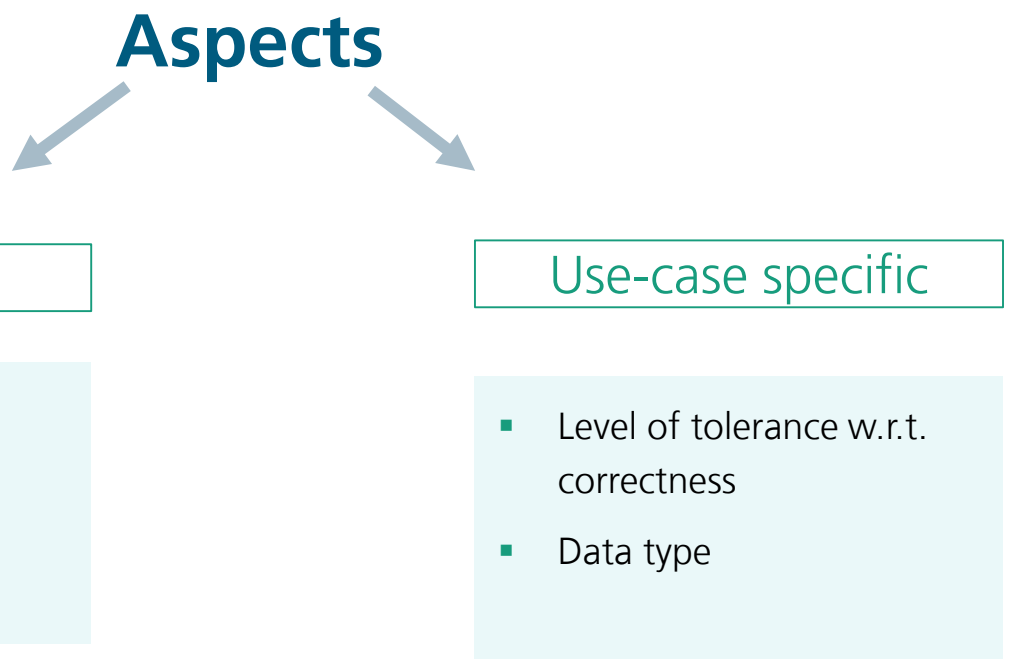
The results???

IT WORKS!

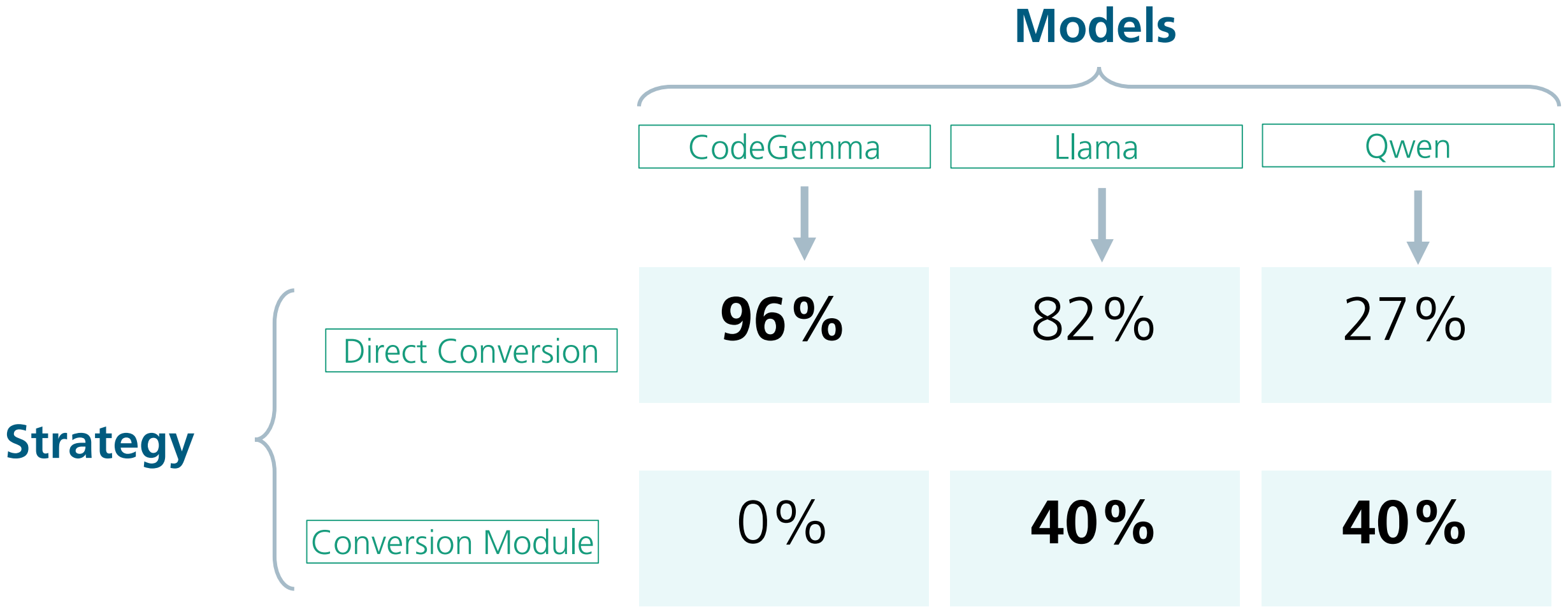
IT WORKS, **HOWEVER**

However, the details make the difference, as many design decisions must be made when setting up an LLM-based interface.

