Refactoring mit der Mikado-Methode

FALK SIPPACH

IT-Tage 365

Donnerstag, 17.06.2021

embarc
Software Consulting GmbH

Refactoring mit der Mikado-Methode

Zusammenfassung

Viele von uns haben tagtäglich mit Legacy-Code zu tun. Mal eben schnell etwas umzubauen, scheitert typischerweise an den fehlenden Tests, zudem ist der Quellcode oft überhaupt schlecht testbar.

In diesem Vortrag wird anhand von praktischen Codebeispielen gezeigt, wie man zunächst ein automatisiertes Sicherheitsnetz aufspannt. Anschließend werden komplexere Refactorings durchgeführt, ohne jedoch zu viele Baustellen gleichzeitig aufzureißen. Die Mikado-Methode hilft dabei, den Überblick zu behalten und in möglichst kleinen und nachvollziehbaren Schritten vorzugehen. Das Ziel ist das Aufbrechen stark gekoppelter Abhängigkeiten, um so neue Tests hinzufügen zu können. Zudem wird der Code besser lesbar sein und lässt sich so auch leichter warten und wiederverwenden.

Falk Sippach

- Softwarearchitekt, Berater, Trainer bei embarc
- früher bei Orientation in Objects (OIO), Trivadis

Schwerpunkte:

- Architekturberatung und -bewertung
- Cloud- und Java-Technologien

fs@embarc.de

@sippsack

➔ xing.to/fsi

embarc Software Consulting GmbH

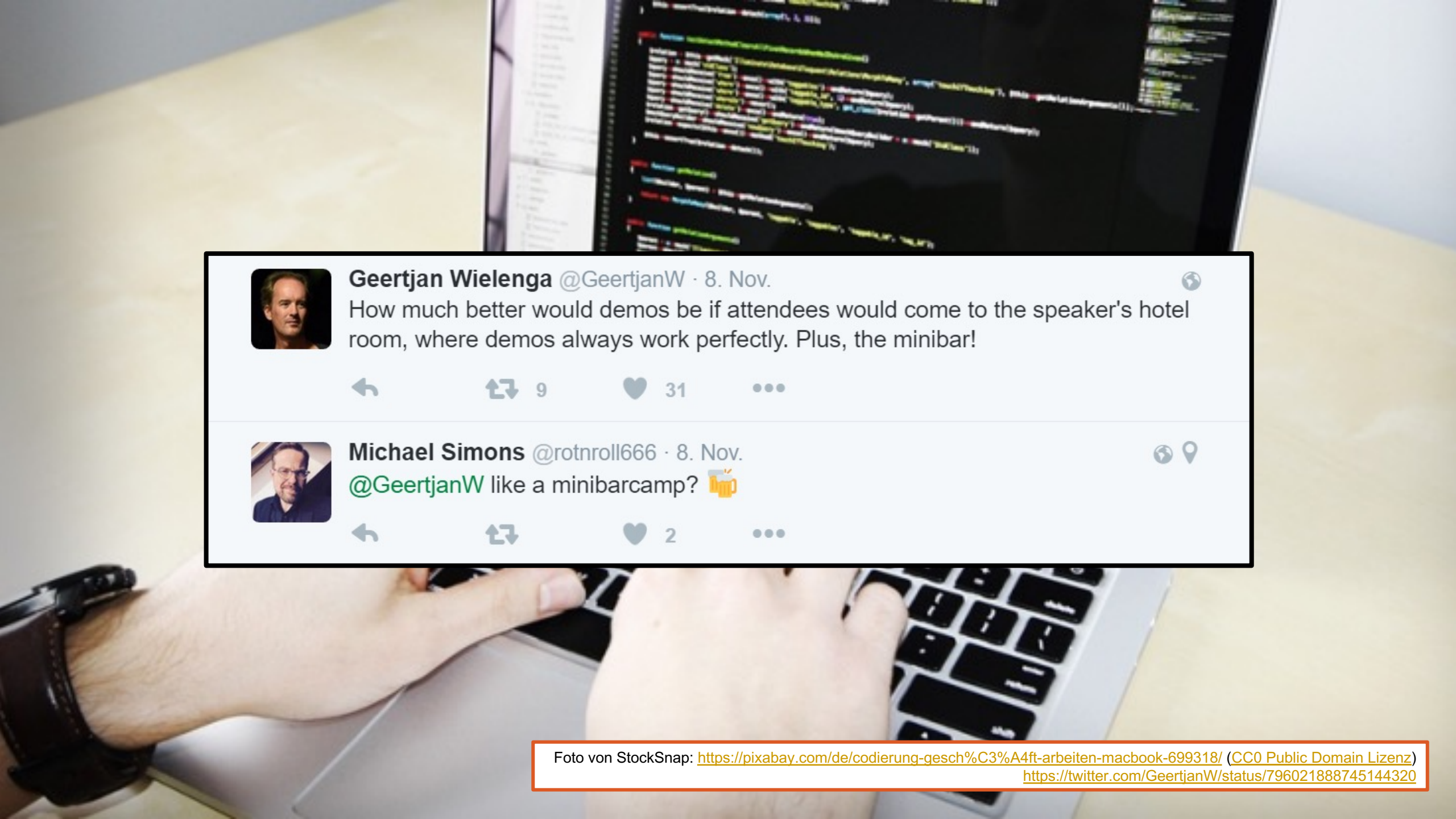


Foto von StockSnap: https://pixabay.com/de/codierung-gesch%C3%A4ft-arbeiten-macbook-699318/ (CC0 Public Domain Lizenz)
https://twitter.com/GeertjanW/status/796021888745144320

Agenda

1. Einstieg
2. Legacy Code
3. Mikado Methode
4. Reale Legacy Projekte
5. Ausblick und weitere Informationen

Agenda

1 **Einstieg**

2 Legacy Code

3 Mikado Methode

4 Reale Legacy Projekte

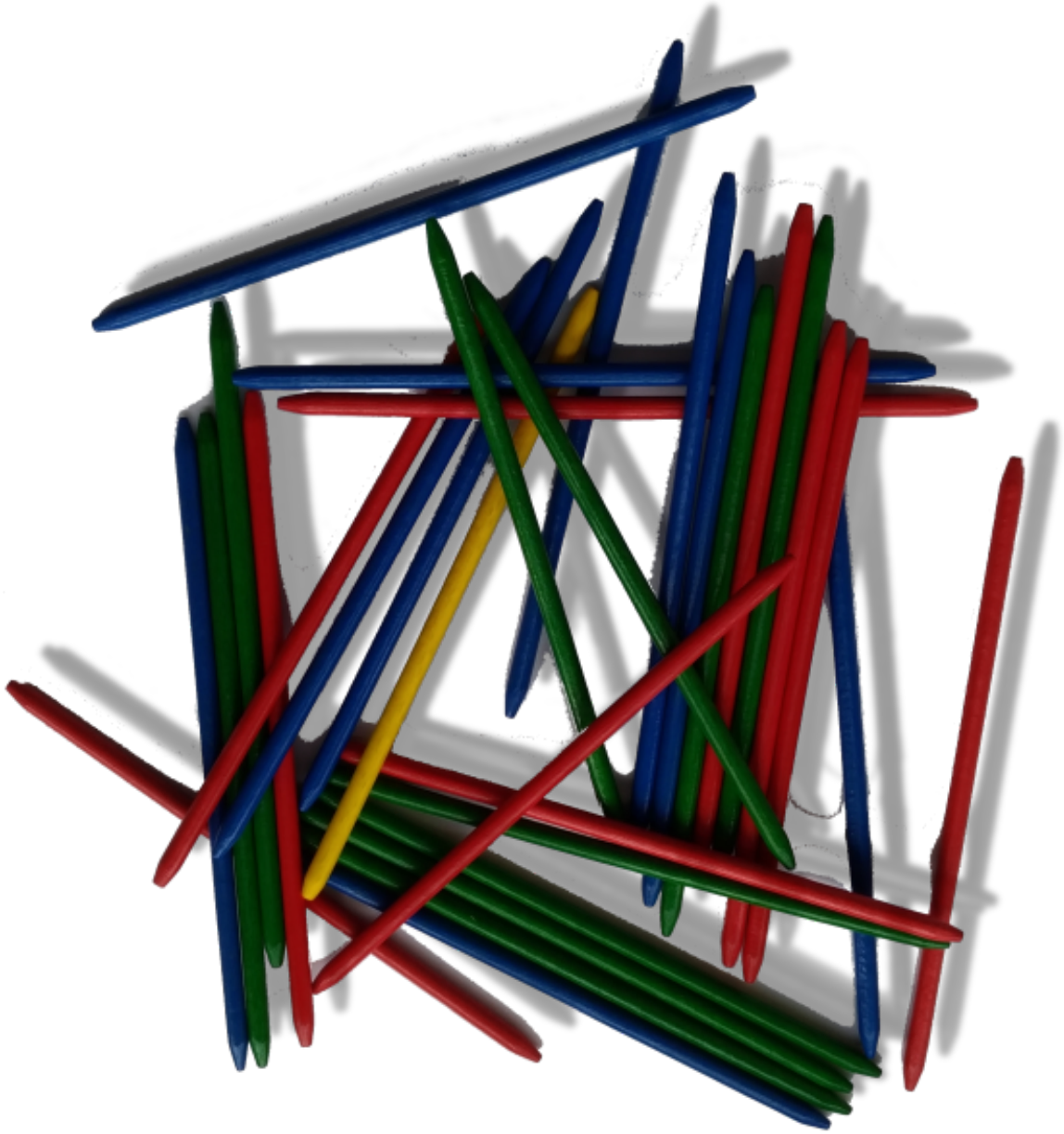
5 Ausblick und weitere Informationen

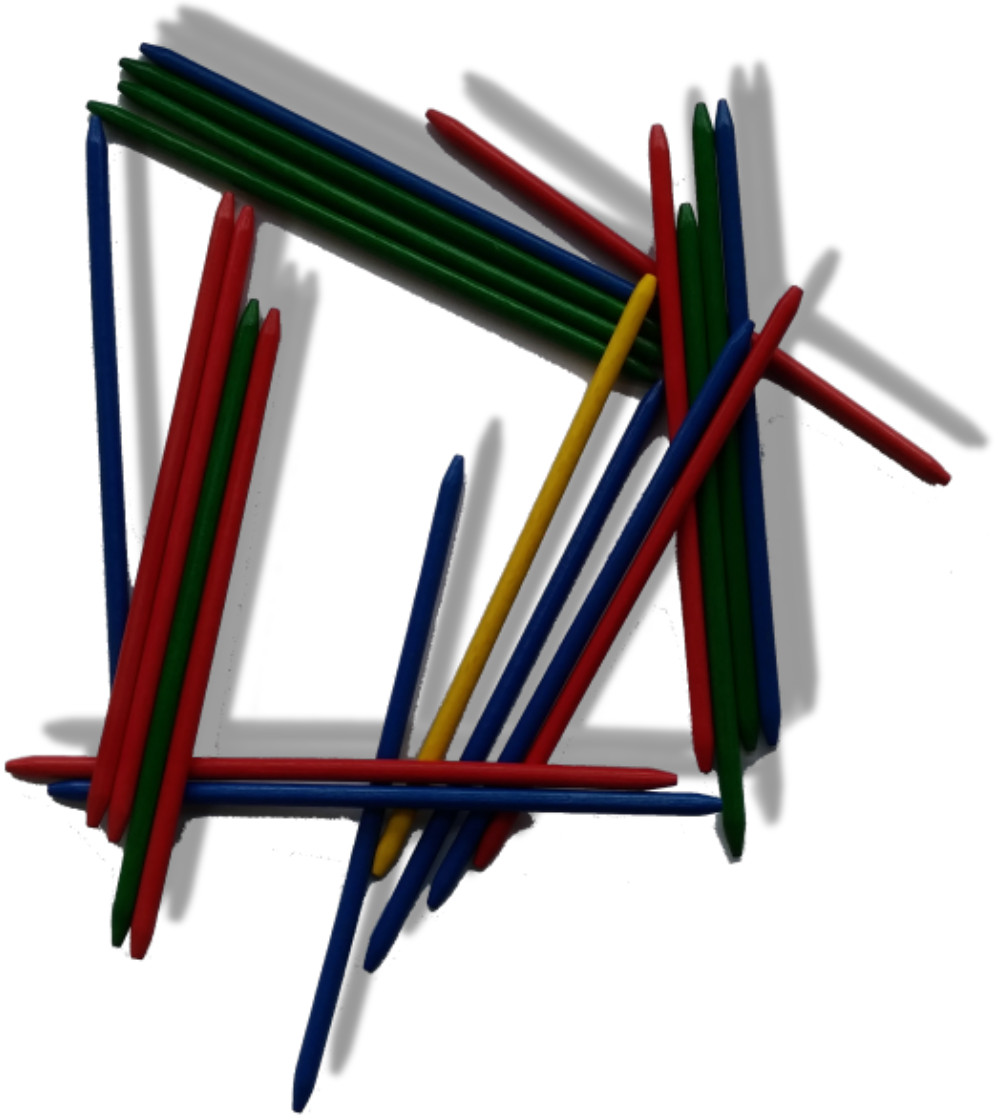
1

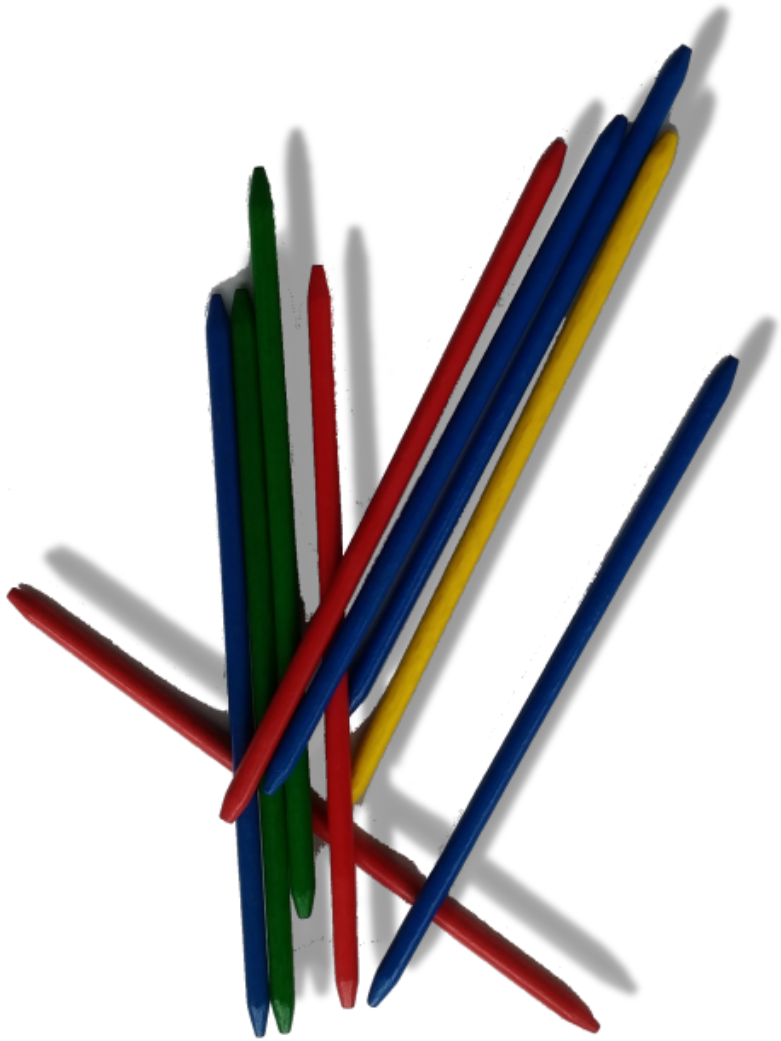
*The **Mikado Method** is a **structured way** to make **significant changes** to **complex code**.*

