TOPLOGIC

THE AUTOMATED APPLICATION ENGINE

@ COOL

COMPACT OSADL ONLINE LECTURES

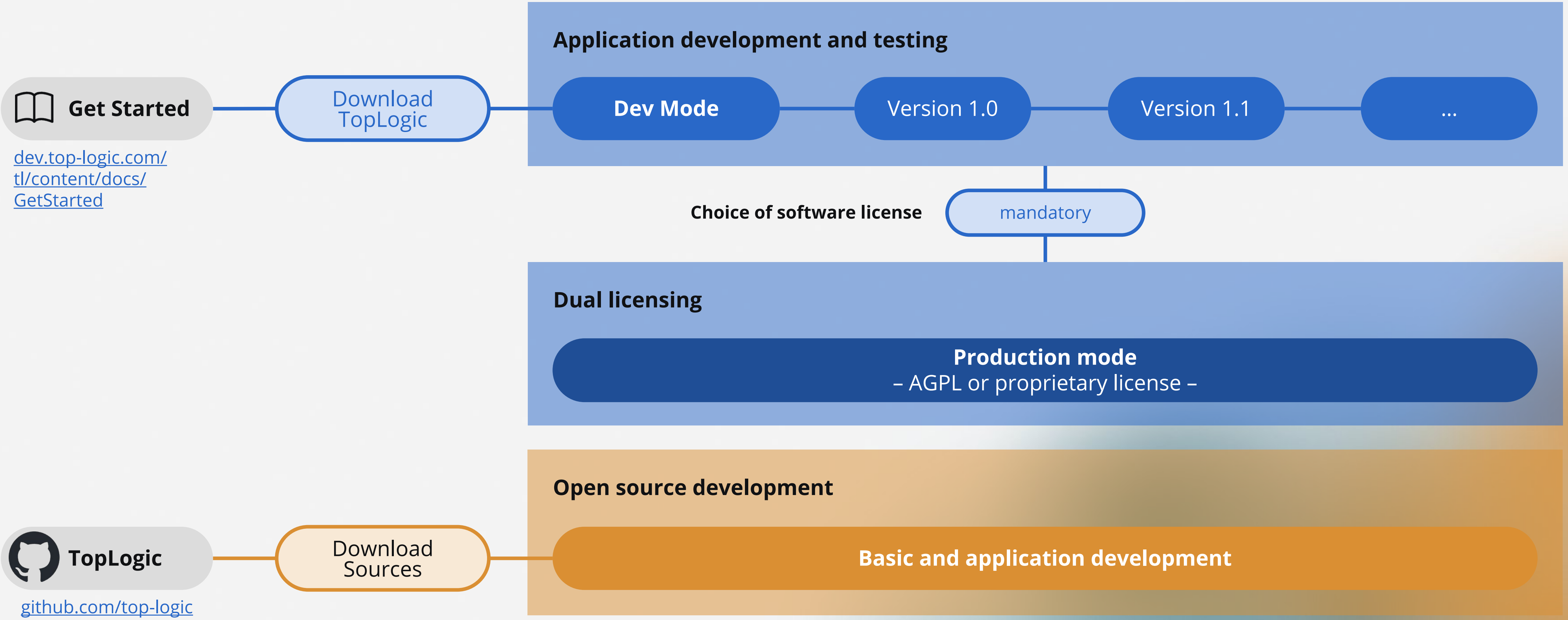
MODEL-CENTRIC NO CODE PLATFORM

WHO WE ARE

- Founded in 2002 with a clear vision: create a framework for building web applications.
- Framework evolved step-by-step and was applied in real customer projects from the start
- In 2023, we transitioned the product to Open Source with a unique Dual Licensing Model developed in collaboration with OSADL

DUAL LICENSING MODEL

Developed in collaboration with OSADL



Get Started

dev.top-logic.com/tl/content/docs/GetStarted

Download TopLogic

Application development and testing

Dev Mode

Version 1.0

Version 1.1

...

Choice of software license

mandatory

Dual licensing

Production mode
– AGPL or proprietary license –

TopLogic

github.com/top-logic

Download Sources

Open source development

Basic and application development

WHY THIS DUAL LICENSING MODEL

For Customers

- More design freedom: Open source or proprietary usage
- Reduced dependence on OEM manufacturers
- Cost reduction

For Company

- More business opportunities
- Growing need for Platform specialists

For Ecosystems

- Community building
- Increase in user numbers

OVERVIEW

Full Code vs. Low Code vs No Code

Full Code

- Everything is programmed manually
- Maximum flexibility, highest complexity
- High dependency on skilled developers