Some aspects are generic, while others are use-case specific

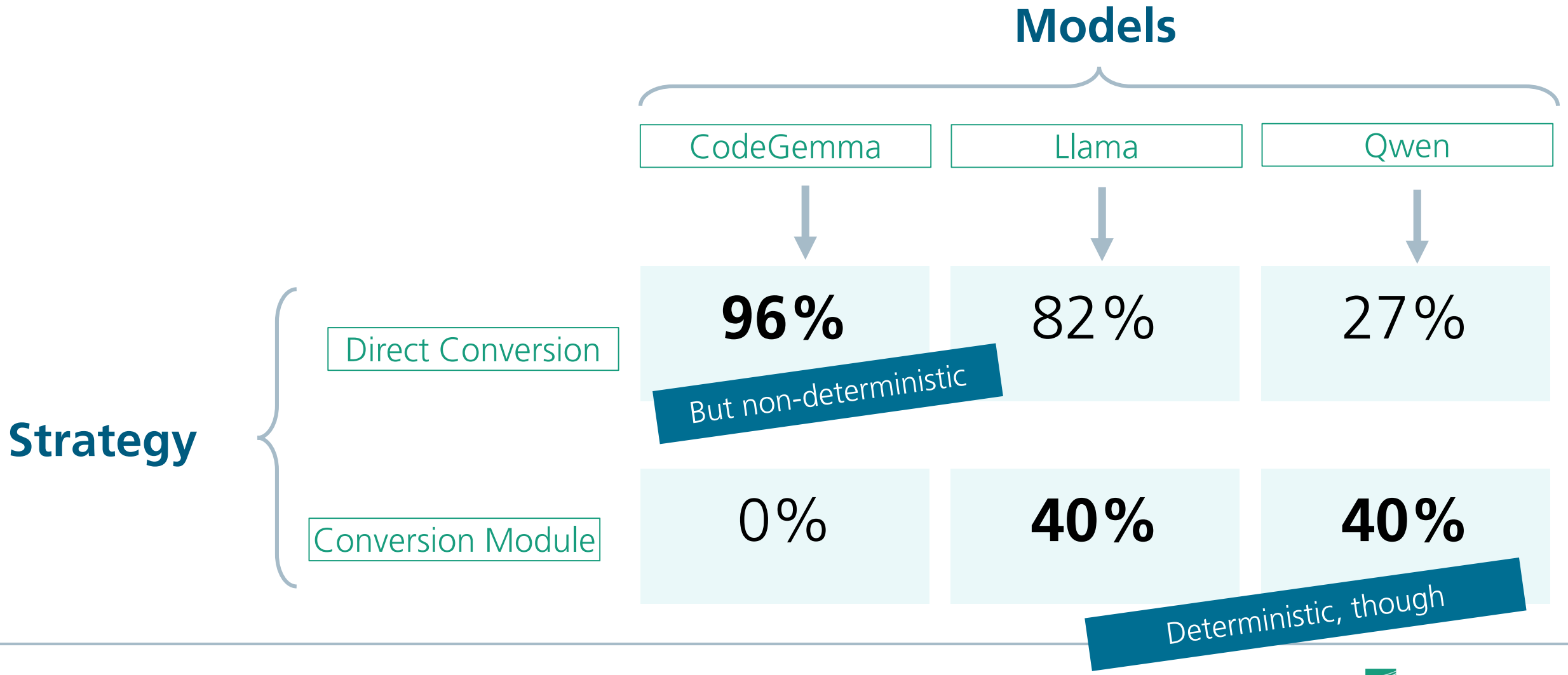


We tested models from the families Gemma, Llama, and Qwen using two strategies



Who’s the best?

We tested models from the families Gemma, Llama, and Qwen using two strategies



Consequences

It is possible to implement effective APIs for data conversion with LLM-based strategies

[Generic] APIs

"Effective" →

- 96%; can it be considered effective enought in my use case?
- Deterministic solutions can be generated

LLM-based strategies →

- Direct Conversion
- Conversion Module Generation

This can save integration efforts, reduce time to market, facilitate the onboarding of partners in your digital ecosystem, enable dynamic integration in open environments, etc.

No “one-size fits all”

Software engineers are required!

What's next?

Next steps includes implementation, expansion, and application in industry

Next steps

- **Implementation** → Our current demonstrator implements only the strategy “Direct Conversion“, we want to have it implementing “Conversion Module Generation“ as well
- **Expansion** → We have initially focused on the agricultural domain because we have developed several projects in the domain in recent years; Other domains will be investigated (e.g., smart cities)
- **Industry** → We are already in contact with industry partners to help them pioneer on the engineering of LLM-based APIs

Thank you for your attention! (but wait!)

This topic will feature October's issue of ITSpektrum

„Architekturtrends – alles bleibt anders“ (25.10.2024)

Stay in touch!

Dr. Rodrigo Falcão
Fraunhofer IESE

rodrigo.falcao@iese.fraunhofer.de

linkedin.com/in/rmfalcao