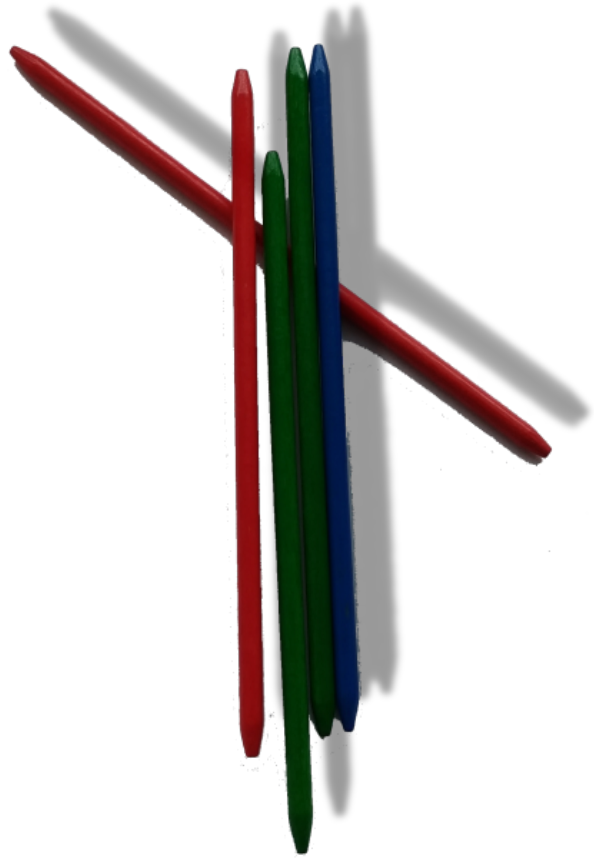
*… for **performing changes** to a **system that's too large for analyze-then-edit**, which means basically **any production system** in the world.*

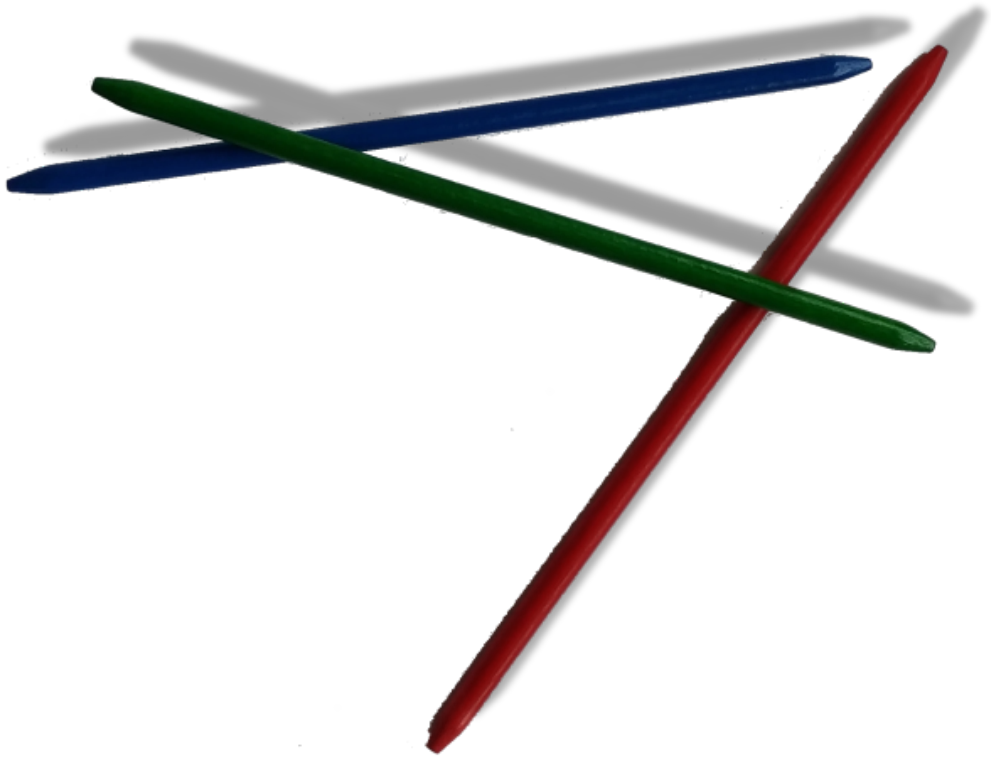
(Ola Ellnestam, Daniel Brolund: "The Mikado Method")











Agenda

1 Einstieg

2 **Legacy Code**

3 Mikado Methode

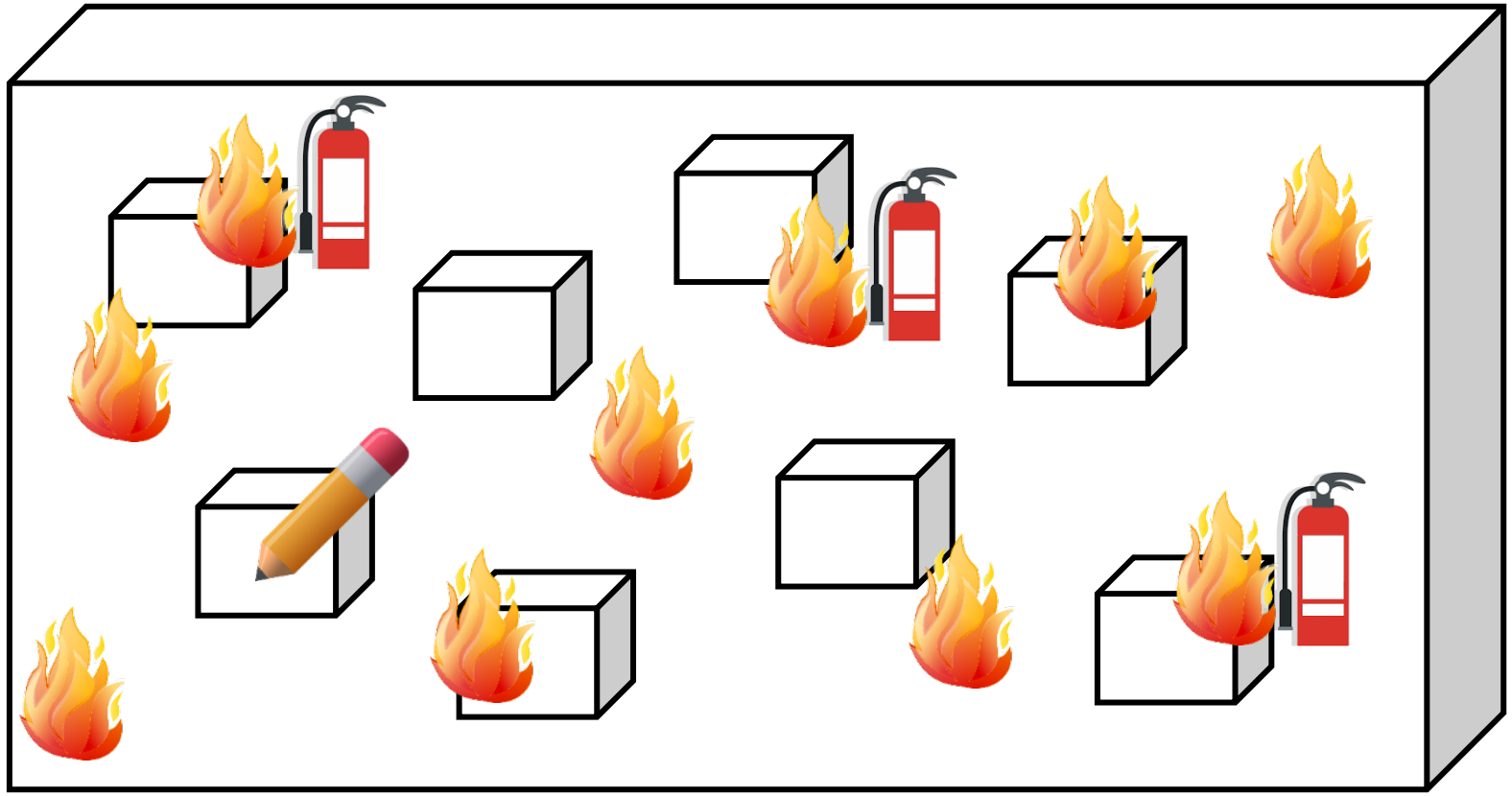
4 Reale Legacy Projekte

5 Ausblick und weitere Informationen

2

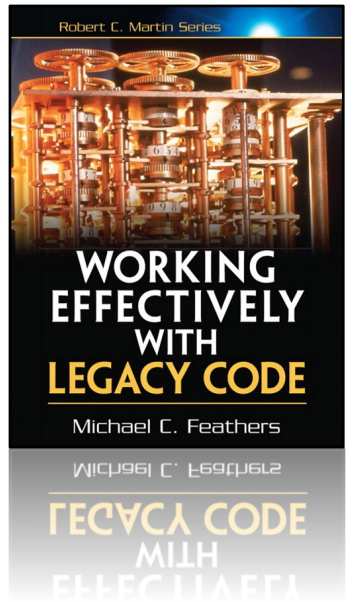
Code ändern? Nur wie?

***Brownfield**-Projekt. Code von **anderen**.*
***Fehlende Dokumentation**, kaum **Tests**.*
*Änderungen geraten **außer Kontrolle**.*



> “Code **without tests** is **bad code**.

Michael Feathers



J. B. Rainsberger

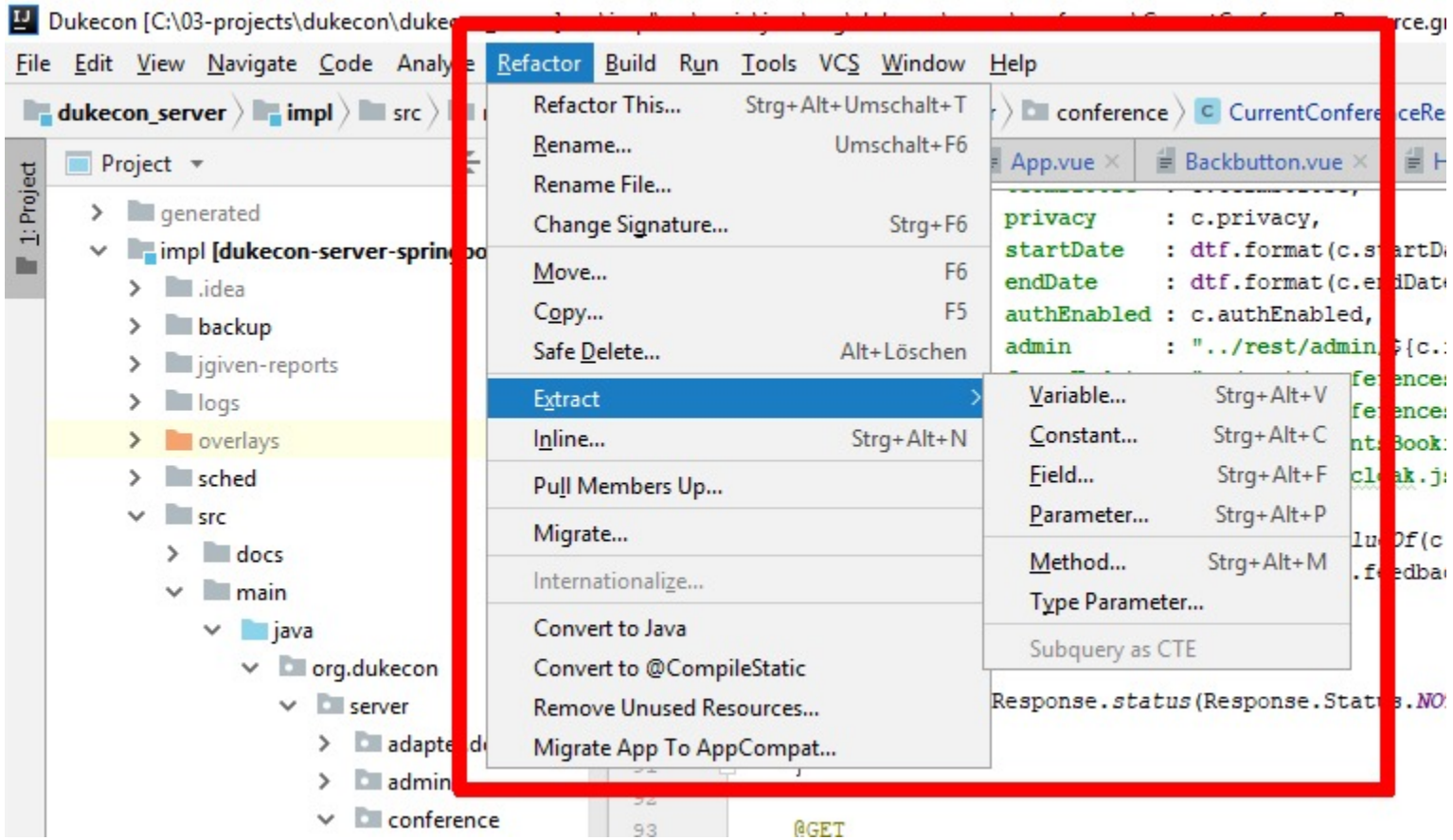
> “Legacy code is **valuable code** that we feel **afraid to change**.