Low Code

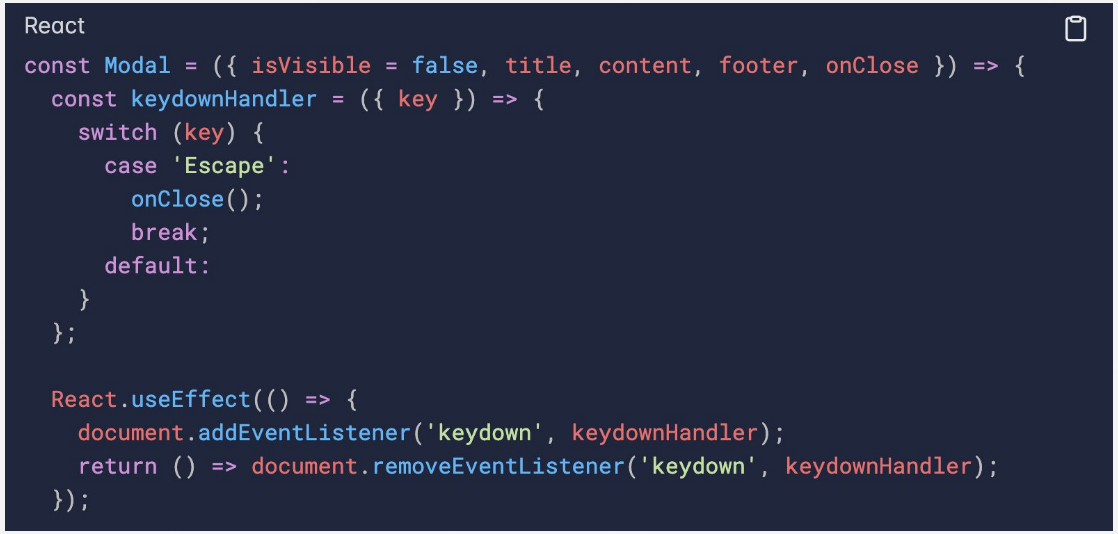
- Visual modeling + code extension where needed
- Faster development
- Balance between automation and flexibility

No Code

- Purely visual configuration
- No programming required
- Ideal for Requirements Engineers, Process & Digital Consultants

IMPERATIVE VS. DECLARATIVE

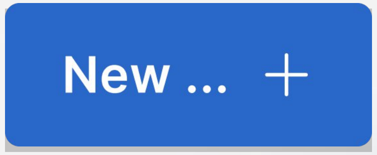
@Code: do the following …	@LCNC: I wish for …



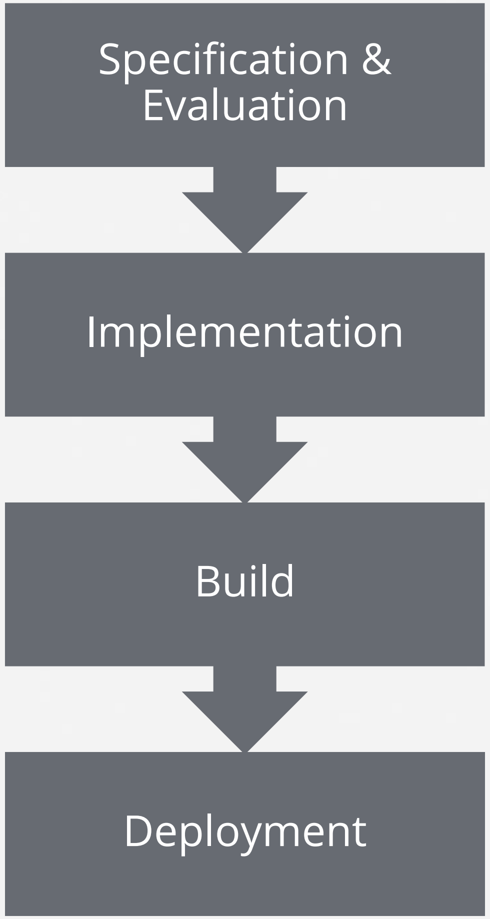
React

```
const Modal = ({ isVisible = false, title, content, footer, onClose }) => {
  const keydownHandler = ({ key }) => {
    switch (key) {
      case 'Escape':
        onClose();
        break;
      default:
    }
  };

  React.useEffect(() => {
    document.addEventListener('keydown', keydownHandler);
    return () => document.removeEventListener('keydown', keydownHandler);
  });
```



PARADIGM SHIFT



Specification & Evaluation

Implementation

Build

Deployment

Conventional development	Opportunities with LCNC
Implementation	
– Manual coding in frontend & backend – Longer implementation cycles – High dependency on specialized developers	– Standard Functions with Visual Configuration – Lower maintenance effort – Can be build by non-developers
Persistence	
– Risk of inconsistencies between logic, code and database – Refactoring data structures	– Automatic schema generation – Automatic data migration

