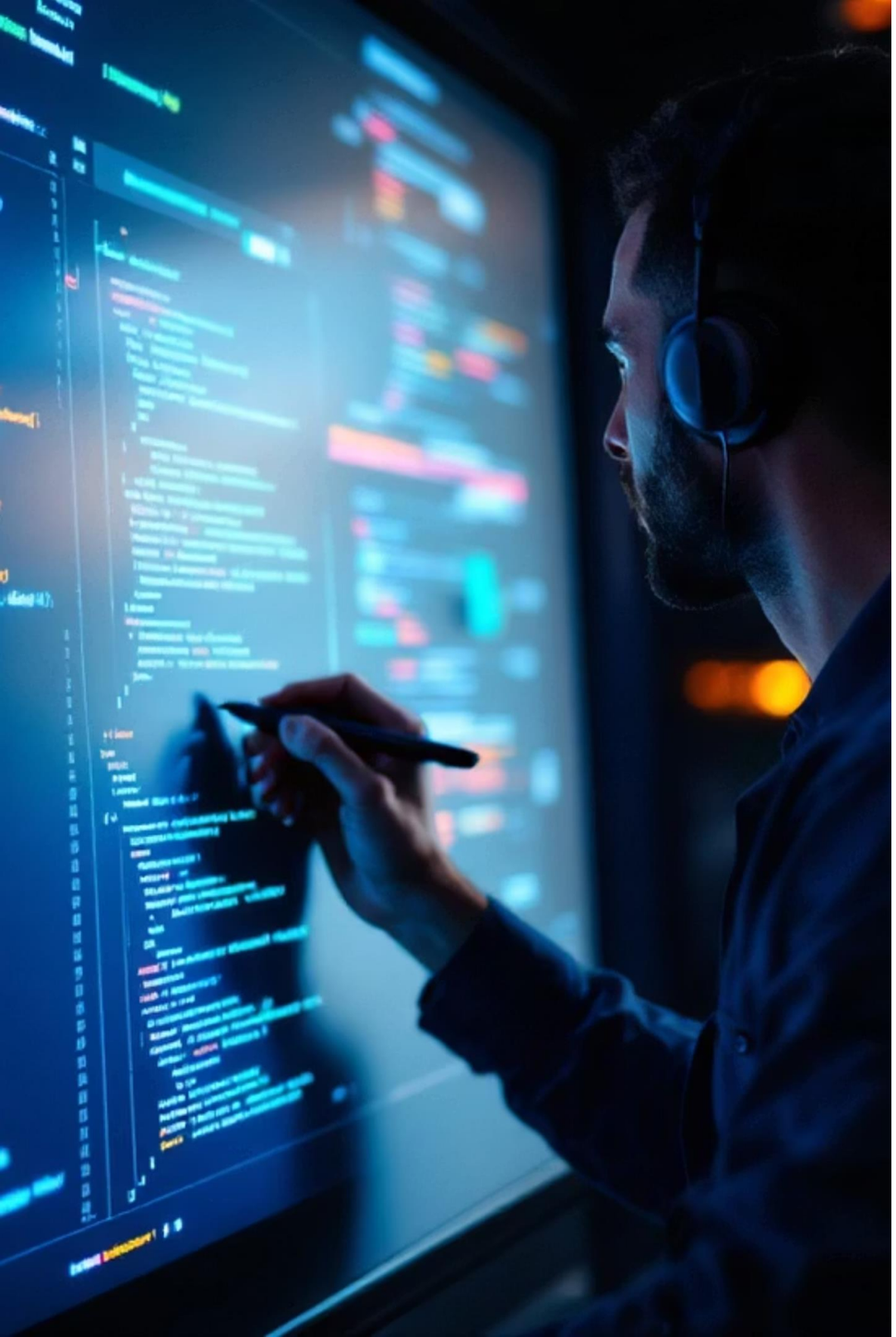


Grundlagen der Programmierlogik

Willkommen zu dieser Einführung in die Programmierlogik - dem Fundament jeder erfolgreichen Softwareentwicklung. In den kommenden Folien lernen Sie, wie strukturiertes Denken und methodisches Vorgehen die Basis für effiziente Programmierung bilden.

