

TDD

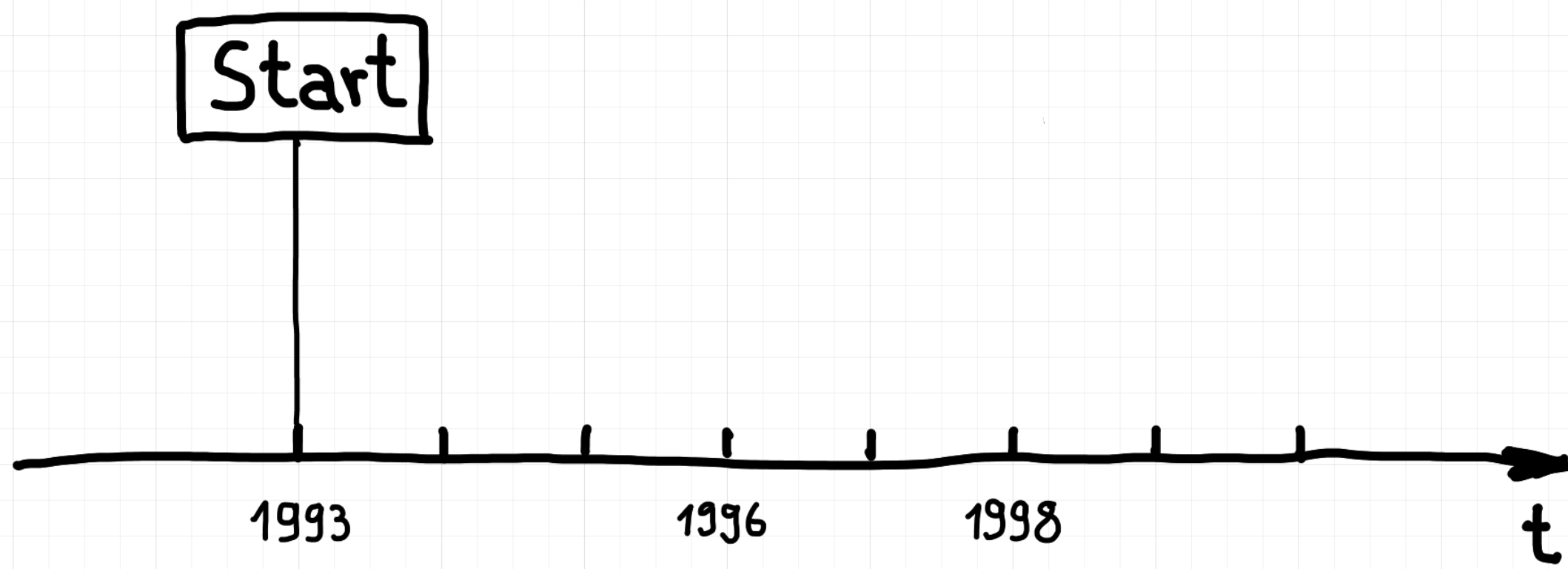
Von der Entstehung bis zur aktuellen Relevanz in 2026