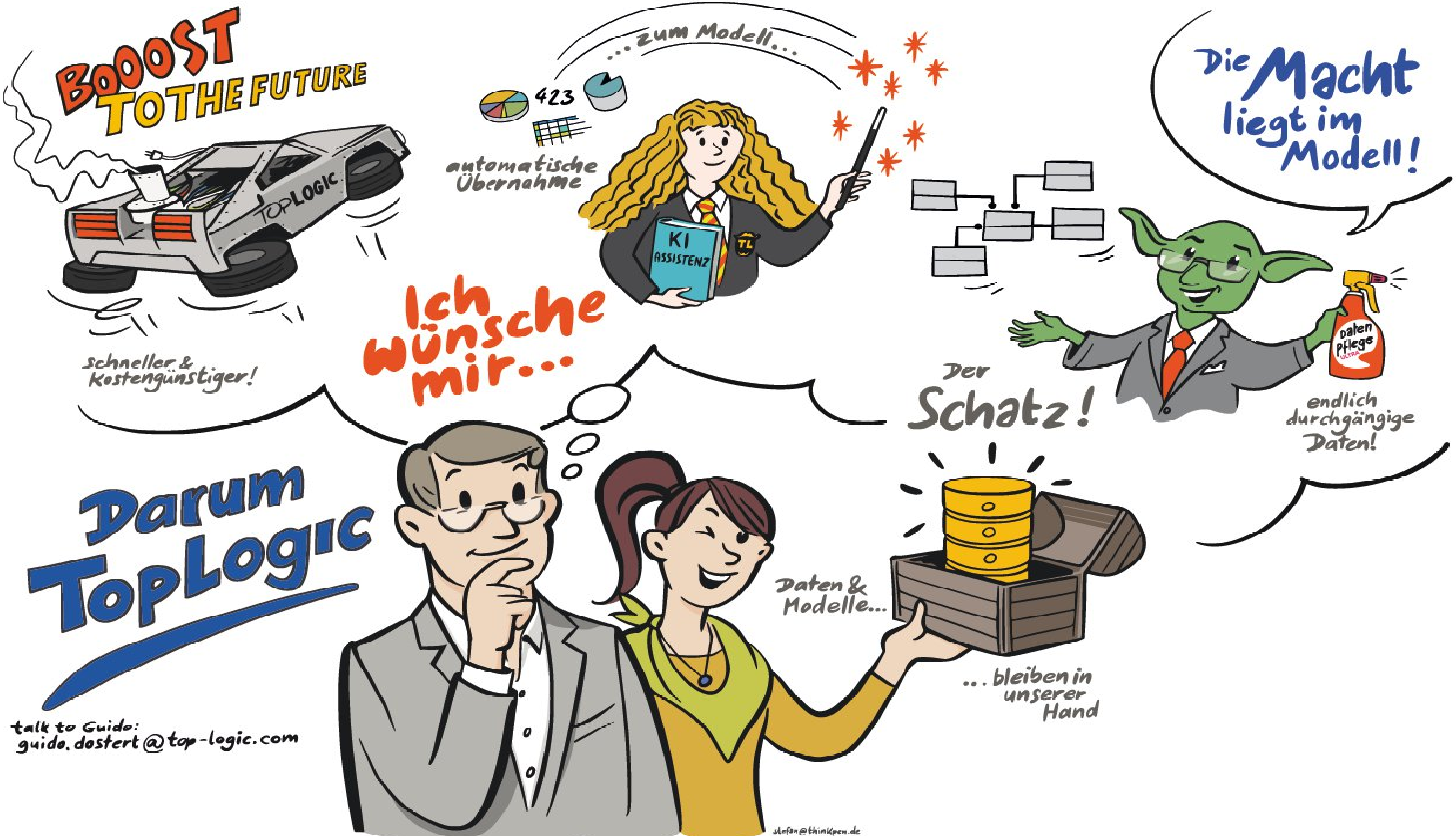
REASONS & CHALLENGES

For LCNC Platforms

- Faster deployment, reduced time to market
- Lower costs: reduction in development and maintenance costs
- Empowerment of domain specialists
- Lower dependency on scarce specialists
- Better quality through reuse of standardized components