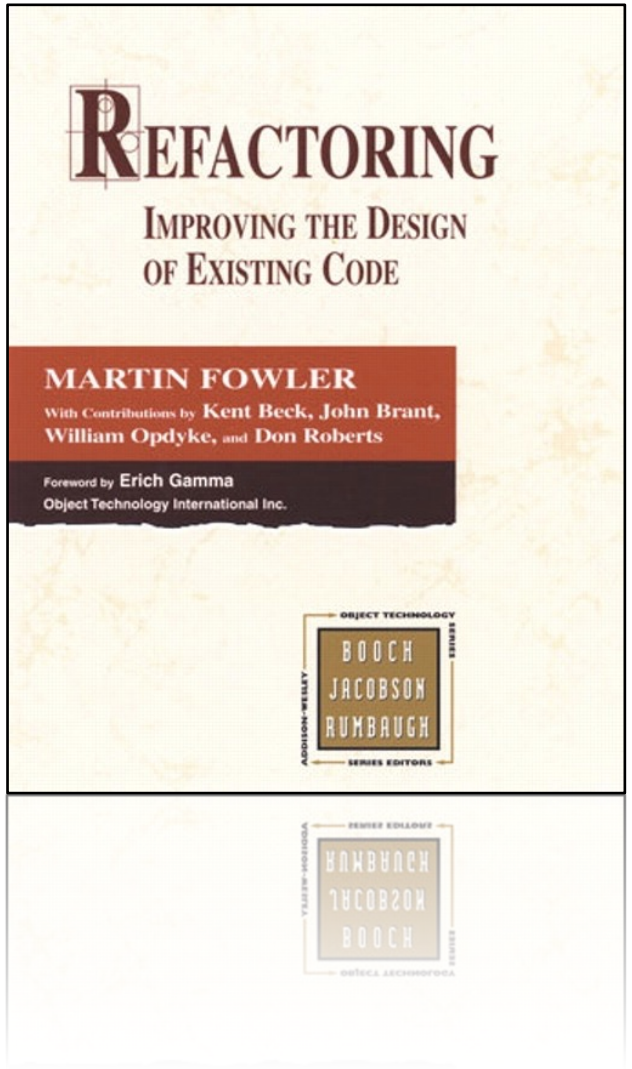
Foto von PublicDomainPictures, CC0 Public Domain Lizenz, https://pixabay.com/de/menschen-abdeckung-schrei-314481/

Gründe für Refactoring

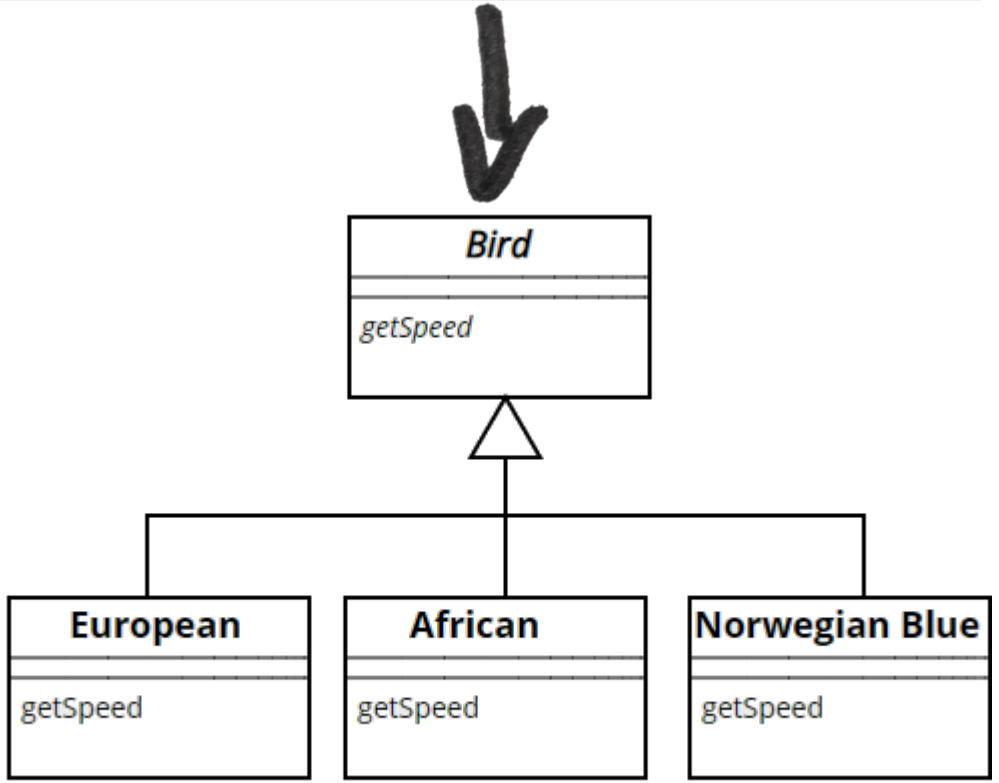
Verstehen
Fehler beheben
Neues Feature
Optimierung

Toolunterstützung bei einfachen Refactorings






```
double getSpeed() {
  switch (_type) {
    case EUROPEAN:
      return getBaseSpeed();
    case AFRICAN:
      return getBaseSpeed() - getLoadFactor() * _numberOfCoconuts;
    case NORWEGIAN_BLUE:
      return (_isNailed) ? 0 : getBaseSpeed(_voltage);
  }
  throw new RuntimeException ("Should be unreachable");
}
```