Der Bauplan als Grundlage

Ein unerfahrener Bauherr beginnt vielleicht damit, wahllos Ziegel aufeinander zu schichten. Ohne Fundament, ohne Struktur, ohne klaren Zweck. Das Ergebnis wird instabil, schief und am Ende unbrauchbar – eine Verschwendung von Zeit, Material und Energie. Jedes Problem, das während des Baus auftaucht, ist eine Ad-hoc-Lösung, die oft neue Probleme schafft.

Ein professioneller Architekt hingegen zeichnet zuerst einen detaillierten **Bauplan**. Dieser Plan ist das Ergebnis gründlicher Überlegung: Er berücksichtigt die Statik, die optimale Raumaufteilung, die effiziente Führung von Leitungen und die ästhetischen Aspekte. All dies geschieht lange, bevor der erste Bagger rollt oder auch nur ein einziger Ziegelstein platziert wird. Der Bauplan ist die Voraussicht, die das spätere Scheitern verhindert.

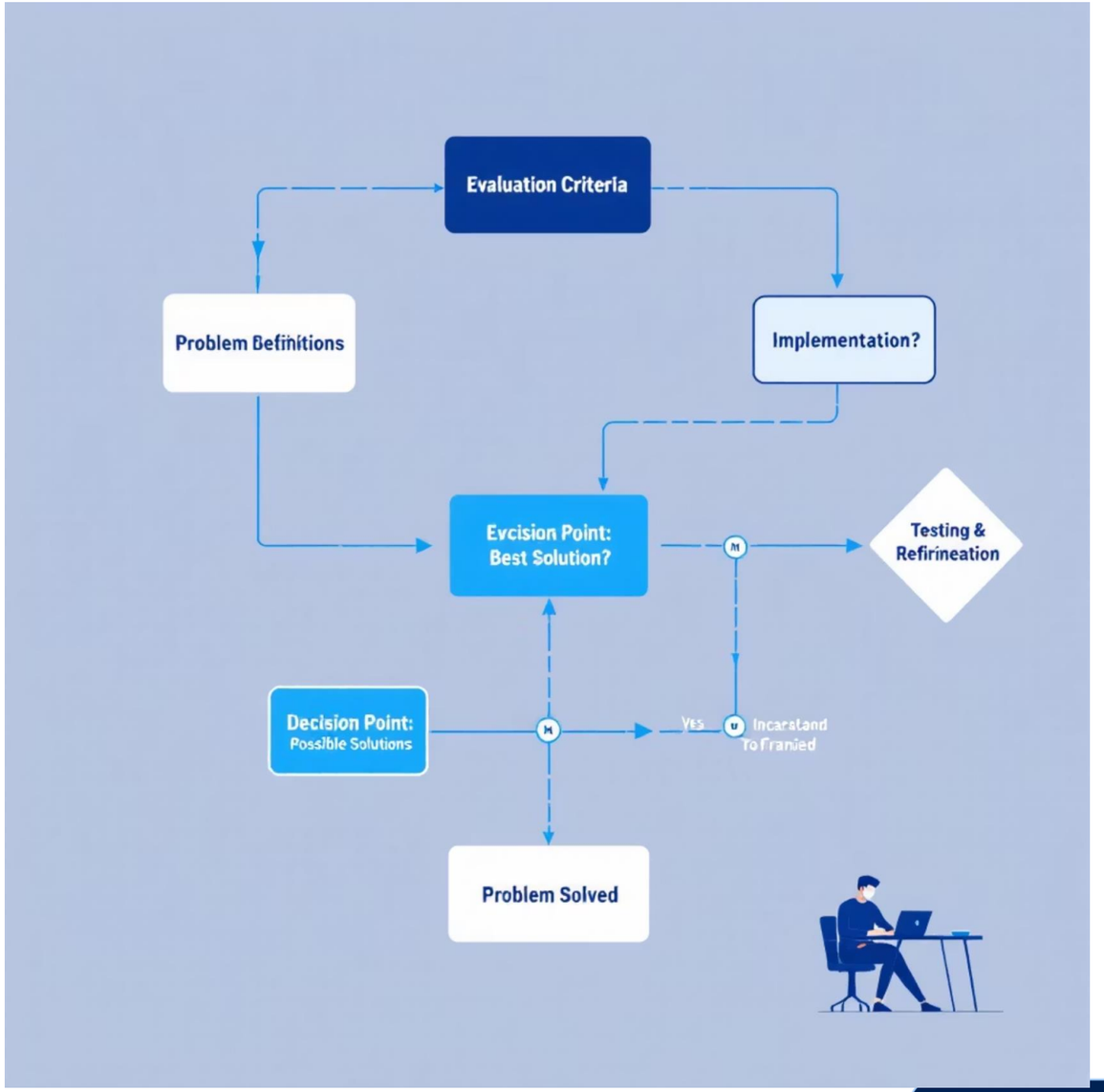
In der Softwareentwicklung ist die **Programmierlogik** unser Bauplan. Sie ist die Kunst, ein Problem vollständig zu durchdenken und eine strukturierte Lösungsstrategie zu entwickeln, *bevor* wir an eine Programmiersprache überhaupt nur denken. Es geht darum, die einzelnen Schritte von der Eingabe bis zur gewünschten Ausgabe zu definieren, Abhängigkeiten zu identifizieren und mögliche Fehlerquellen vorauszusehen.