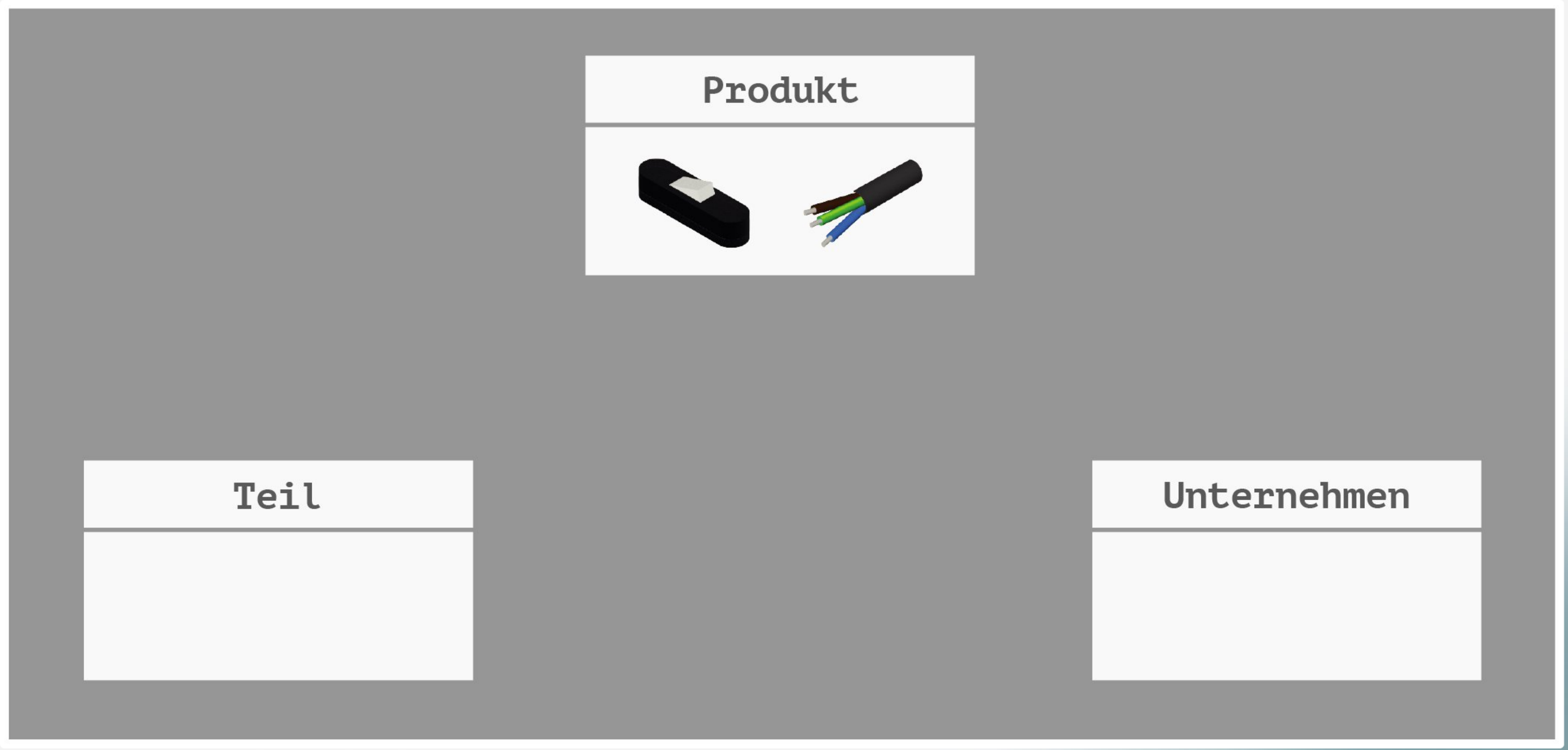
Challenges

- Requires a strong underlying platform
- Mindset shift from coding to modeling
- Integration with existing legacy systems


BOOOST TO THE FUTURE
TOPLOGIC
schneller & Kostengünstiger!
automatische Übernahme
423
...zum Modell...
KI ASSISTENZ
TL
Die Macht liegt im Modell!
Daten Pflege ULTRA
endlich durchgängige Daten!
Ich wünsche mir...
Der Schatz!
Daten & Modelle...
...bleiben in unserer Hand
Darum TopLogic
talk to Guido:
guido.dostert@top-logic.com
stefan@thinkpen.de