Agenda

1 Einstieg

2 Legacy Code

3 **Mikado Methode**

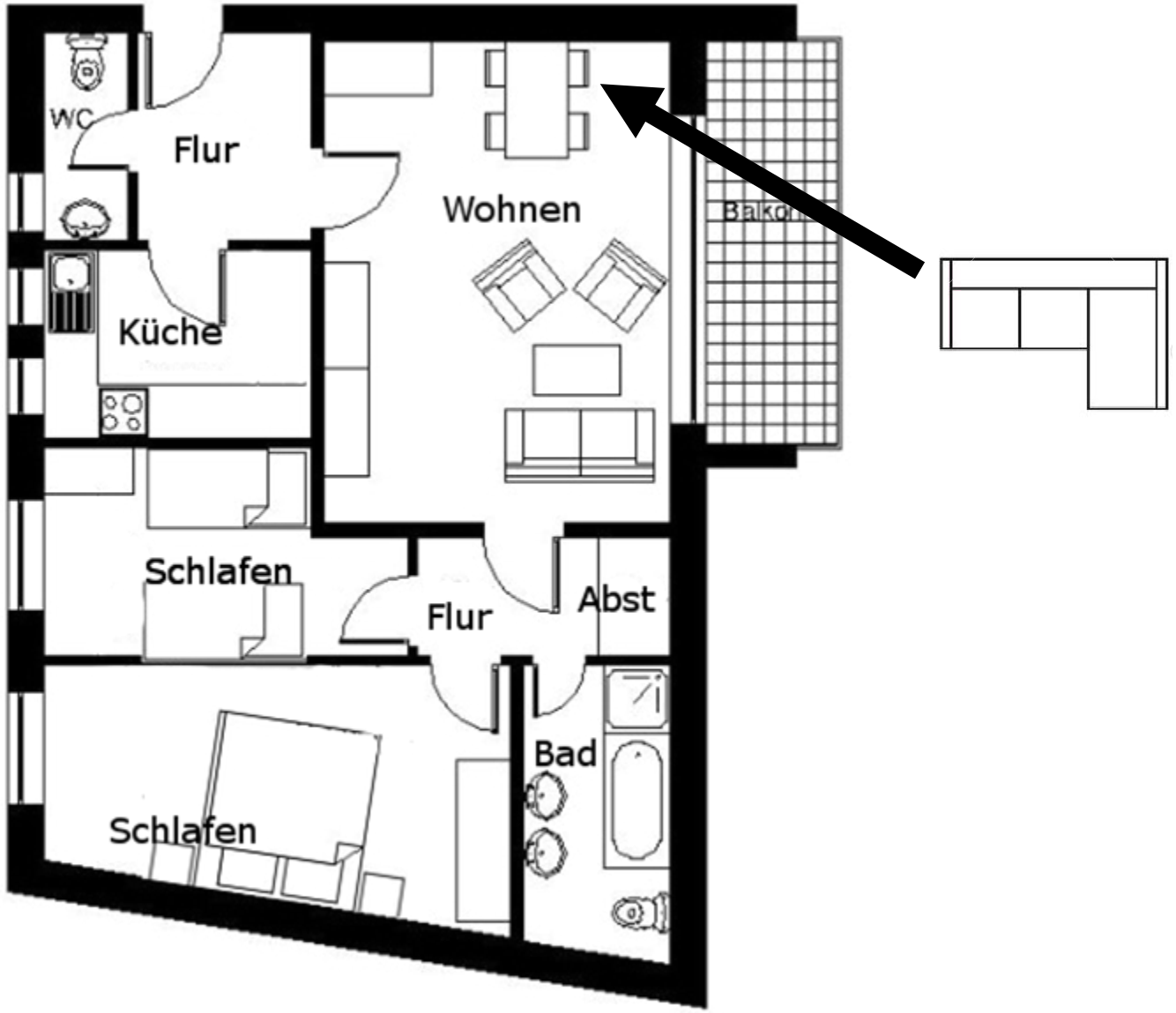
4 Reale Legacy Projekte

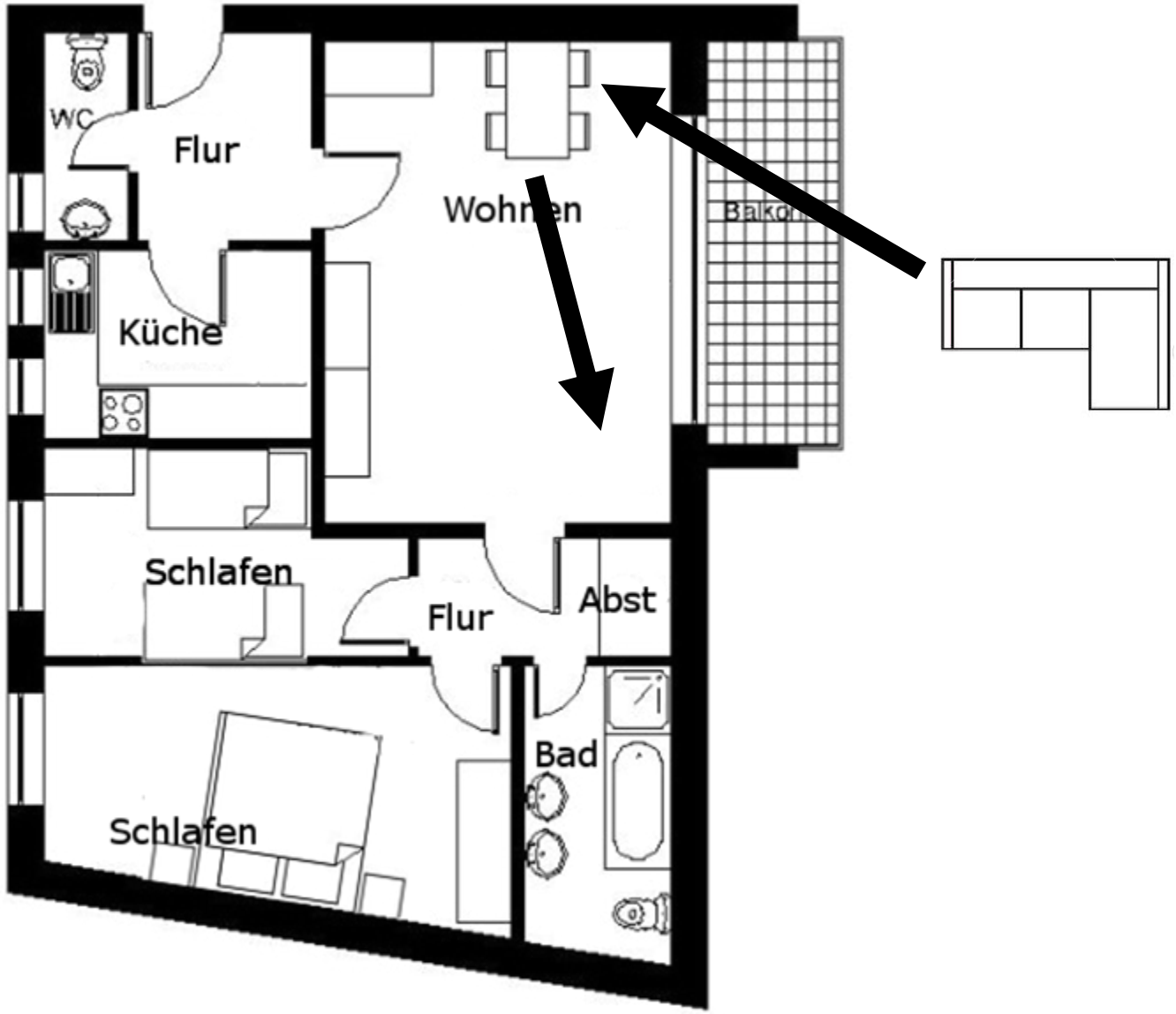
5 Ausblick und weitere Informationen

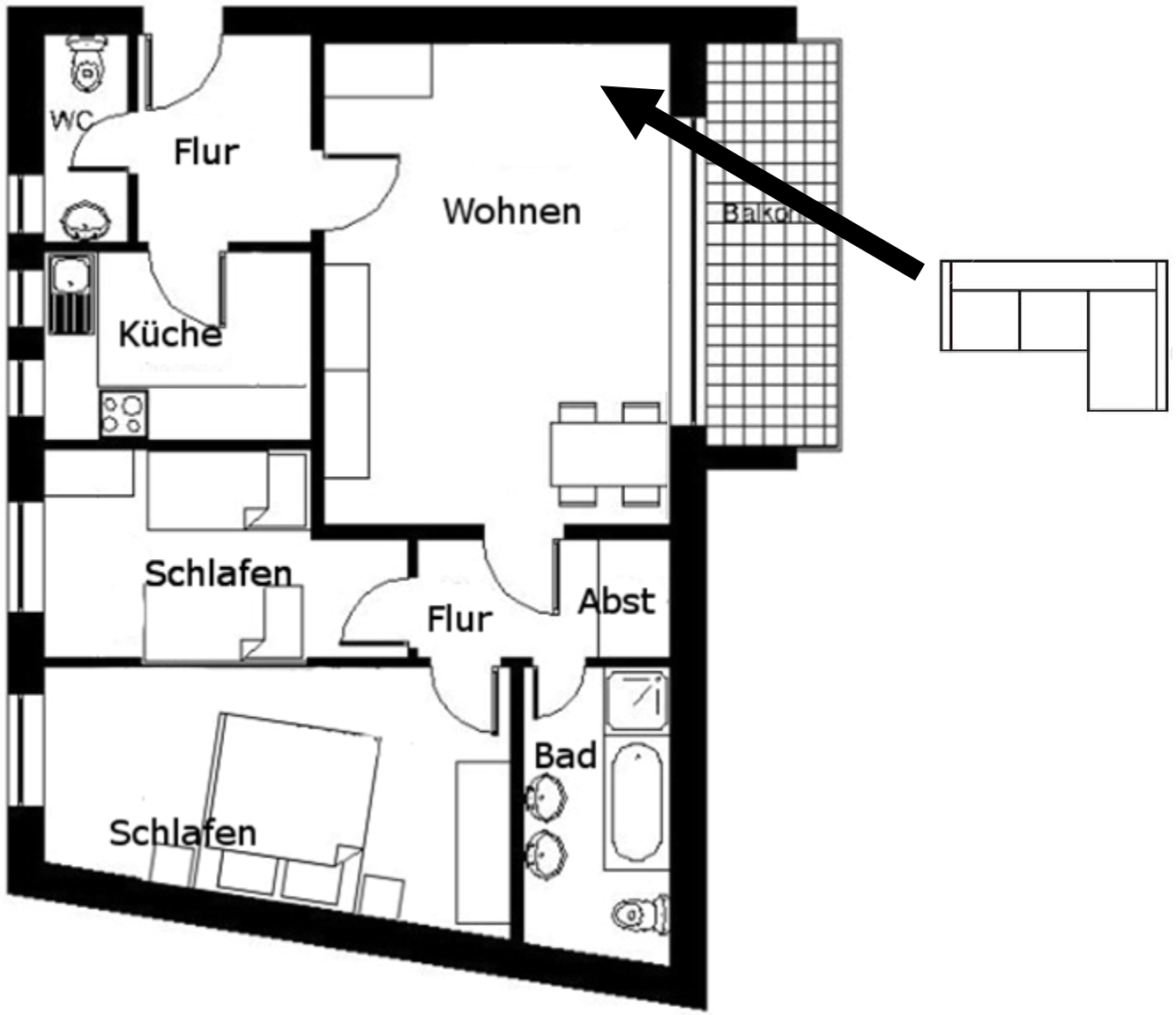
3

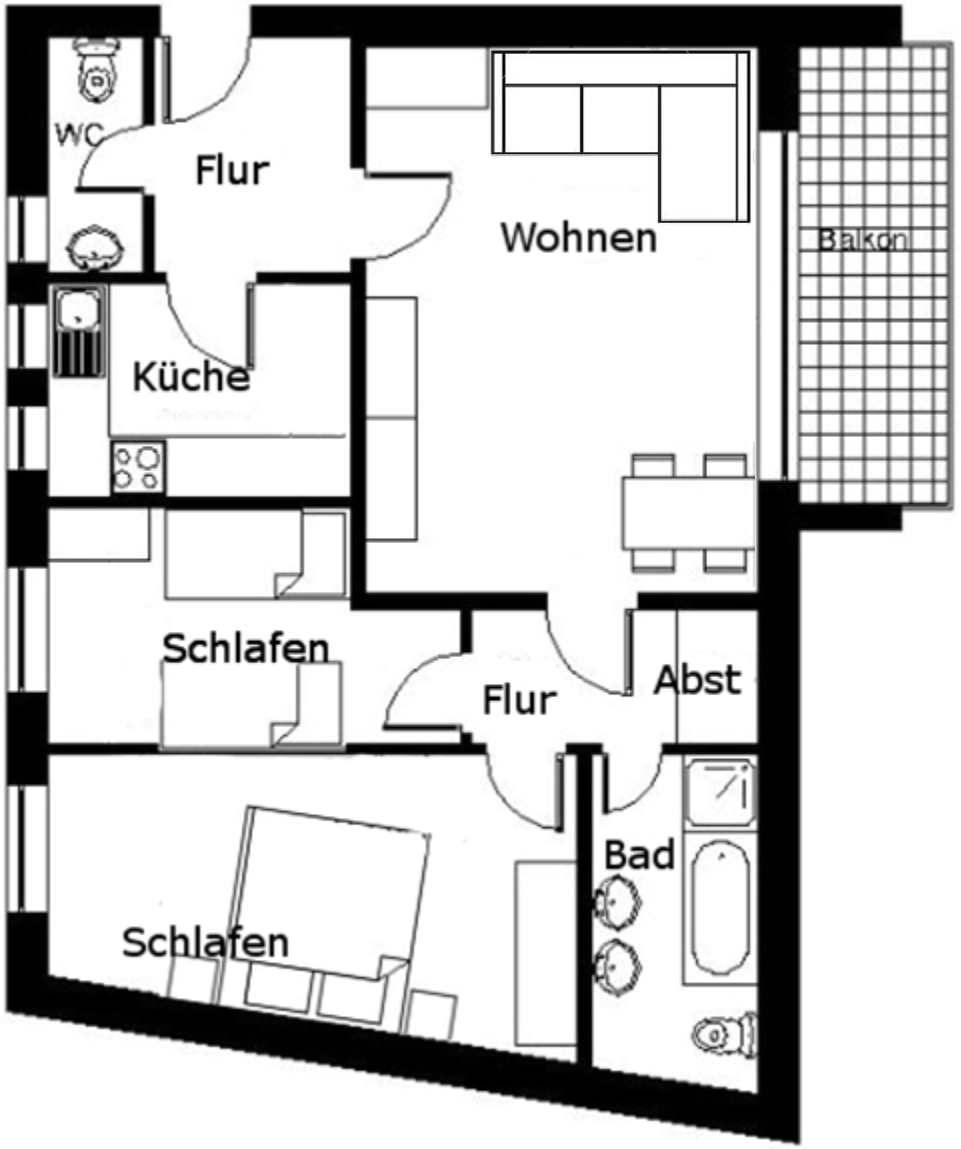
Mikado Methode für Dummies

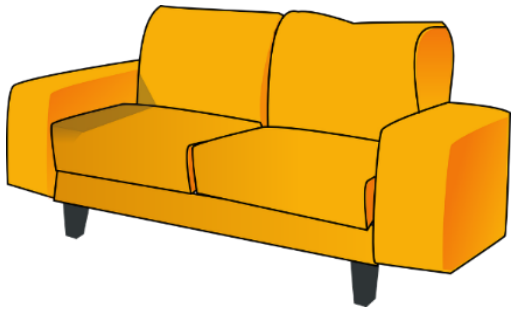
Möbel rücken


WC
Flur
Wohnen
Balkon
Küche
Schlafen
Flur
Abst
Bad
Schlafen


WC
Flur
Wohnen
Balkon
Küche
Schlafen
Flur
Abst
Bad
Schlafen


WC
Flur
Wohnen
Balkon
Küche
Schlafen
Flur
Abst
Bad
Schlafen

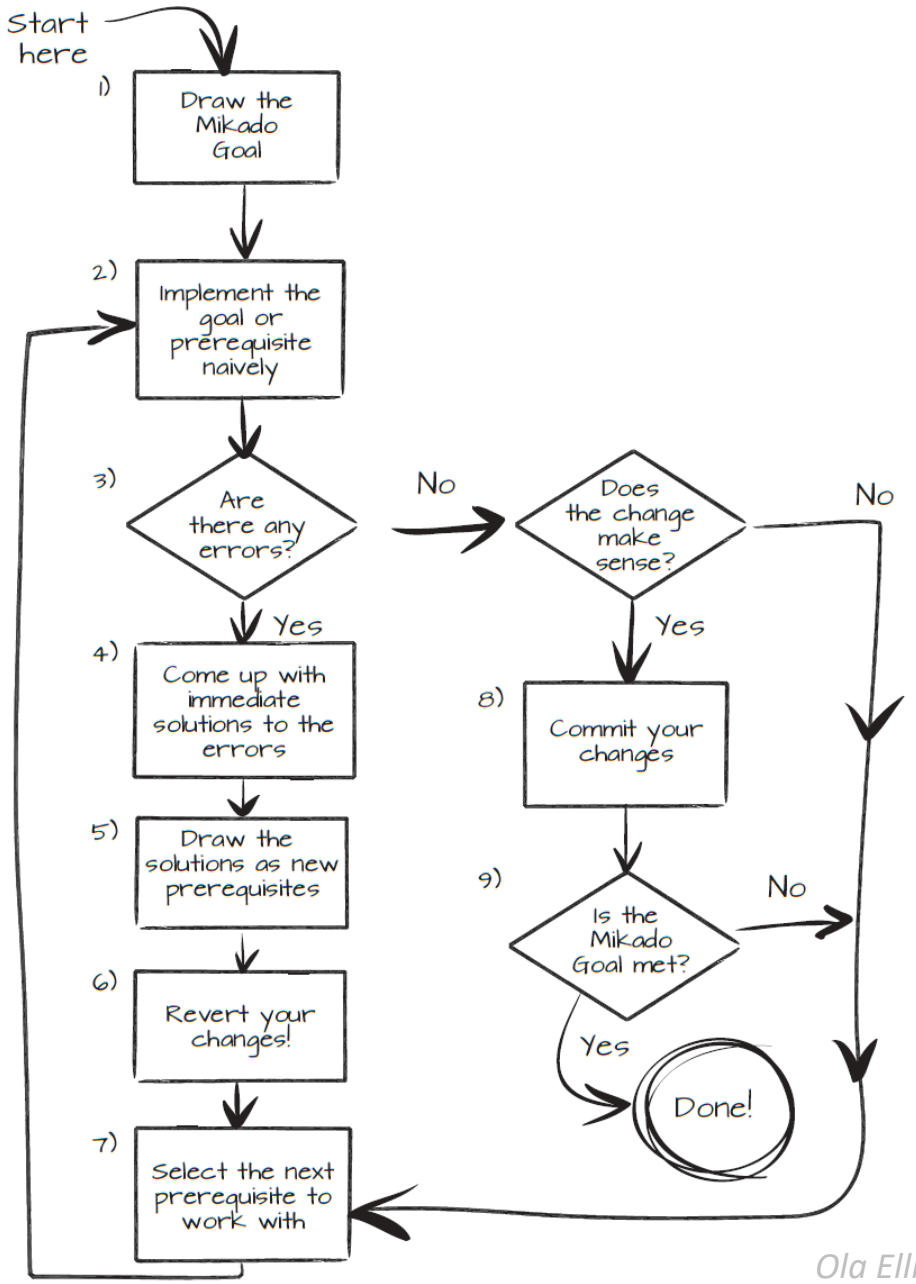

WC
Flur
Wohnen
Balkon
Küche
Schlafen
Flur
Abst
Bad
Schlafen



The Mikado Method can help you ***visualize, plan, and perform*** *business value–focused* ***improvements*** *over several iterations and increments of work,* ***without*** *ever having* ***a broken codebase*** *during the process.*

(Ola Ellnestam, Daniel Brolund: "The Mikado Method")

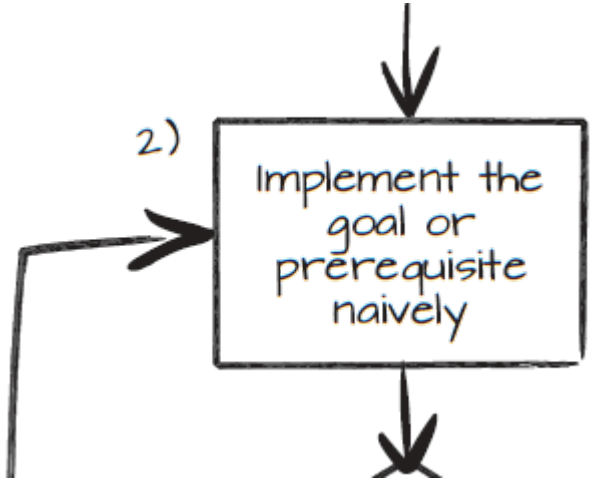
Ablauf im Großen:



Ola Ellnestam, Daniel Brolund: "The Mikado Method"

Konkrete Aufgabe, z. B. User Story.

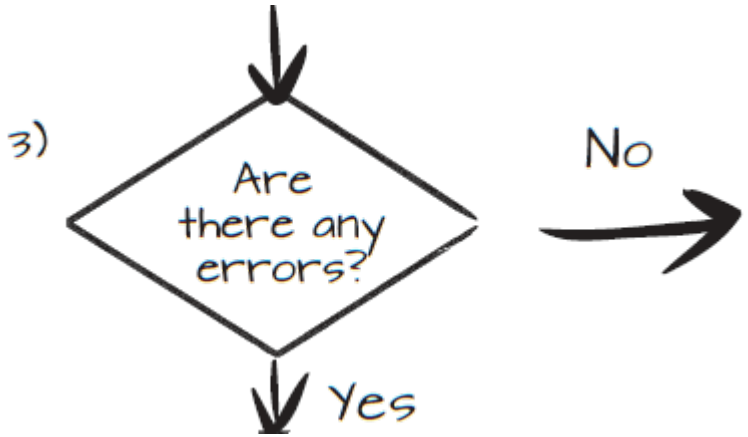
Auf Papier schreiben.