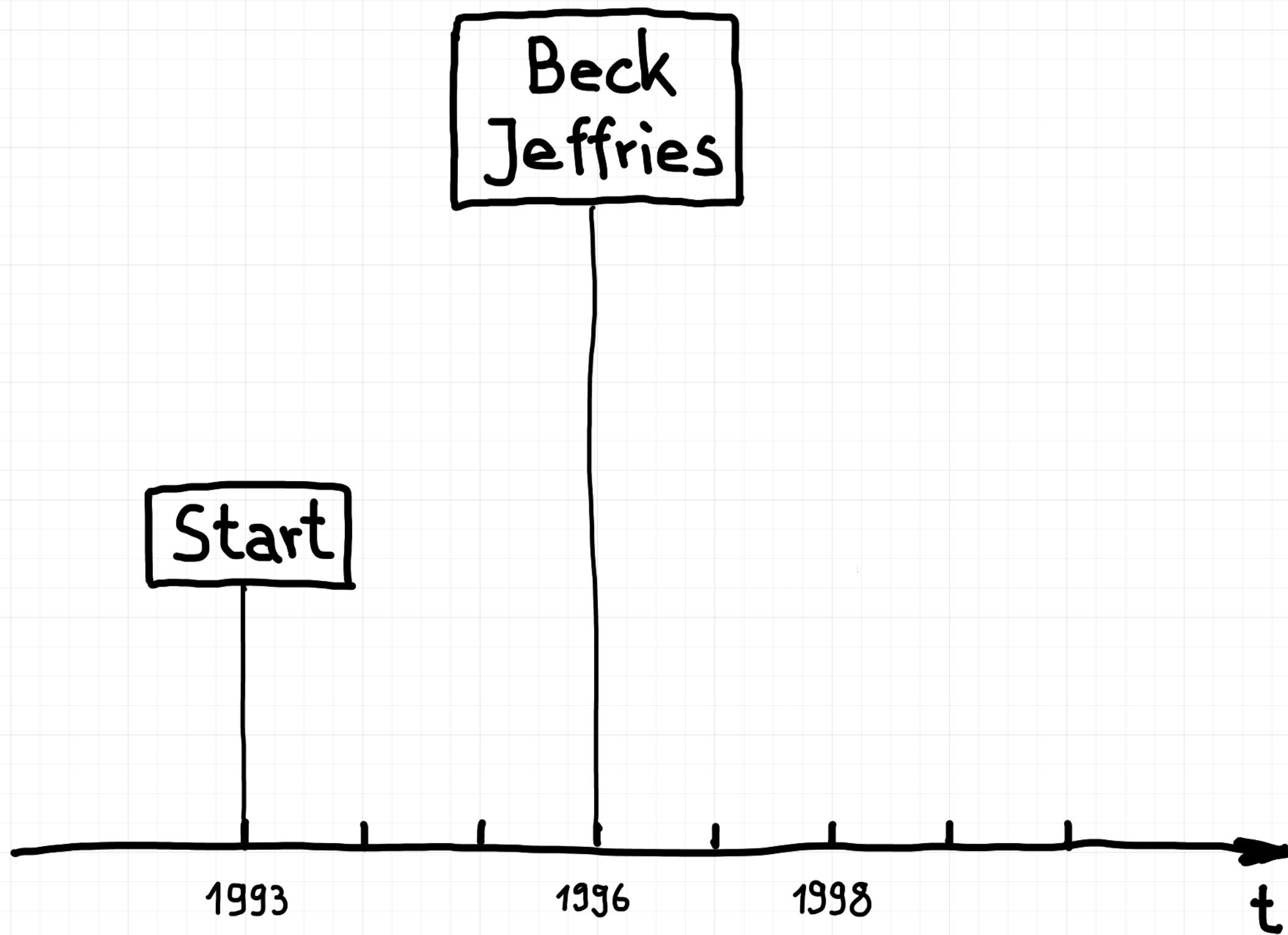
Dominik Hauser, @dasdom
iOS- und macOS-Entwickler

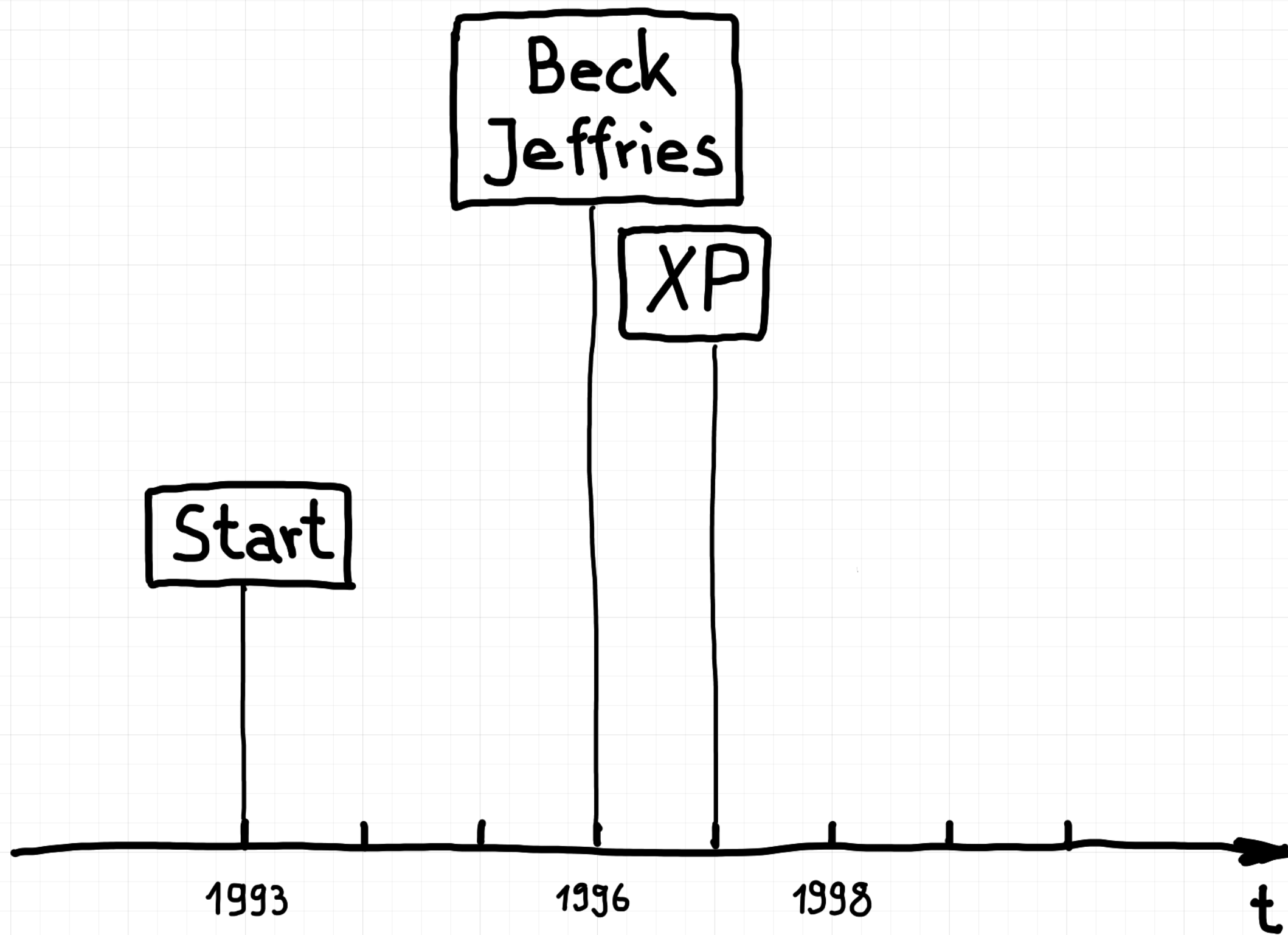
Das C3-Projekt¹