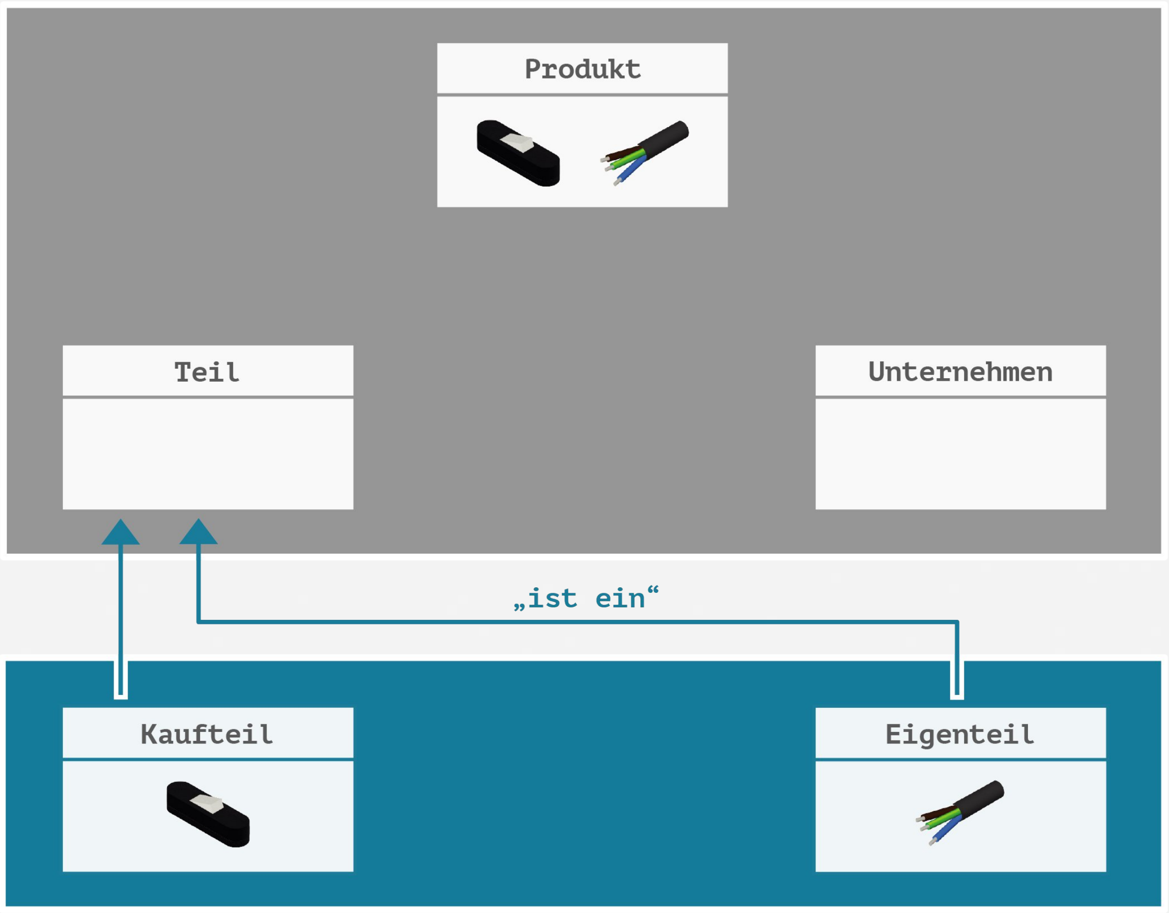
BASIS: OBJECT-ORIENTED MODELING

(1) Identification



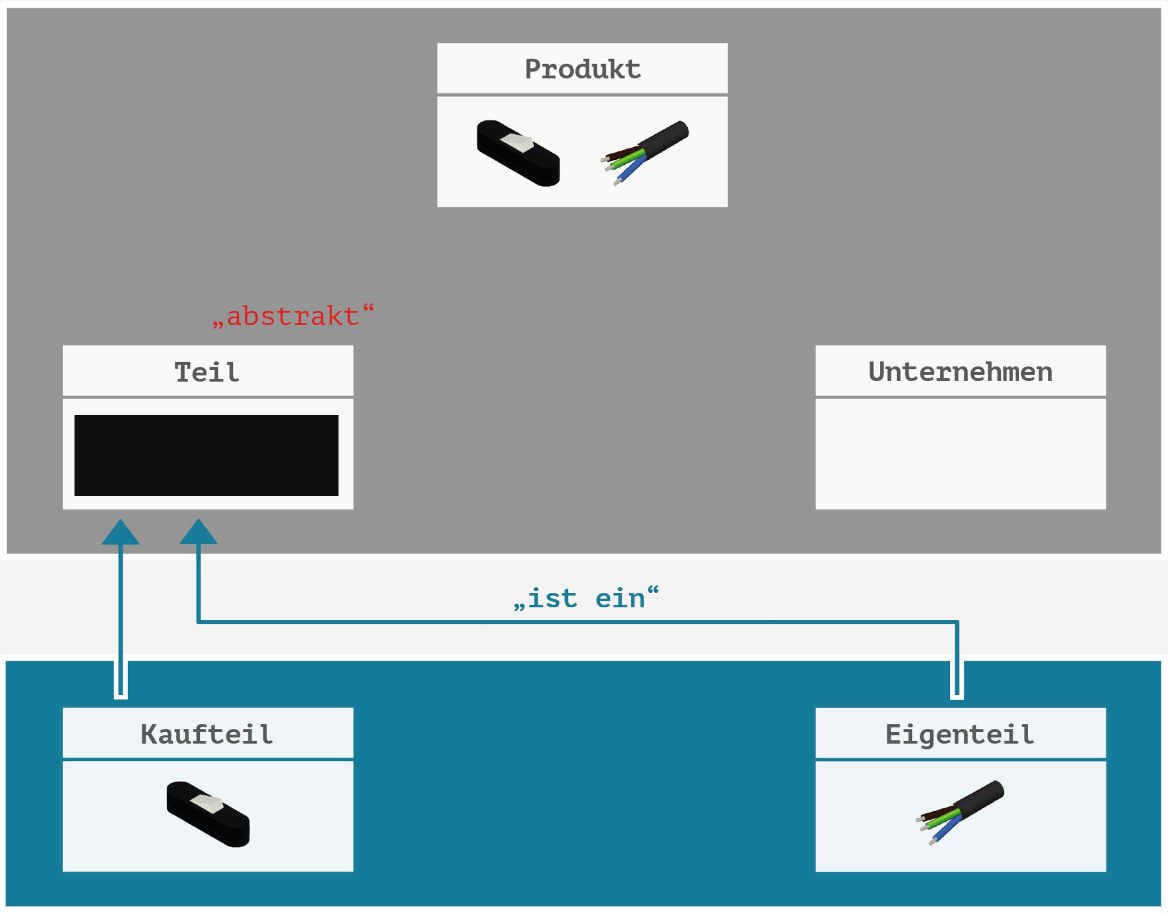
BASIS: OBJECT-ORIENTED MODELING

(2) Generalization



BASIS: OBJECT-ORIENTED MODELING

(3) Abstract classes



Produkt

„abstrakt“

Teil

Unternehmen

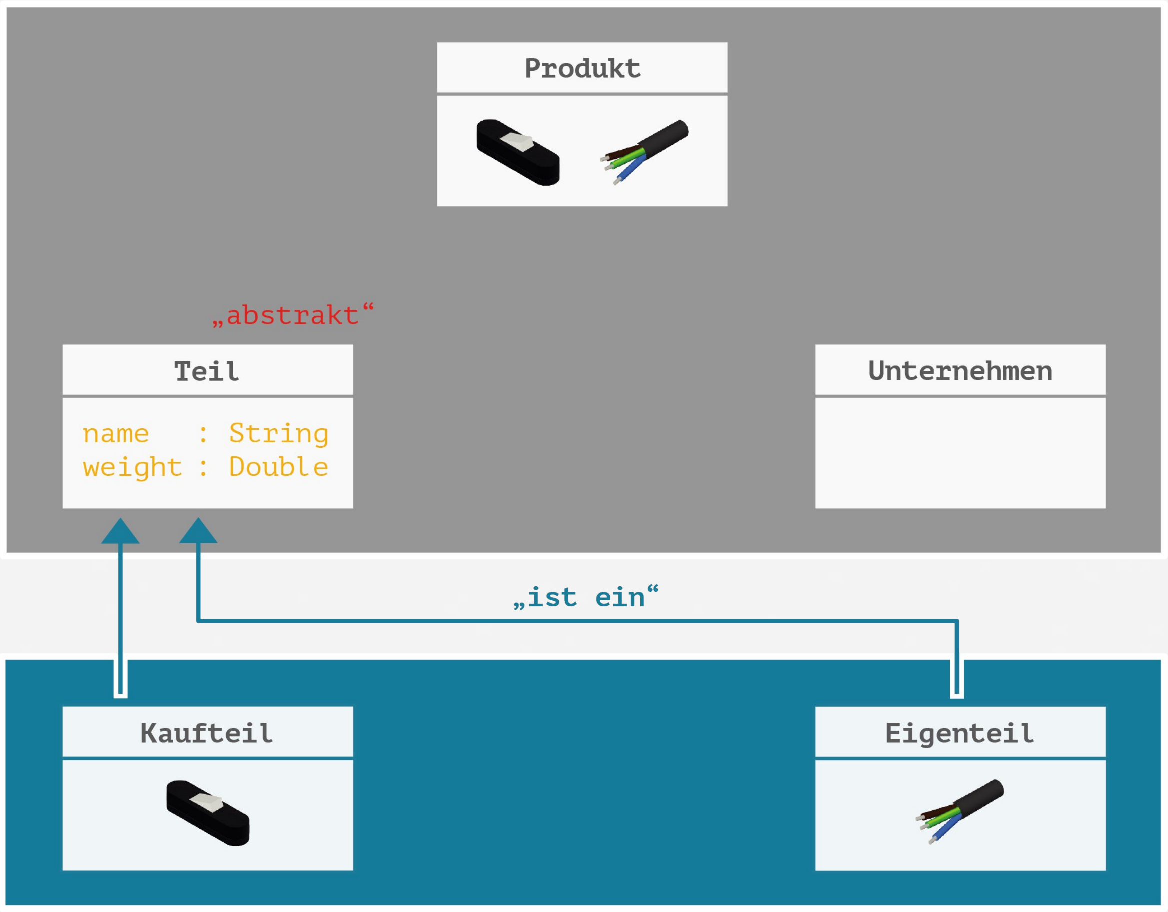
„ist ein“

Kaufteil

Eigenteil

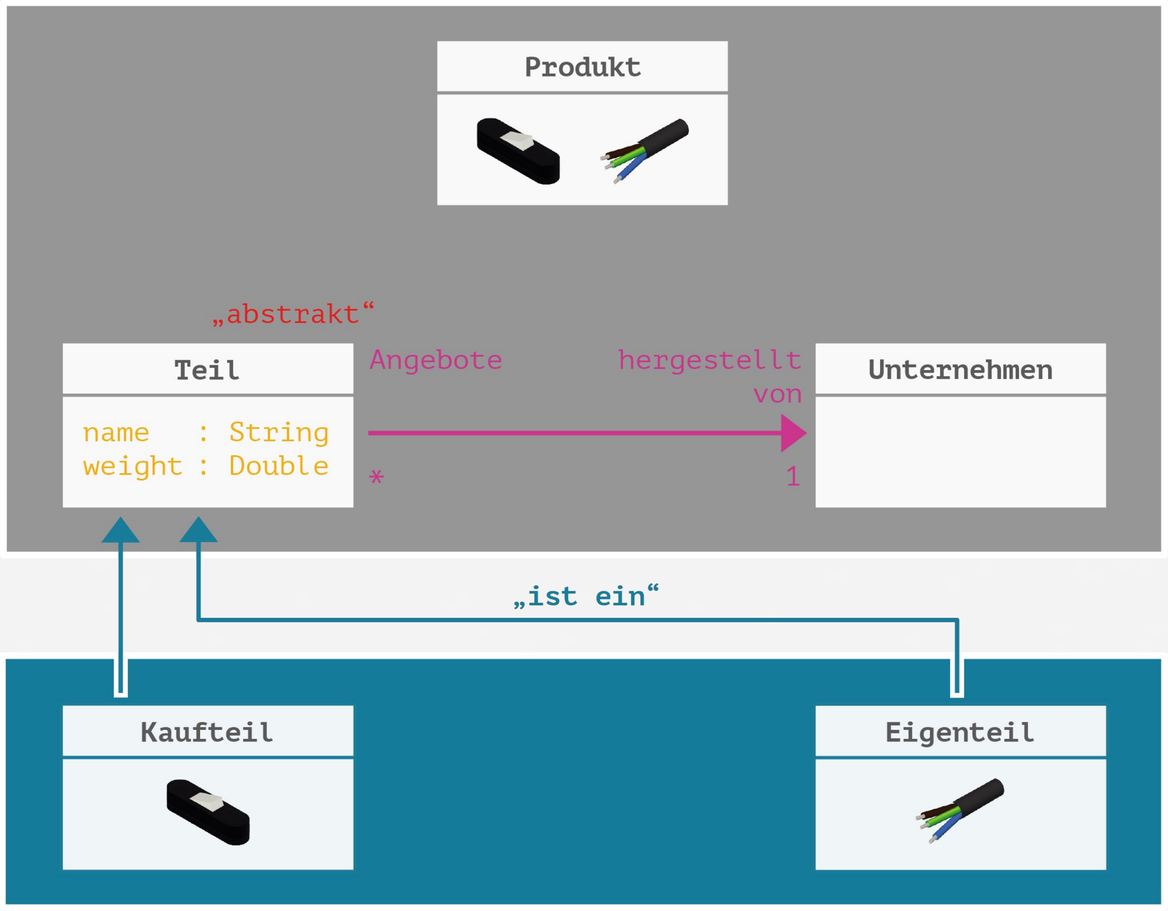
BASIS: OBJECT-ORIENTED MODELING

(4) Properties of data types



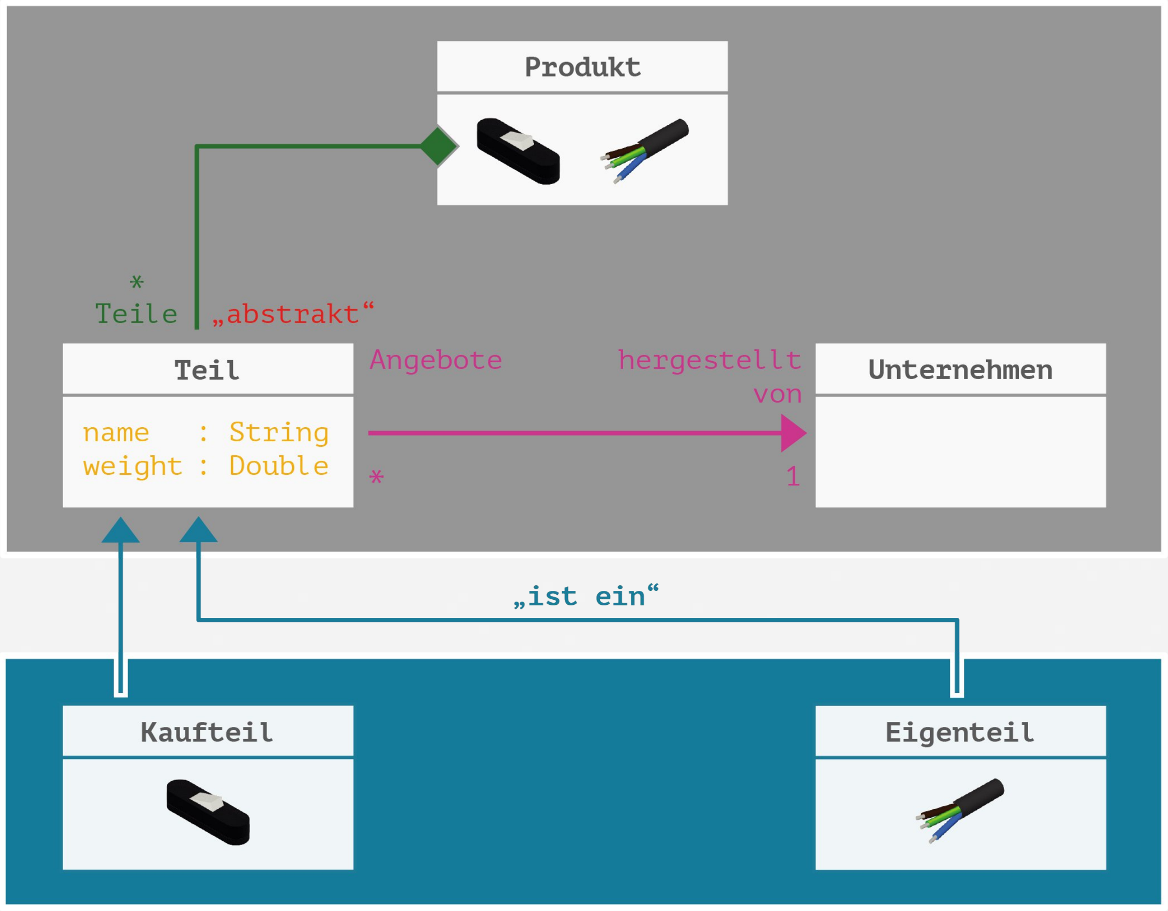