Direkt eine einfache Lösung ausprobieren.

Experimentieren statt analyisieren.

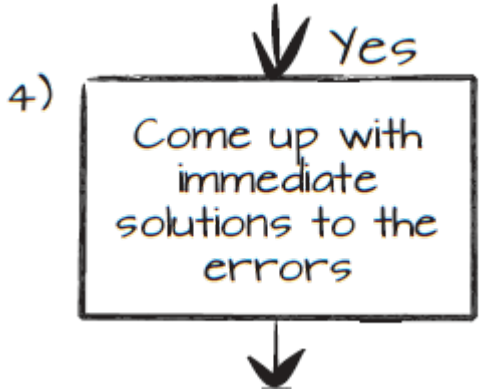
Hilfe durch Compiler und Tests.



Auftretende Fehler haben die Ursache in weiteren Abhängigkeiten.

Zu Schritt 8, wenn keine Fehler.

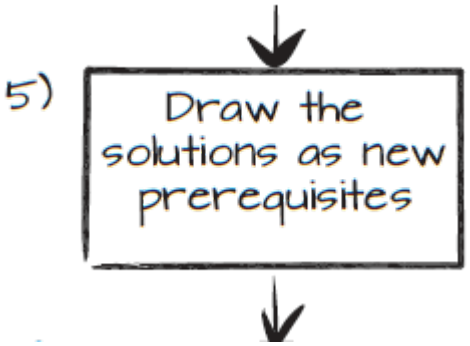
Vorsicht bei Laufzeitfehlern (Nullpointer).



Direkt sofortige Lösungen zu den Fehlern finden.

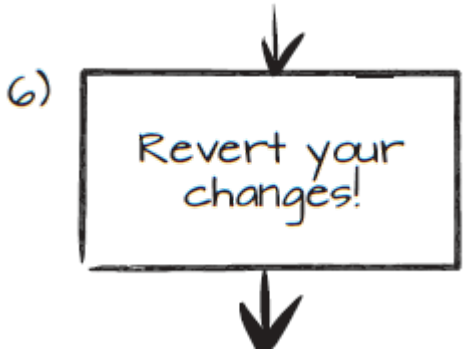
Überanalyisieren vermeiden.

Weiter zugrundeliegende Einschränkungen werden in späteren Iterationen behandelt.



Lösung aufzeichnen und mit Vorgänger verbinden.

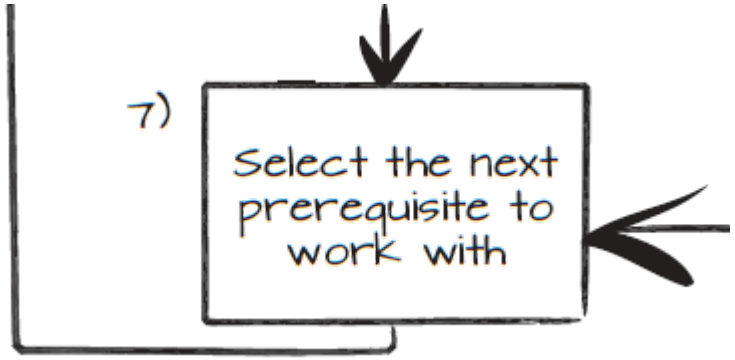
Wissen über das System aufbauen.



Bei Fehlern immer zurückrollen.

Weitere Bearbeitung auf kaputten Zustand sehr fehleranfällig.

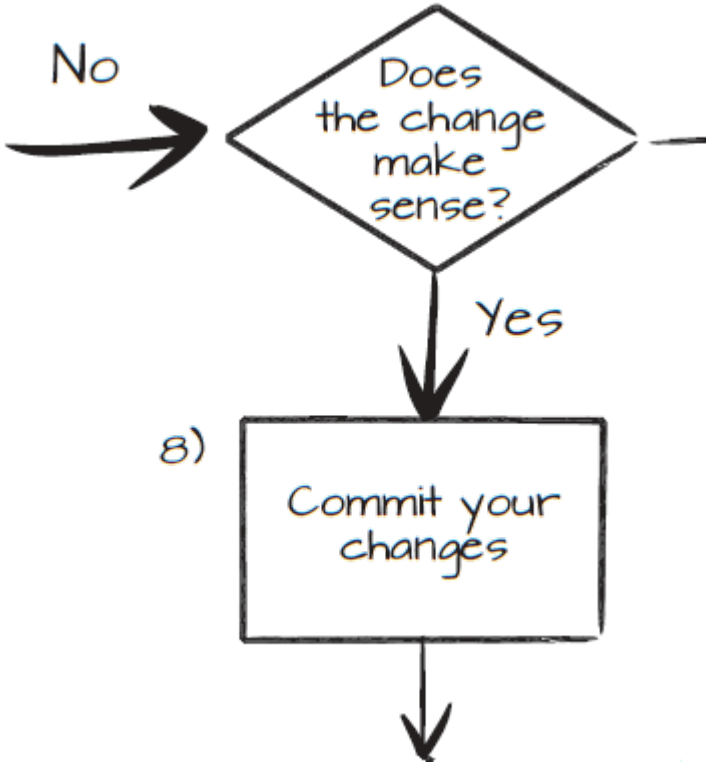
Zurückgerollte Informationen stecken im Graph.



Nächste Vorbedingung auswählen und zu Schritt 2.

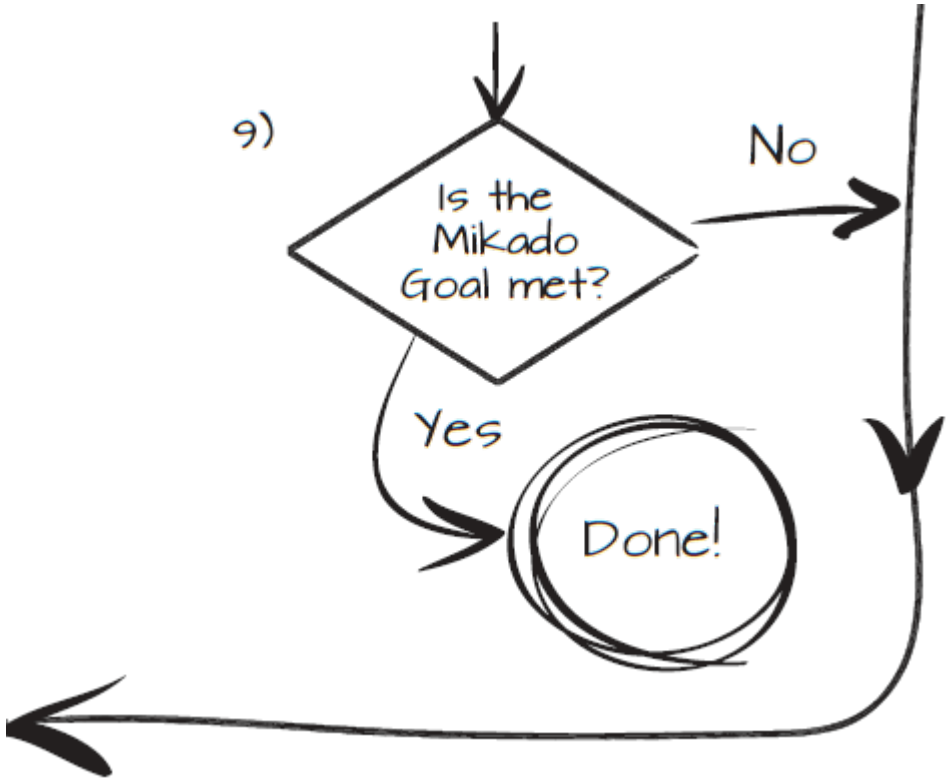
Immer mit sauberen System starten.

Nächste Vorbedingung naiv implementieren …



Vorbedingung abhaken, wenn keine Fehler mehr.

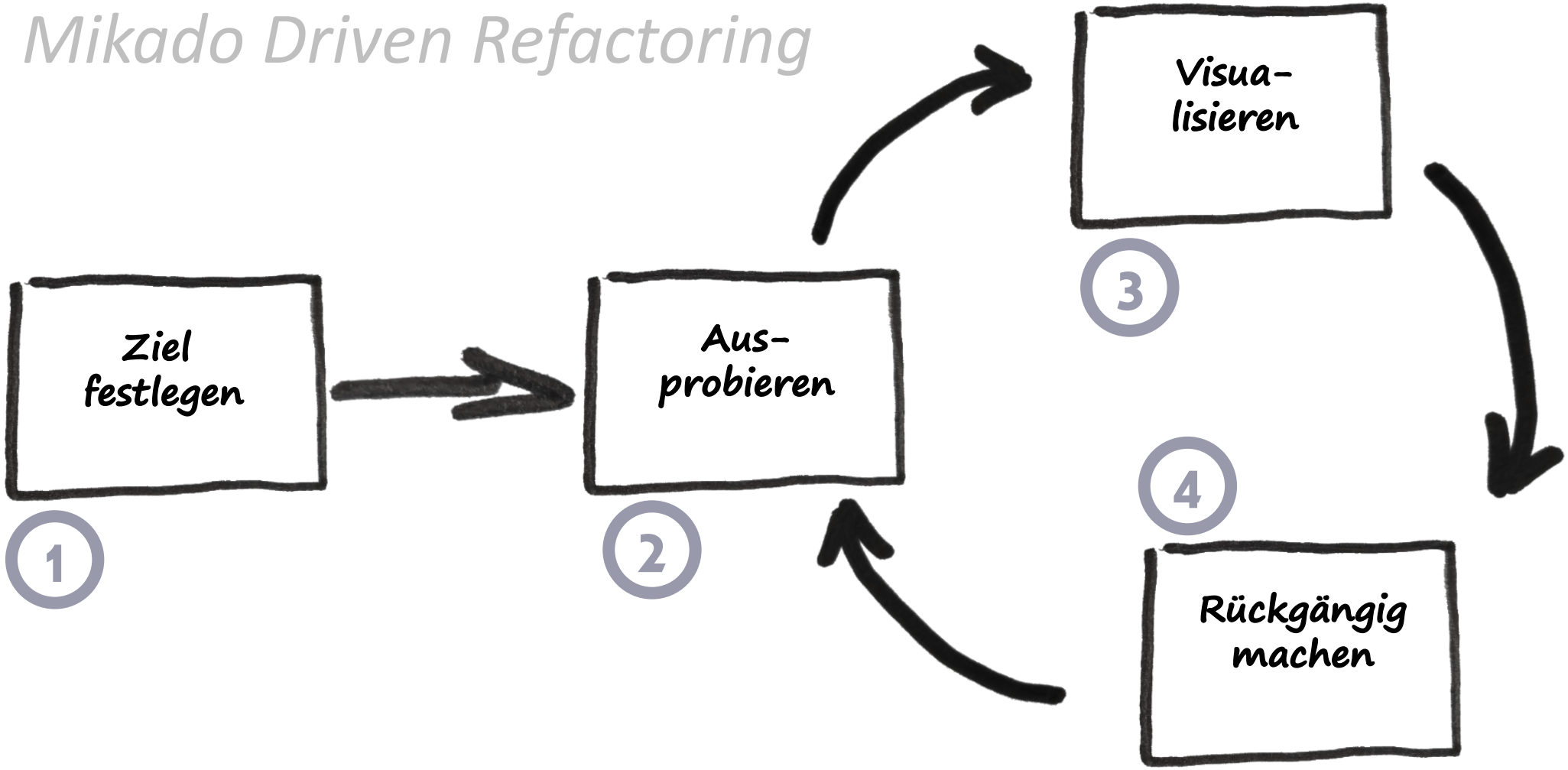
Committen, wenn es Sinn ergibt.



Mikado-Goal und alle Vorbedingungen erfüllt?

Fertig!

Mikado Driven Refactoring



1

Ziel festlegen

=

***Startpunkt** der Änderung.*

***Erfolgskriterium** für Ende.*

2

Ausprobieren

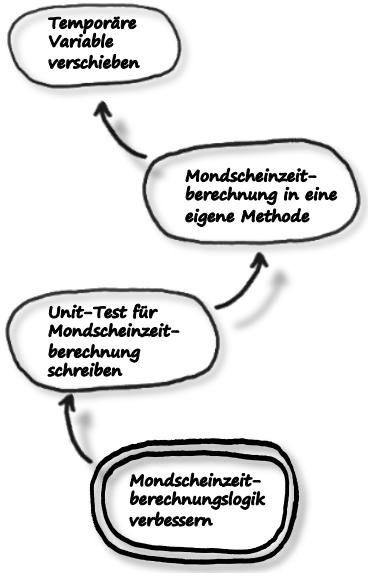
=

Experimentieren.