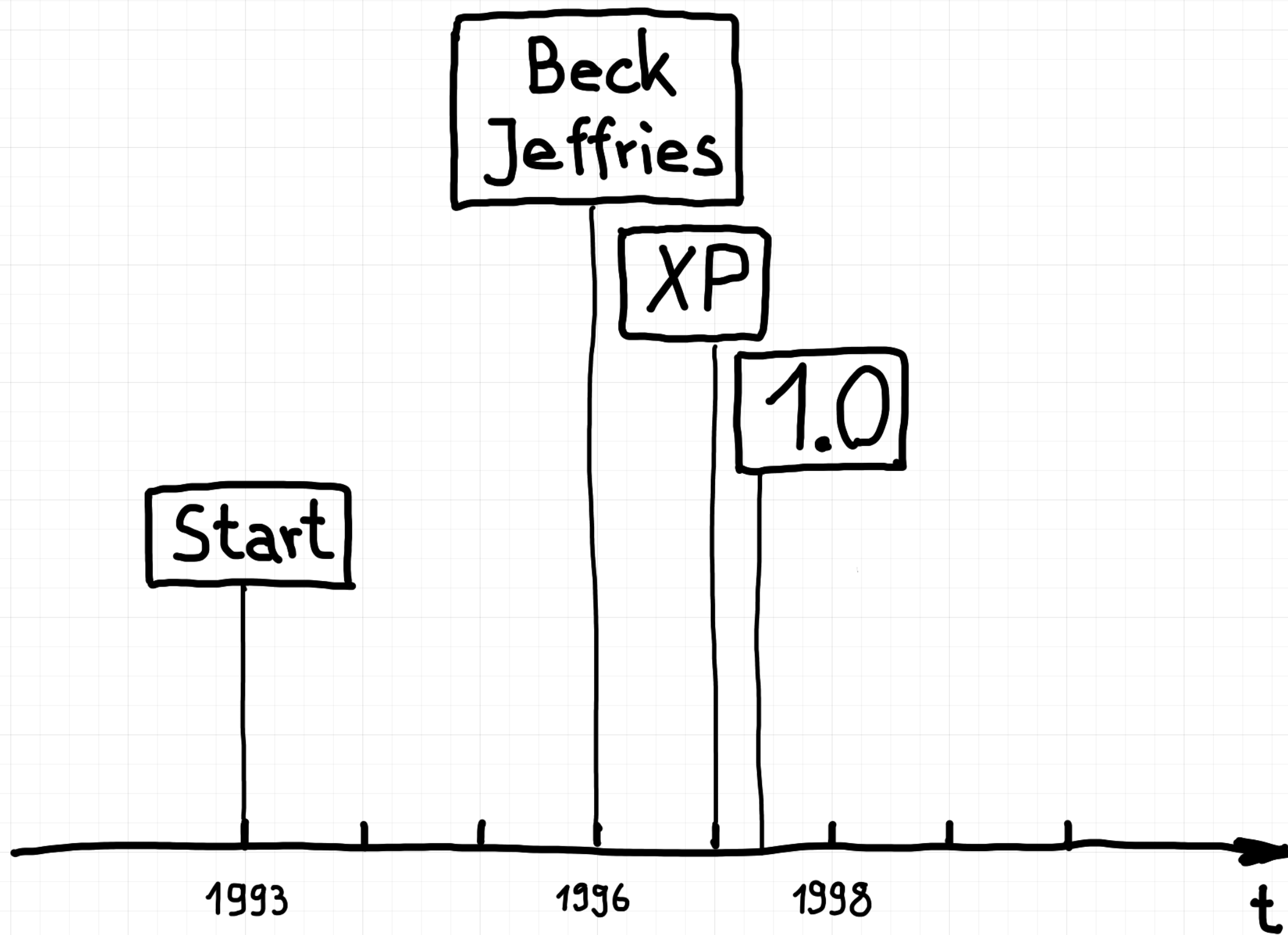
- Chrysler Comprehensive Compensation System
- Smalltalk und GemStone