Was ist Programmierlogik?

In der Softwareentwicklung ist die **Programmierlogik** unser Bauplan. Bevor wir eine einzige Zeile Code schreiben, müssen wir das Problem vollständig verstehen und den Lösungsweg strukturieren.

- Sie definiert, WAS zu tun ist
- Sie legt die Reihenfolge der Schritte fest
- Sie identifiziert notwendige Entscheidungen
- Sie erkennt Wiederholungsmuster

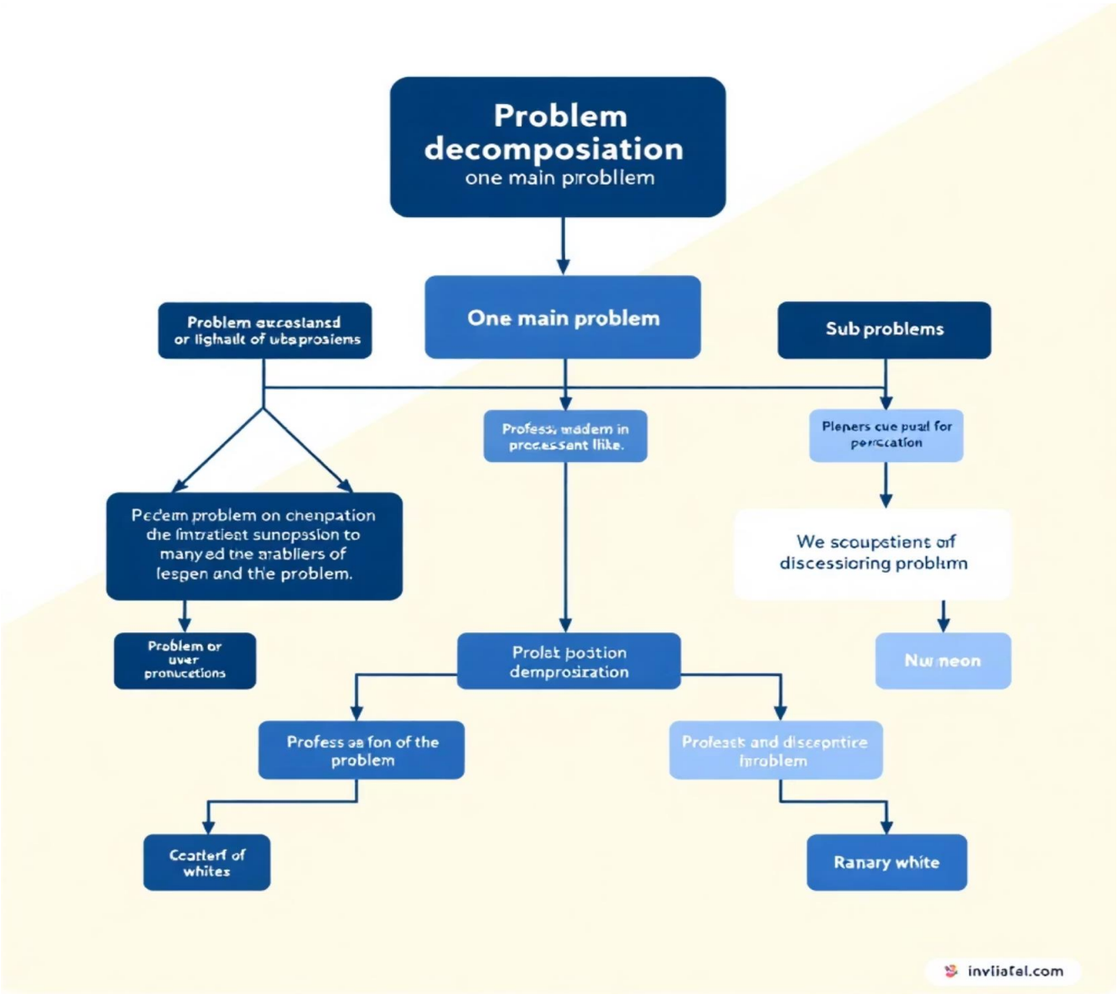
Eine gute Programmierlogik ist unabhängig von der konkreten Programmiersprache und kann später in jede beliebige Sprache übersetzt werden.