***Hypothesen** prüfen.*

*Sehen, was **kaputt geht.***

3



Visualisieren

=

***Ziel** und notwendige*

***Vorbedingungen** aufschreiben.*

***Mikado Graph** erstellen.*

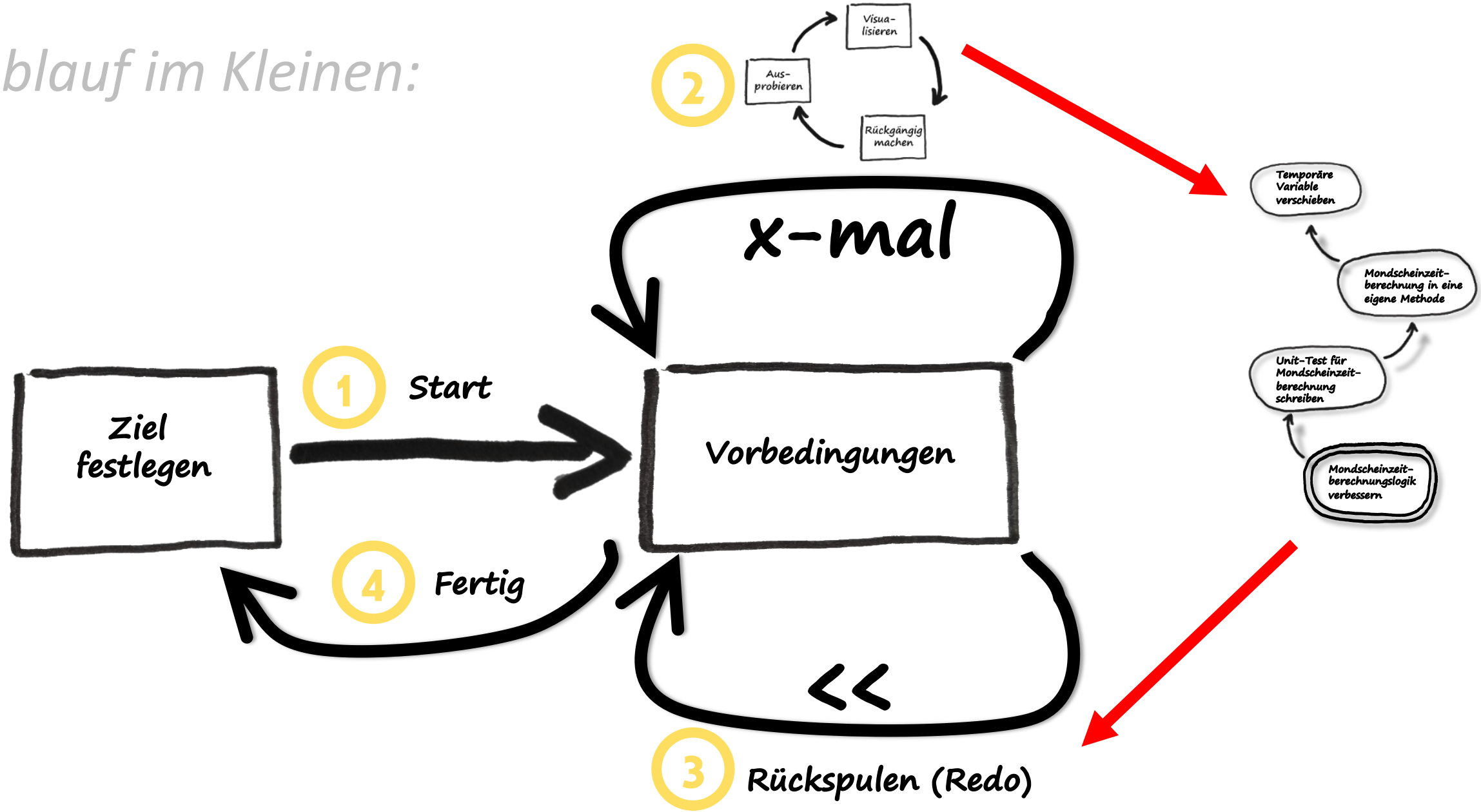
Rückgängig machen

=

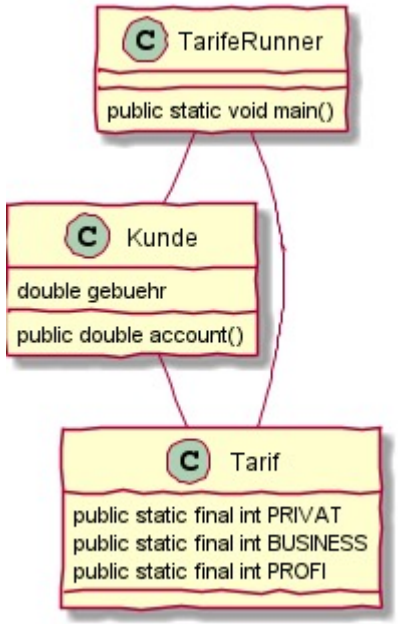
***Vorherigen** funktionierenden*

*Stand **wiederherstellen.***

Ablauf im Kleinen:



```
package de.oio.refactoring.badtelefon;

public class Kunde {
  double gebuehr = 0.0;
  Tarif tarif;

  public Kunde(int tarifArt) {
    this.tarif = new Tarif(tarifArt);
  }

  public void account(int minuten, int stunde, int minute) {
    boolean mondschein = false;
    double preis = 0;

    // Mondscheinzeit ?
    if (stunde < 9 || stunde > 18)
      mondschein = true;

    // Gespraechspreis ermitteln
    switch (tarif.tarif) {
      case Tarif.PRIVAT:
      […]
    }
    gebuehr += preis;
  }

  public double getGebuehr() {
    return gebuehr;
  }
}
```




```
package de.oio.refactoring.badtelefon;

import java.util.Arrays;
import java.util.Random;

public class TarifeRunner {
 public static void main(String args[]) {
  Random random = new Random();
  for(Integer tarif : Arrays.asList(Tarif.PRIVAT, Tarif.BUSINESS, Tarif.PROFI)) {
   System.out.println(String.format("\nVerarbeitung von Tarif %s", tarif));
   Kunde k = new Kunde(tarif);
   [..]
 }
}
```

```
package de.oio.refactoring.badtelefon;

public class Tarif {
     public final static int PRIVAT = 0;
     public final static int BUSINESS = 1;
     public final static int PROFI = 2;

     int tarif = 0;

     public Tarif(int tarif) {
            this.tarif = tarif;
     }
}
```

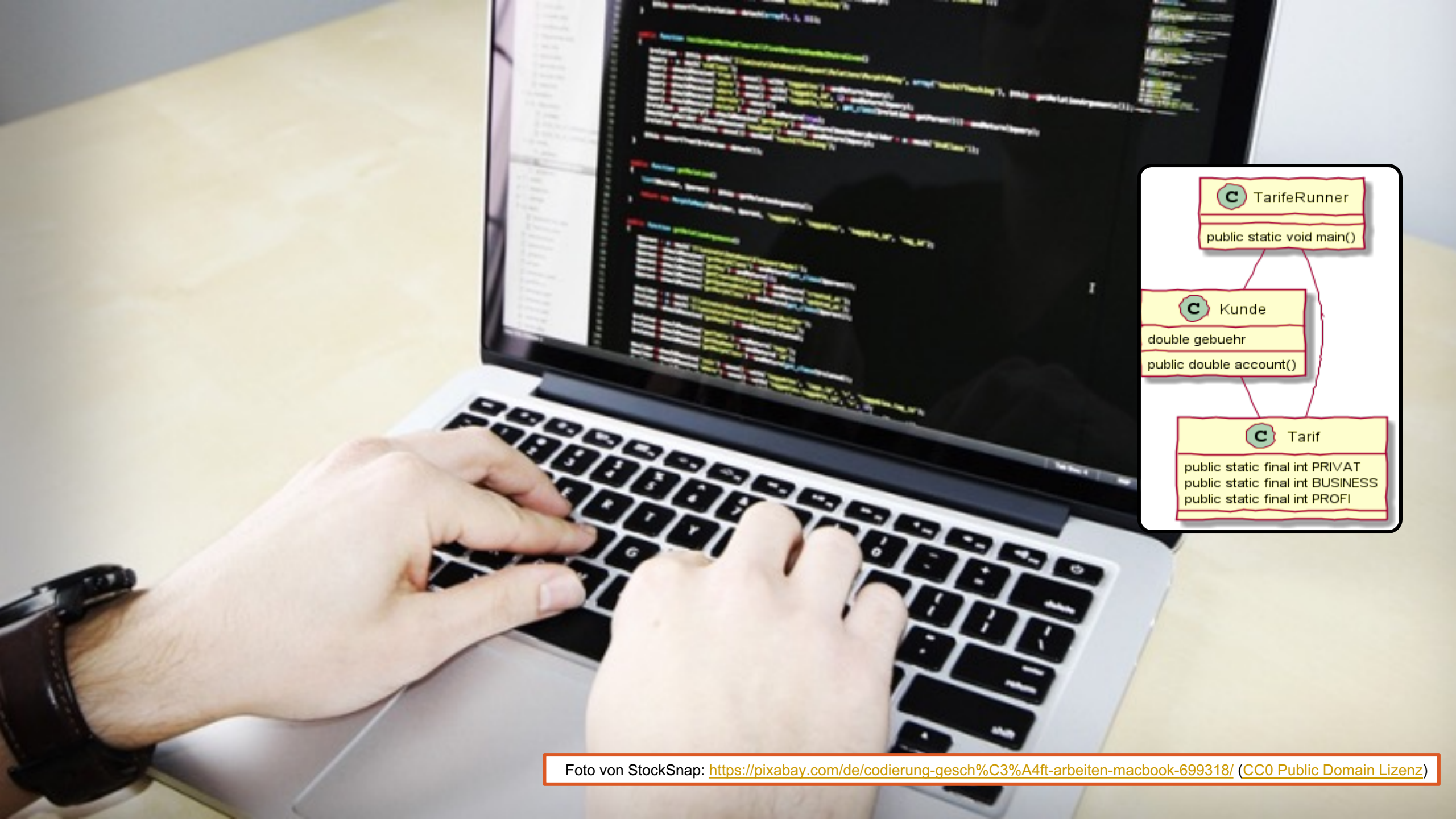


Foto von StockSnap: https://pixabay.com/de/codierung-gesch%C3%A4ft-arbeiten-macbook-699318/ (CC0 Public Domain Lizenz)

Zurückspulen
Experimentieren

Zurückrollen
Visualisieren
Experimentieren

Zurückrollen
Visualisieren
Experimentieren

Zurückrollen
Visualisieren
Experimentieren
Mikado-Ziel festlegen


Temporäre Variable verschieben
Mondscheinzeit-berechnung in eine eigene Methode
Unit-Test für Mondscheinzeit-berechnung schreiben
Mondscheinzeit-berechnungslogik verbessern
Vorbedingungen
<<
Rückspulen (Redo)

Agenda

1 Einstieg

2 Legacy Code

3 Mikado Methode

4 **Reale Legacy Projekte**

5 Ausblick und weitere Informationen

4

Einfach loslegen?

*Es **fehlen** automatisierte **Tests!***

*Machen wir auch nichts **kaputt?***

```
# suppose that our legacy code is this program called 'game'
$ game > GOLDEN_MASTER

# after some changes we can check to see if behaviour has changed
$ game > OUT-01
$ diff GOLDEN_MASTER OUT-01
# GOLDEN_MASTER and OUT-01 are the same

# after some other changes we check again and...
$ game > OUT-02
$ diff GOLDEN_MASTER OUT-02
# GOLDEN_MASTER and OUT-02 are different -> behaviour changed
```

Golden Master

(aka characterization tests)

```java
@Before
public void init() {
    originalSysOut = System.out;
    consoleStream = new ByteArrayOutputStream();
    PrintStream printStream = new PrintStream(consoleStream);
    System.setOut(printStream);
}

@Test
public void testSimpleOutput() {
    System.out.println("Hallo Publikum!");
    System.out.print("Hallo Falk!");
    assertEquals("Hallo Publikum!\r\nHallo Falk!", consoleStream.toString());
}

@After
public void teardown() {
    System.setOut(originalSysOut);
}
```

1

2

3

```
public void account(int minuten, int stunde, int minute) {
    System.out.println(String.format("Berechne Gespräch mit
    boolean mondschein = false;
    double preis = 0;
    this.
    // Mon
    if (st
        mo
    // Ges
```

gebuehr : double - Kunde
tarif : Tarif - Kunde
account(int minuten, int stunde, int minute) : void - Kunde
berechnePreis(int minuten, double d) : double - Kunde

retest

recheck

```
public static void main(String... args) throws Except
  WebDriver driver = new FirefoxDriver();
  driver.get("http://www.retest.de");
  while (true) {
    List<WebElement> links = driver.findElements(By.tagName("a"));
    links.get(random.nextInt(links.size())).click();
    Thread.sleep(500);
    List<WebElement> fields =
        driver.findElements(By.xpath("//input[@type='text']"));
    WebElement field = fields.get(random.nextInt(fields.size()));
    field.sendKeys(randomString());
    Thread.sleep(500);
  }
}
```

4

https://entwicklertag.de/frankfurt/2016/sites/entwicklertag.de.frankfurt.2016/files/slides/Bei%20uns%20testen%20lauter%20Affen_0.pdf