BASIS: OBJECT-ORIENTED MODELING

(5) Associations



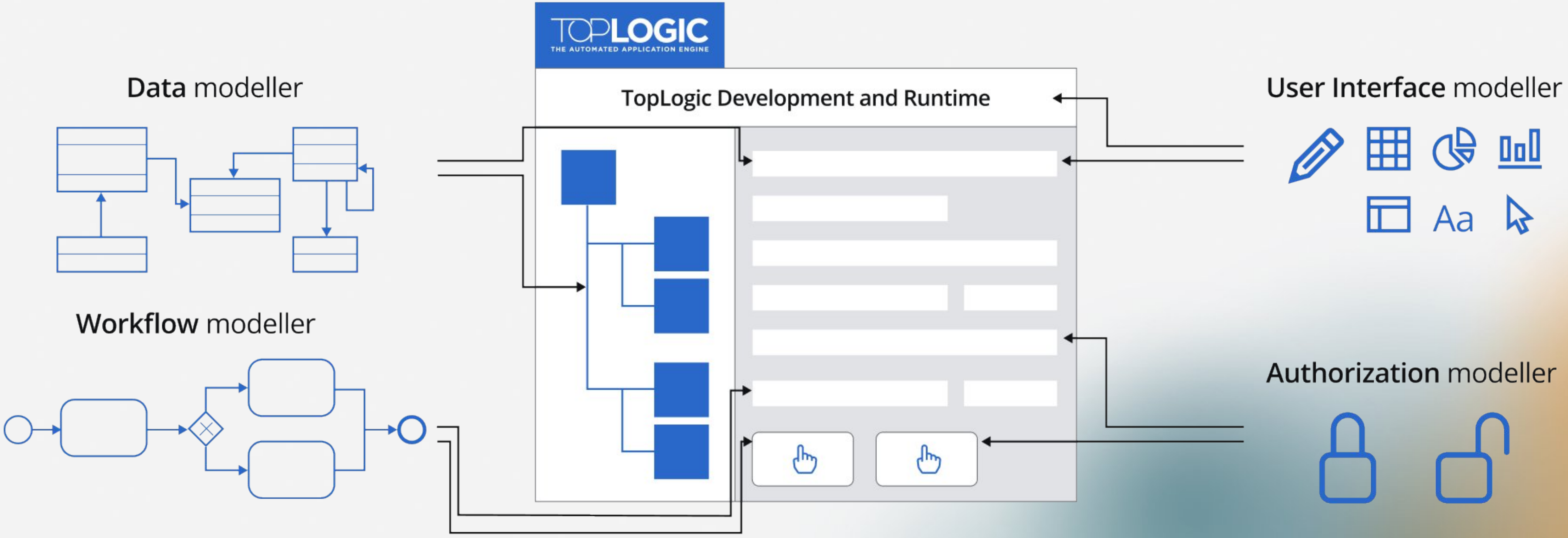
BASIS: OBJECT-ORIENTED MODELING

(5) Composition



TOPLOGIC THE AUTOMATED APPLICATION ENGINE @ COOL COMPACT OSADL ONLINE LECTURES

IN-APP DEVELOPMENT IN 4 STEPS



USE CASE

- Pizza shop wants to introduce a new pizza: “Diavolo”
- **Today**: Excel sheets for ingredients, costs & budgeting
- Every new pizza = new spreadsheet chaos
- They want a simple internal app to:
  - define ingredients
  - calculate costs automatically
  - compare costs vs. budget
- **Goal**: Build a small, reusable TopLogic app

USE CASE IMPLEMENTATION

RECAP

- Dual Licensing Model → flexibility & ecosystem growth
- Why LCNC → faster, cheaper, less dependency on specialists
- Key opportunities & challenges
- Model-centric approach → everything driven by the model
- Use Case: Pizza App → from Excel chaos to a structured TL app

WHERE TO LEARN MORE

Freely accessible documentation

- top-logic.com, Company presentation | ***de/en***
- opensource.top-logic.com: Landing page for productization, “TopLogic goes OpenSource”| ***de/en***
- github.com/top-logic: Repository for software development with TopLogic (open source and proprietary) | ***en***
- dev.top-logic.com/tl/content/releases: Release information | ***de/en***
- dev.top-logic.com/tl/content/docs: Manuals for using TopLogic | ***de/en***
  - Get Started
  - Tutorial
  - Developer's Guide
  - User guide
  - Administrative manual
- github.com/top-logic/tl-engine: Complete source code | ***en***