Top-Down-Design: Vom Ganzen ins Detail

Was ist Top-Down-Design?

- Wichtigstes Prinzip der Programmierlogik (auch "schrittweise Verfeinerung").
- Methode zur Bewältigung komplexer Probleme.
- Systematisches Zerlegen großer, unübersichtlicher Probleme in:
  - Immer kleinere, greifbare Teilprobleme.
  - Bis jedes Teilproblem einfach lösbar oder als einzelne Code-Funktion beschreibbar ist.



Beispiel für Top-Down-Design

1

Großes Problem: "Erstelle einen Online-Shop"

Das übergeordnete Ziel, das in kleinere Teilprobleme zerlegt werden muss

2

Teilproblem 1: "Benutzerverwaltung"

- Unter-Teilproblem 1.1: "Login-Funktion erstellen"
- Unter-Teilproblem 1.2: "Registrierungs-Funktion erstellen"

3

Teilproblem 2: "Produktkatalog"

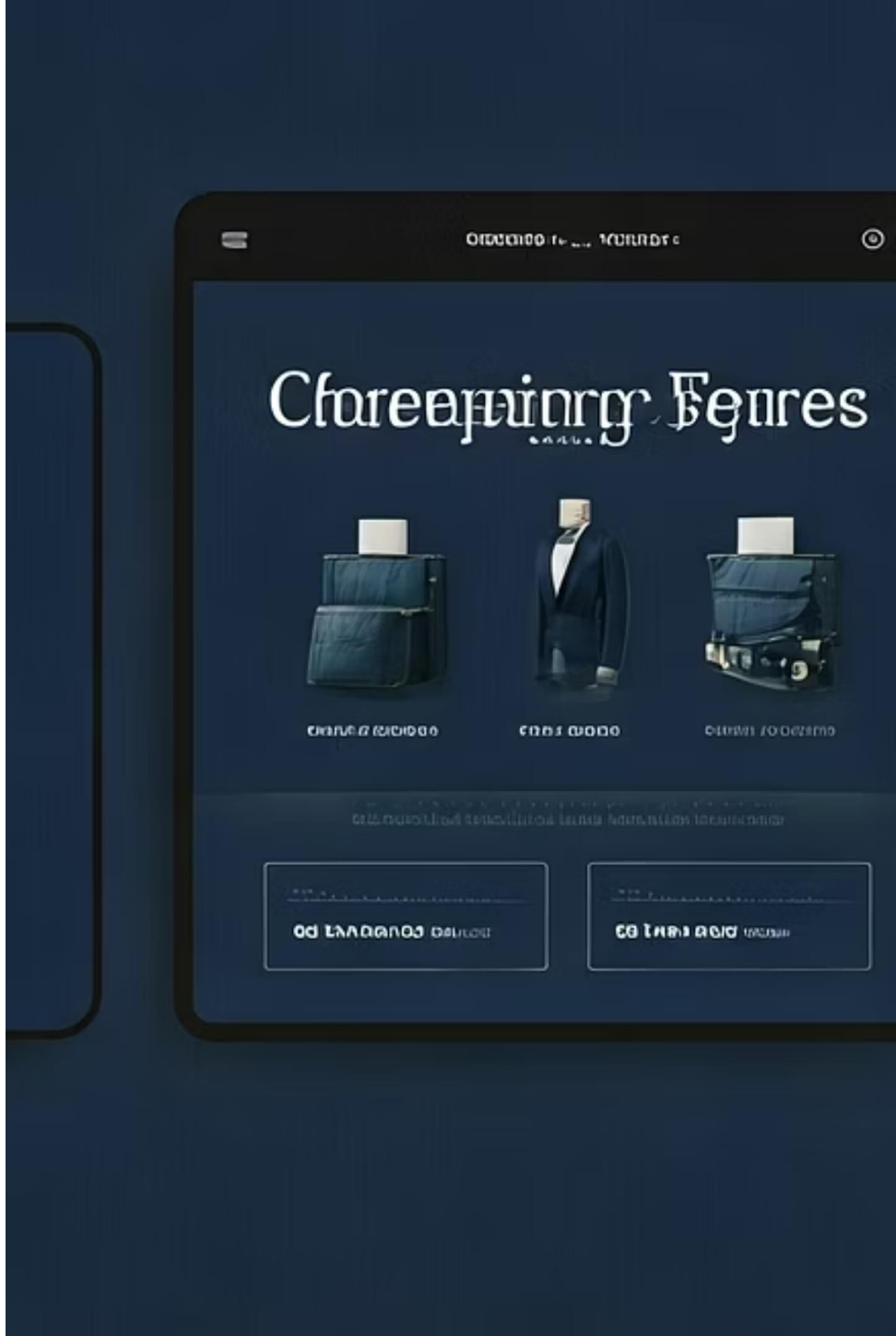
Auflistung und Darstellung aller verfügbaren Produkte im Shop

4

Teilproblem 3: "Warenkorb-Funktion"

Verwaltung der ausgewählten Produkte vor dem Kauf

Dieses Vorgehen stellt sicher, dass wir nicht den Überblick verlieren und systematisch arbeiten.



Die Werkzeuge des Architekten

1

Sequenz

Eine feste Reihenfolge von Anweisungen.

Beispiel: Zuerst den Salat waschen, DANN schneiden, DANN das Dressing machen.

2

Selektion (Verzweigung)

Eine Entscheidung, die den Weg teilt.

Beispiel: WENN der Gast Vegetarier ist, DANN biete die Gemüse-Spieße an, SONST das Steak.