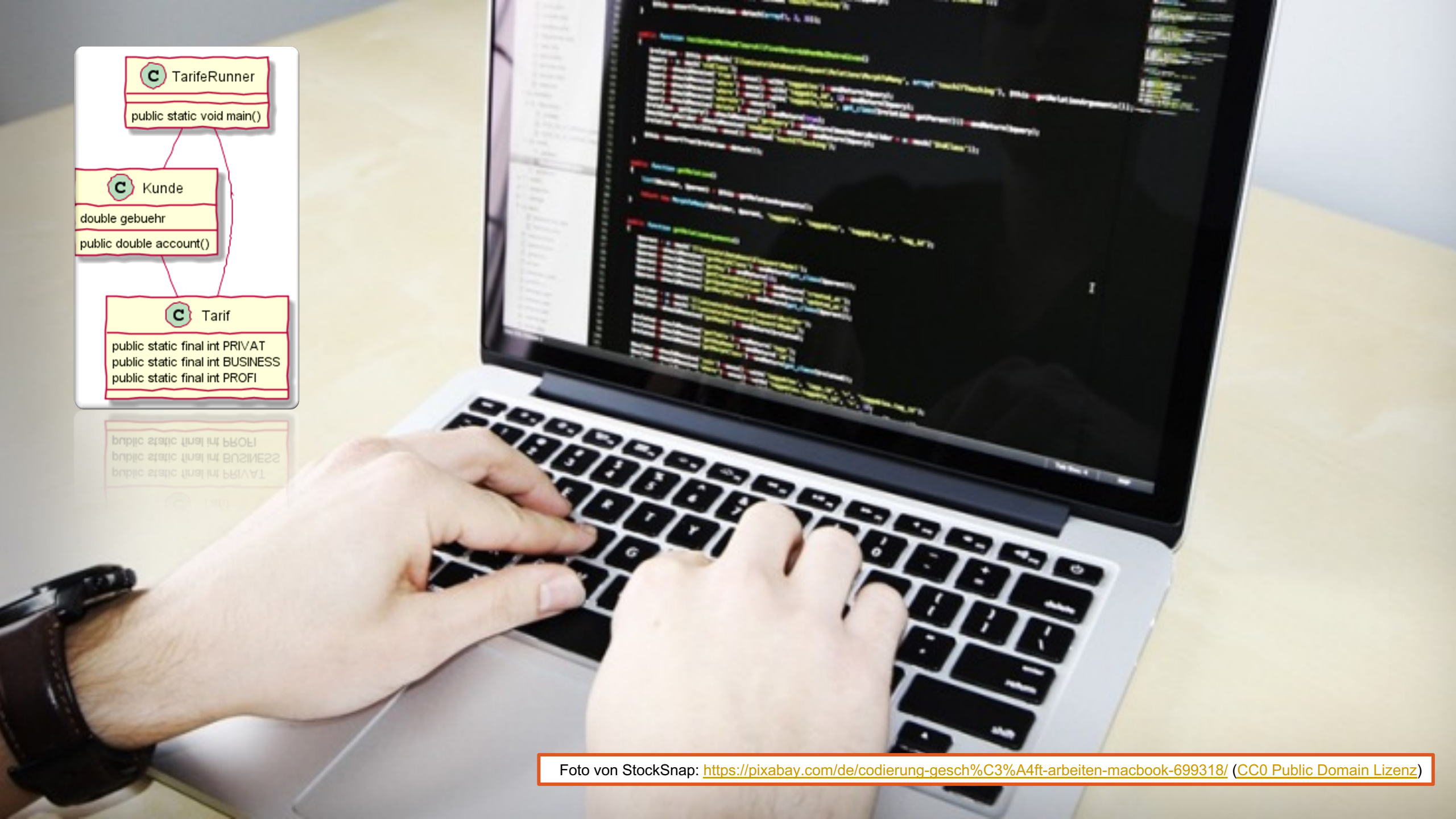
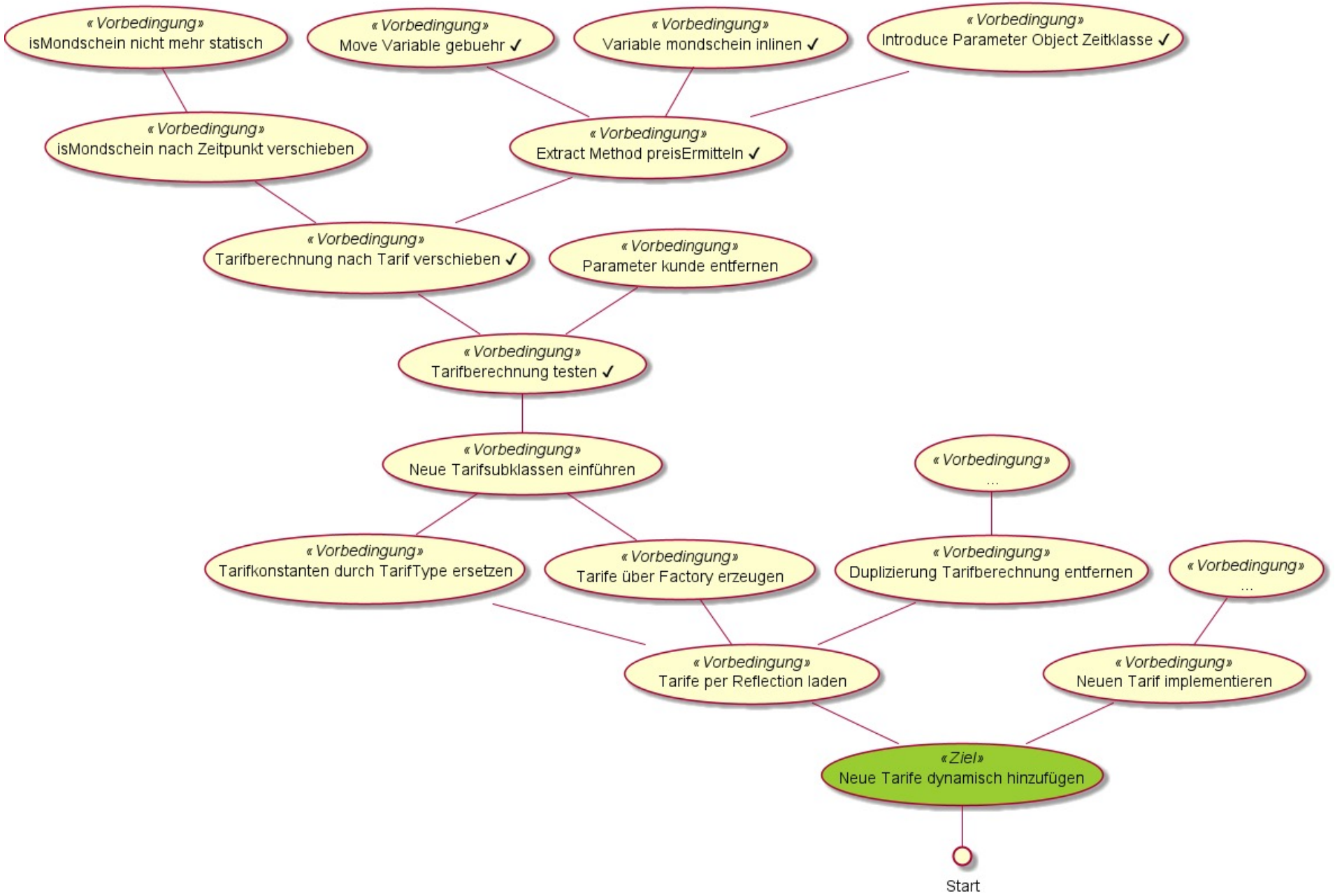


Foto von StockSnap: https://pixabay.com/de/codierung-gesch%C3%A4ft-arbeiten-macbook-699318/ (CC0 Public Domain Lizenz)


« Vorbedingung »
isMondschein nicht mehr statisch
« Vorbedingung »
Move Variable gebuehr ✓
« Vorbedingung »
Variable mondschein inlinen ✓
« Vorbedingung »
Introduce Parameter Object Zeitklasse ✓
« Vorbedingung »
isMondschein nach Zeitpunkt verschieben
« Vorbedingung »
Extract Method preisErmitteln ✓
« Vorbedingung »
Tarifberechnung nach Tarif verschieben ✓
« Vorbedingung »
Parameter kunde entfernen
« Vorbedingung »
Tarifberechnung testen ✓
« Vorbedingung »
Neue Tarifsubklassen einführen
« Vorbedingung »
...
« Vorbedingung »
Tarifkonstanten durch TarifType ersetzen
« Vorbedingung »
Tarife über Factory erzeugen
« Vorbedingung »
Duplizierung Tarifberechnung entfernen
« Vorbedingung »
...
« Vorbedingung »
Tarife per Reflection laden
« Vorbedingung »
Neuen Tarif implementieren
« Ziel »
Neue Tarife dynamisch hinzufügen
Start

Agenda

1 Einstieg

2 Legacy Code

3 Mikado Methode

4 Reale Legacy Projekte

5 **Ausblick und weitere Informationen**

5

Vorteile

***Stabiles System** trotz Änderungen.*

*Bessere **Kommunikation** und **Zusammenarbeit**.*

***Leichtgewichtig** und **fokussiert**.*

Wann nutzen?

1. ***Architektur im Betrieb verbessern.***
2. ***Brownfield Entwicklung.***
3. ***Refactoring Projekt.***

*In **kleinen Schritten** verbessern.*

*Im **gleichen Branch** neue Features **kontinuierlich ausliefern.***

Häufige Situation.

Existierende Anwendungen verbessern.

***Langdauernde** Verbesserung.*

Eigener Branch.

Angst vorm Revert:

Reverting** scheint wie **Wegwerfen.

*Aber Mikado-Graph **enthält Infos.***

***Wissensaufbau/-transfer** über*
System, Domäne, Technologie.

Angst vorm Revert: Änderungen doch aufheben

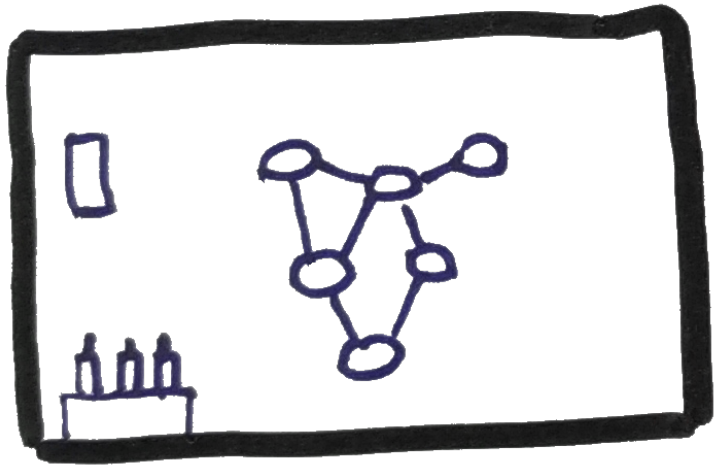
Stash (Git)

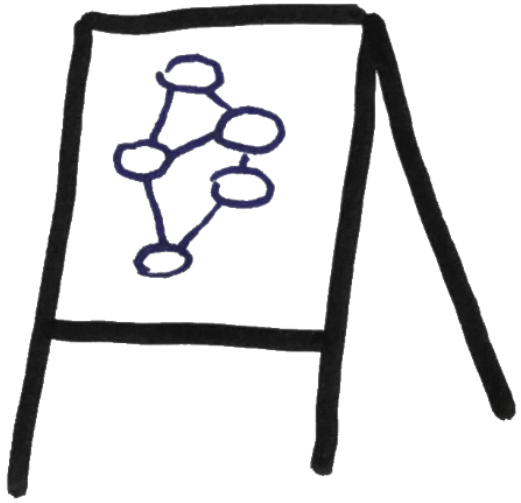
Patch-File (SVN, …)

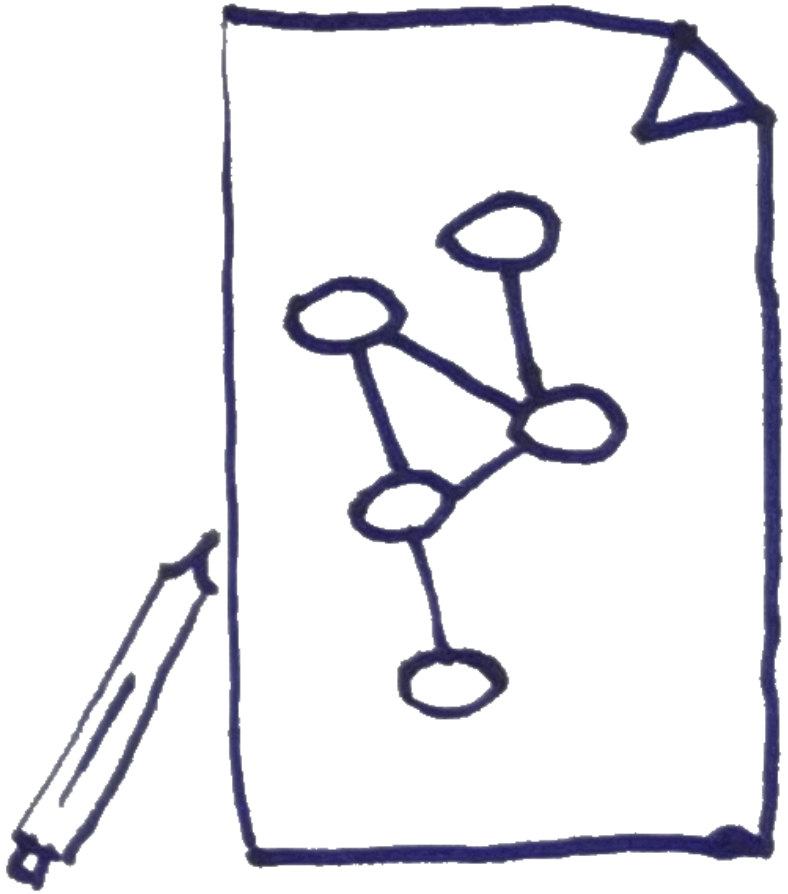
Arbeitsmittel:

1. ***Whiteboard/Flipchart/Papier***
2. ***Mindmap***
3. ***Visio, yEd***

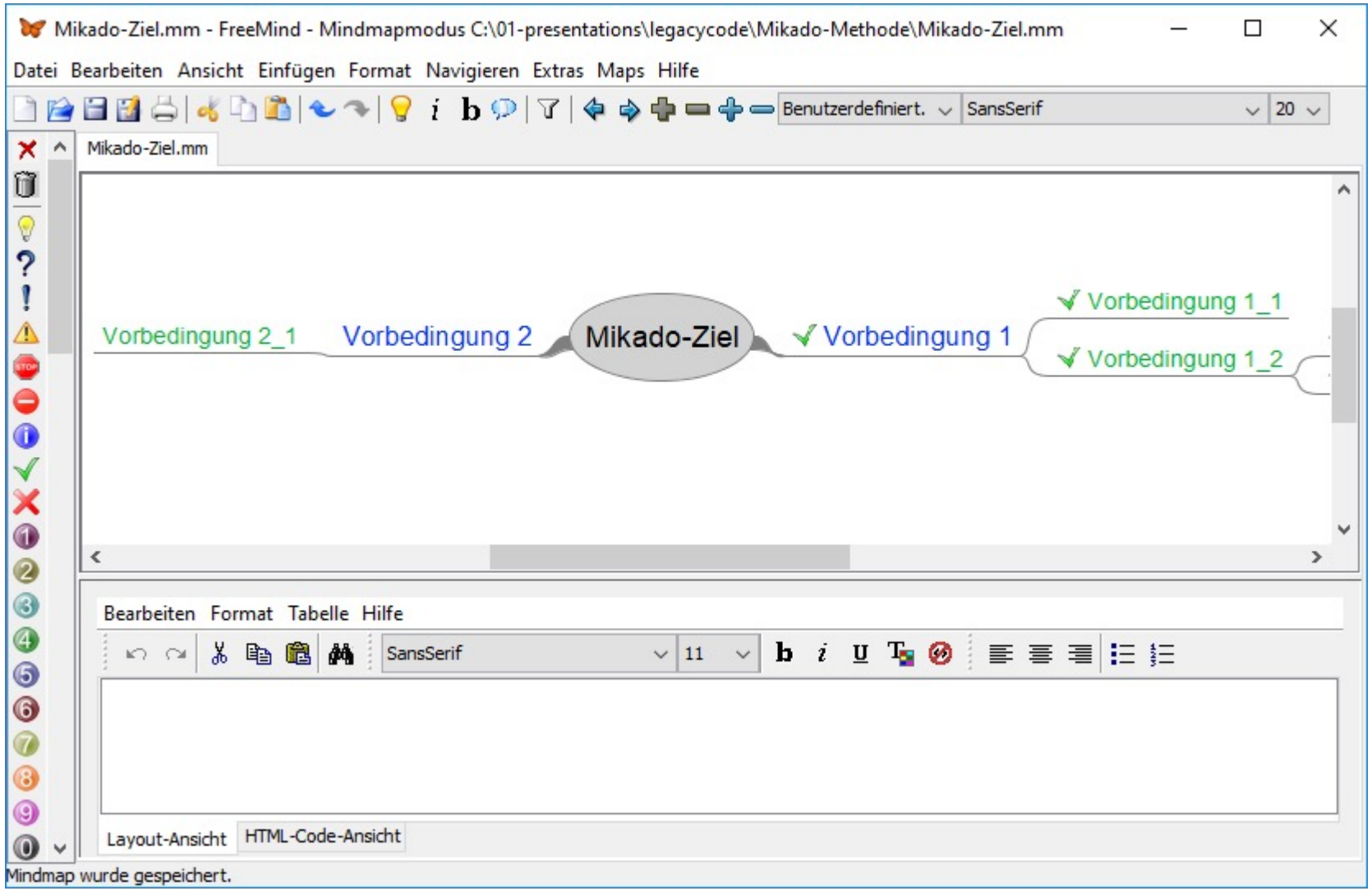
1



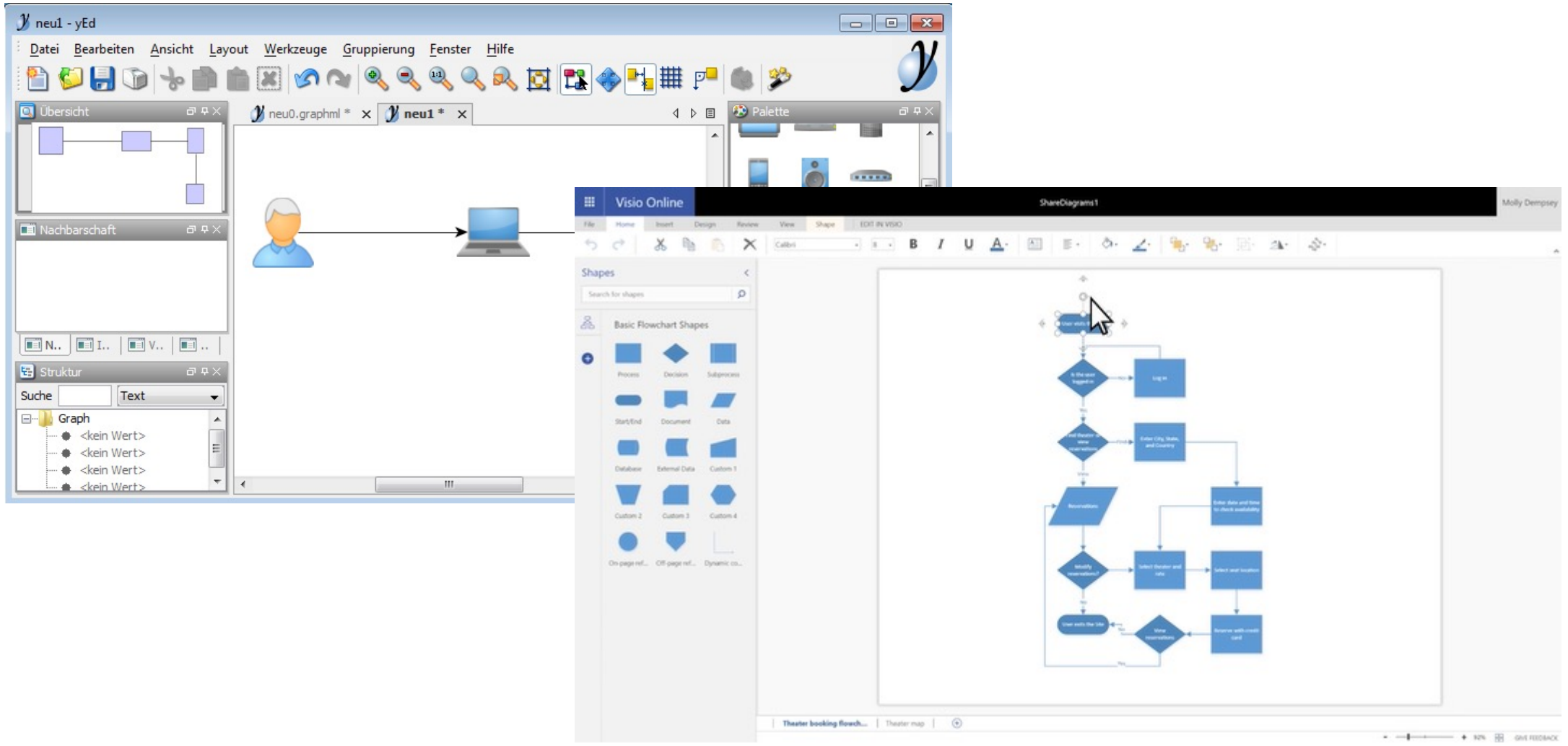


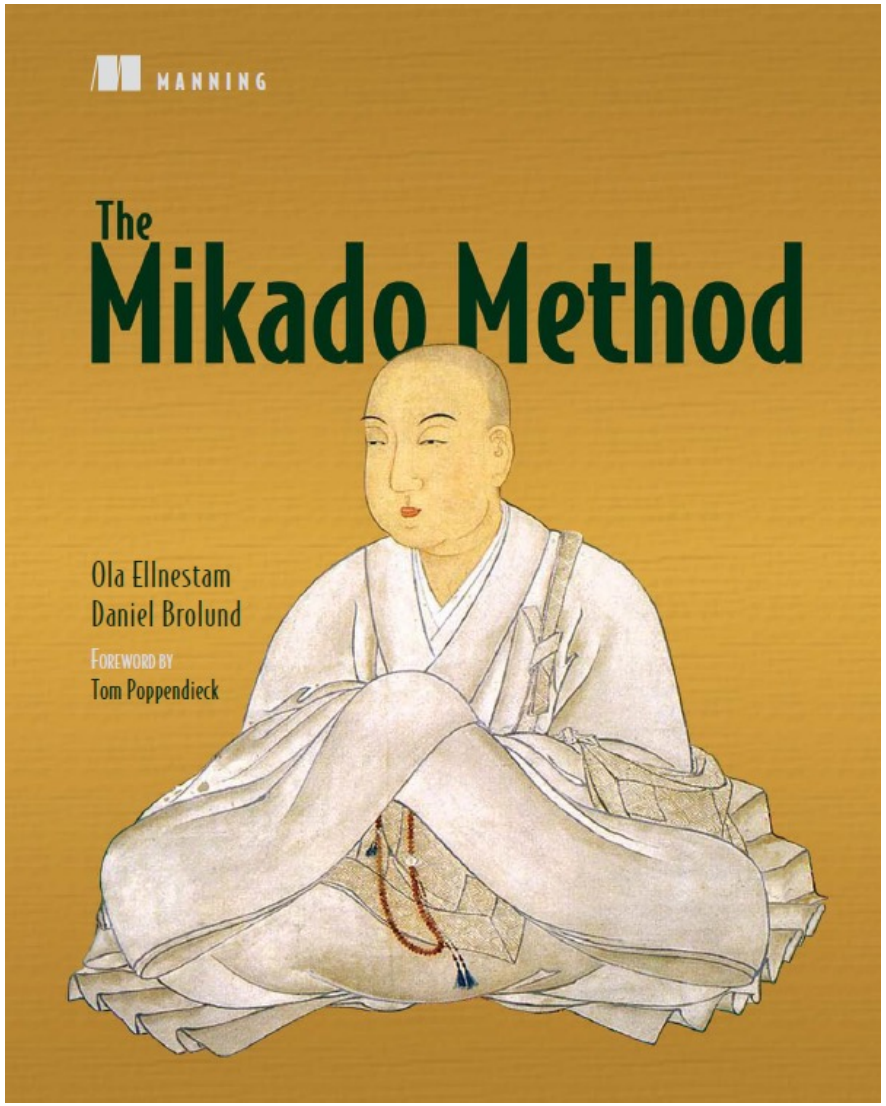


2

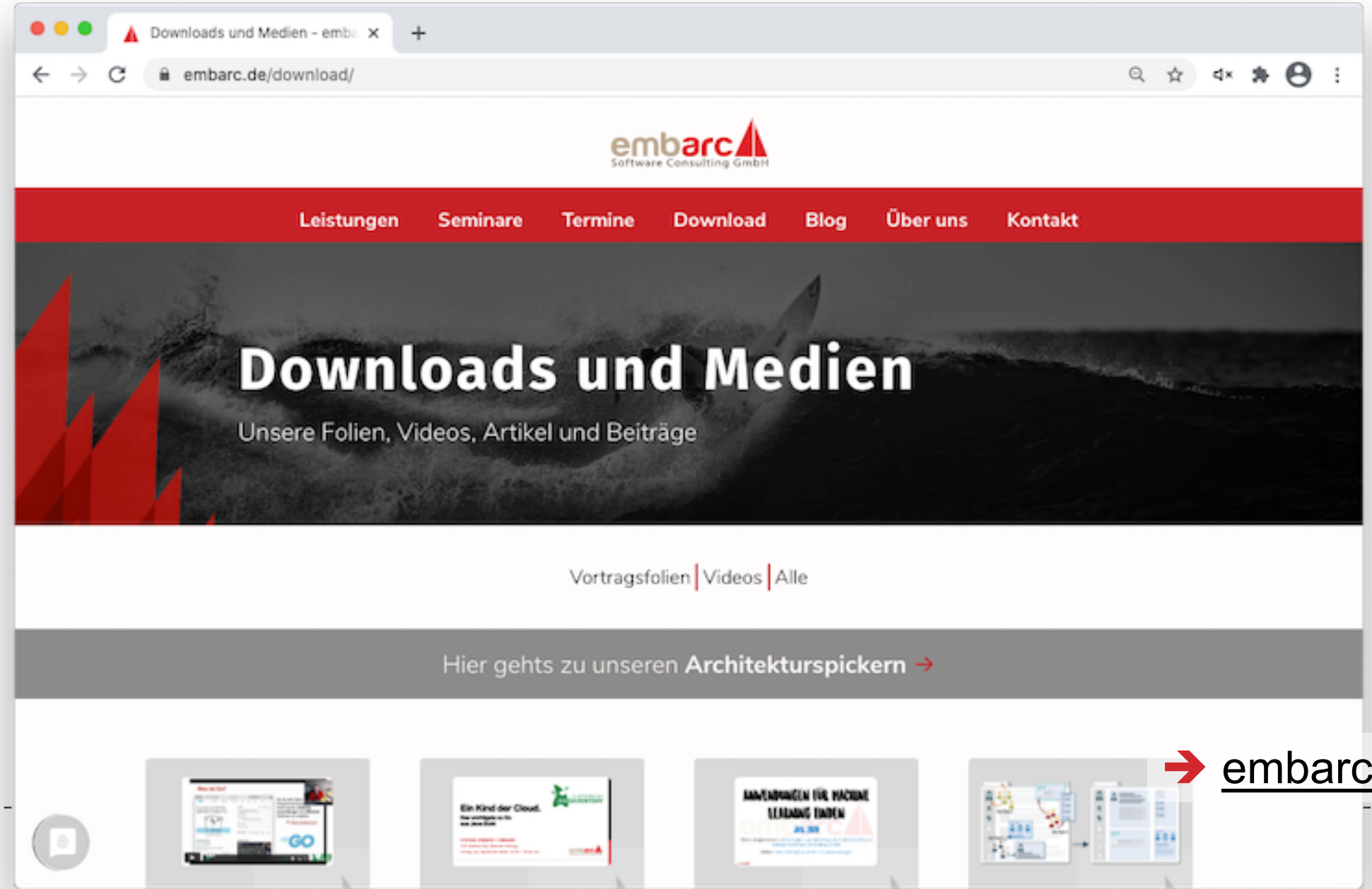

Mikado-Ziel.mm - FreeMind - Mindmapmodus C:\01-presentations\legacycode\Mikado-Methode\Mikado-Ziel.mm
Datei Bearbeiten Ansicht Einfügen Format Navigieren Extras Maps Hilfe
Benutzerdefiniert.
SansSerif
20
Mikado-Ziel.mm
Vorbedingung 2_1
Vorbedingung 2
Mikado-Ziel
Vorbedingung 1
Vorbedingung 1_1
Vorbedingung 1_2
Bearbeiten Format Tabelle Hilfe
SansSerif
11
Layout-Ansicht
HTML-Code-Ansicht
Mindmap wurde gespeichert.

3


neu1 - yEd
Datei Bearbeiten Ansicht Layout Werkzeuge Gruppierung Fenster Hilfe
Übersicht
Nachbarschaft
Struktur
Suche
Text
Graph
<kein Wert>
<kein Wert>
<kein Wert>
<kein Wert>
neu0.graphml *
neu1 *
Palette
Visio Online
Shapes
Basic Flowchart Shapes


MANNING
The
Mikado Method
Ola Ellnestam
Daniel Brolund
Foreword by
Tom Poppendieck

Folien von heute als PDF zum Download



→ embarc.de/download/

Vielen Dank.

Ich freue mich auf Eure Fragen!

Falk Sippach

fs@embarc.de

@sippsack

➔ xing.to/fsi