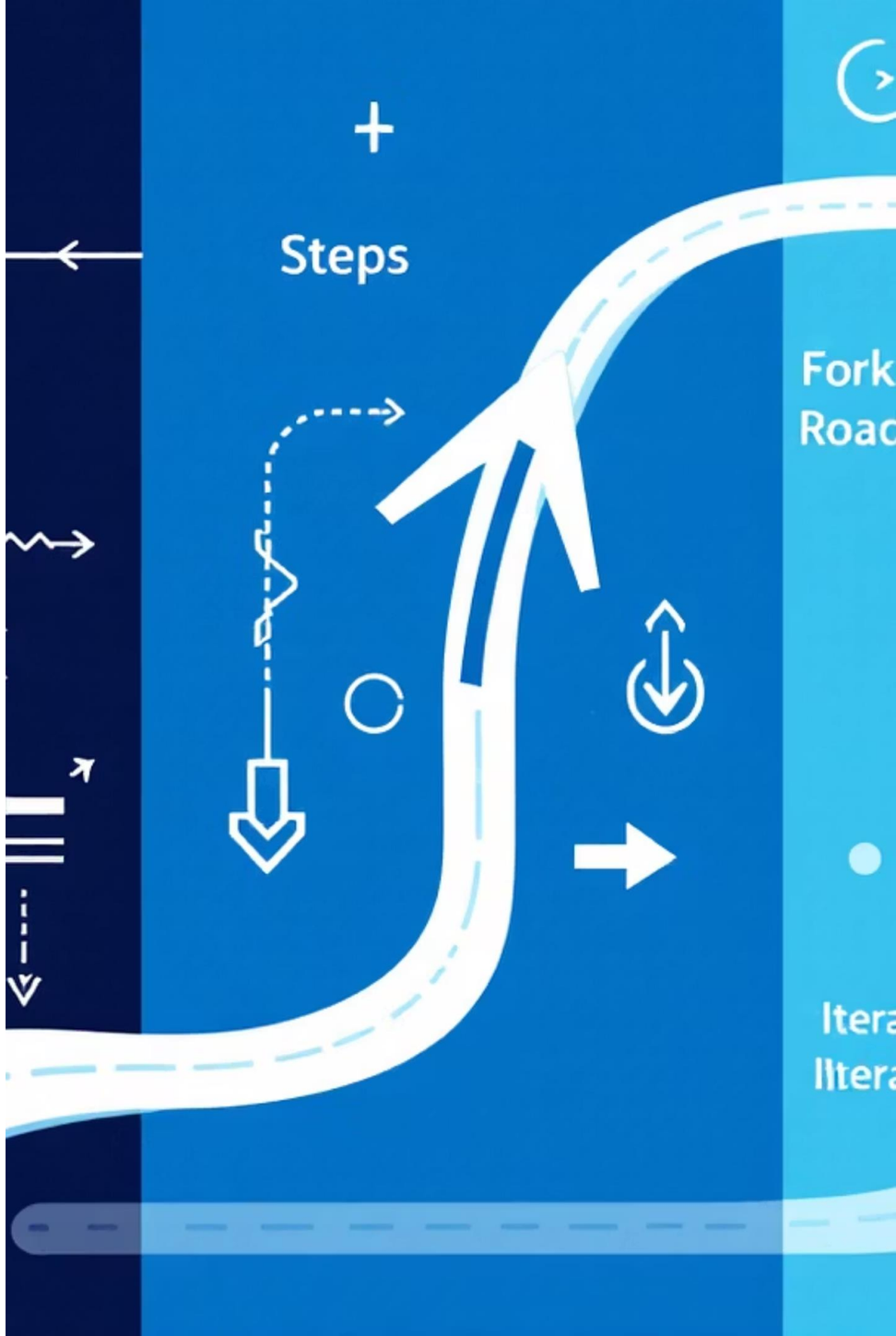
3

Iteration (Schleife)

Eine sich wiederholende Aufgabe.

Beispiel: FÜR jeden Gast auf der Einladungsliste, MACHE folgendes: 'Schreibe eine Einladungskarte'.

Wenn man diese drei Strukturen beherrscht, kann man theoretisch jedes lösbare Problem der Welt programmieren.



Abstraktion: Das Prinzip der "Schwarzen Kiste"

Abstraktion ist die direkte Folge von gutem Top-Down-Design. Es bedeutet, die inneren Details eines gelösten Teilproblems zu ignorieren und es wie eine "Schwarze Kiste" (Black Box) zu behandeln.

- Wir wissen, *was* die Kiste tut
- Es interessiert uns nicht mehr, *wie* sie es tut
- Wir vertrauen darauf, dass sie funktioniert

In der Programmierung ist das eine Funktion oder eine Klasse.

Praxis-Demo: Kaffee kochen

Maschine vorbereiten

1. Kanne mit Wasser füllen
2. Wasser in den Tank gießen
3. Filtertüte einlegen
4. Kaffeepulver einfüllen
5. Kanne auf Heizplatte stellen

Brühvorgang starten

1. Maschine einschalten
2. Warten, bis Wasser durchgelaufen ist
3. Maschine ausschalten

Kaffee entnehmen

Die fertige Tasse Kaffee genießen

Wir haben ein komplexes Problem erfolgreich in eine logische **Sequenz** von einfachen, unmissverständlichen Anweisungen zerlegt. Das ist die Essenz der Programmierlogik.