- Ziel: Gehaltsabrechnungen für 87.000 Mitarbeiter bis 1999

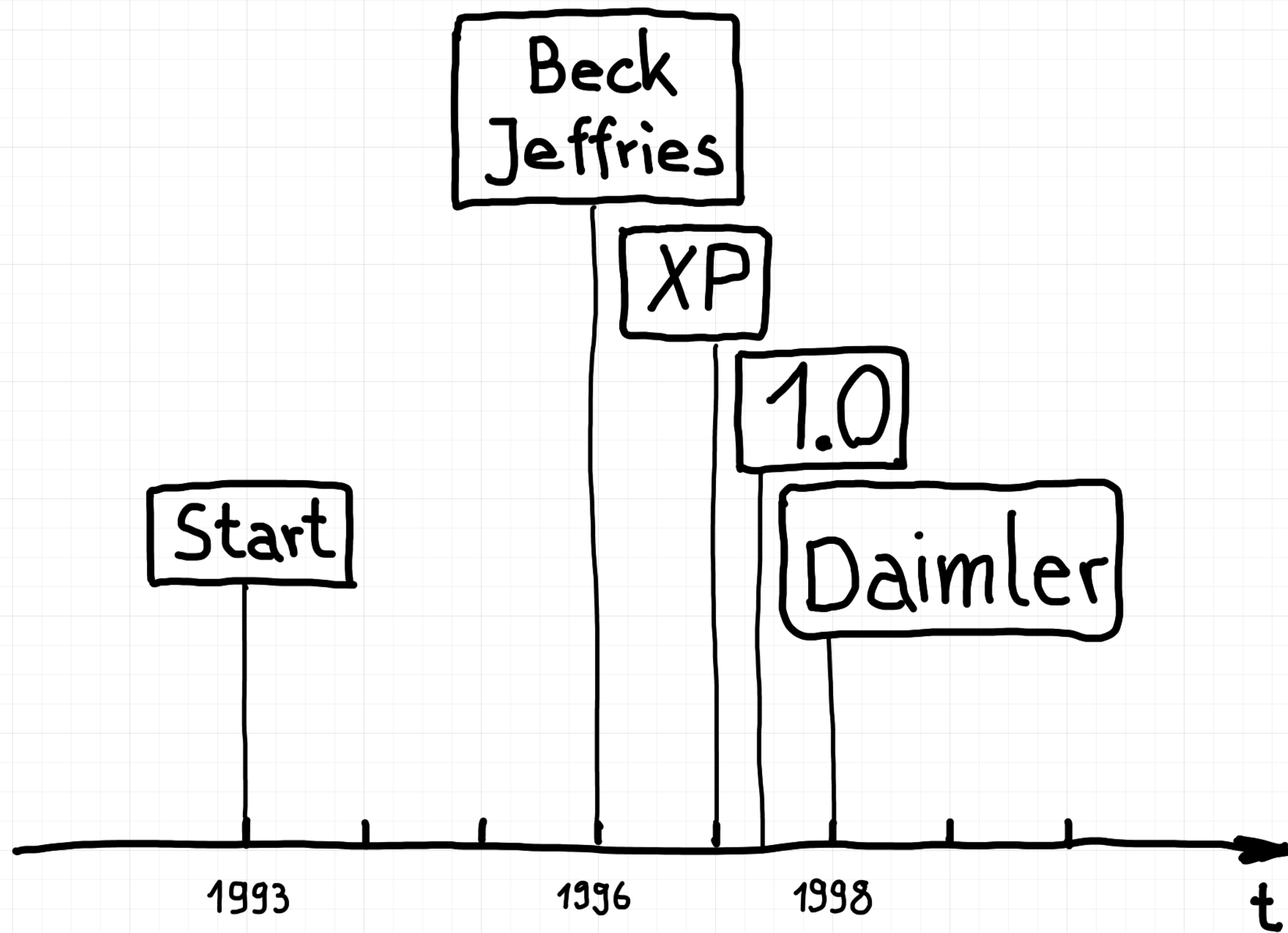
¹ Quelle [Wikipedia](#).

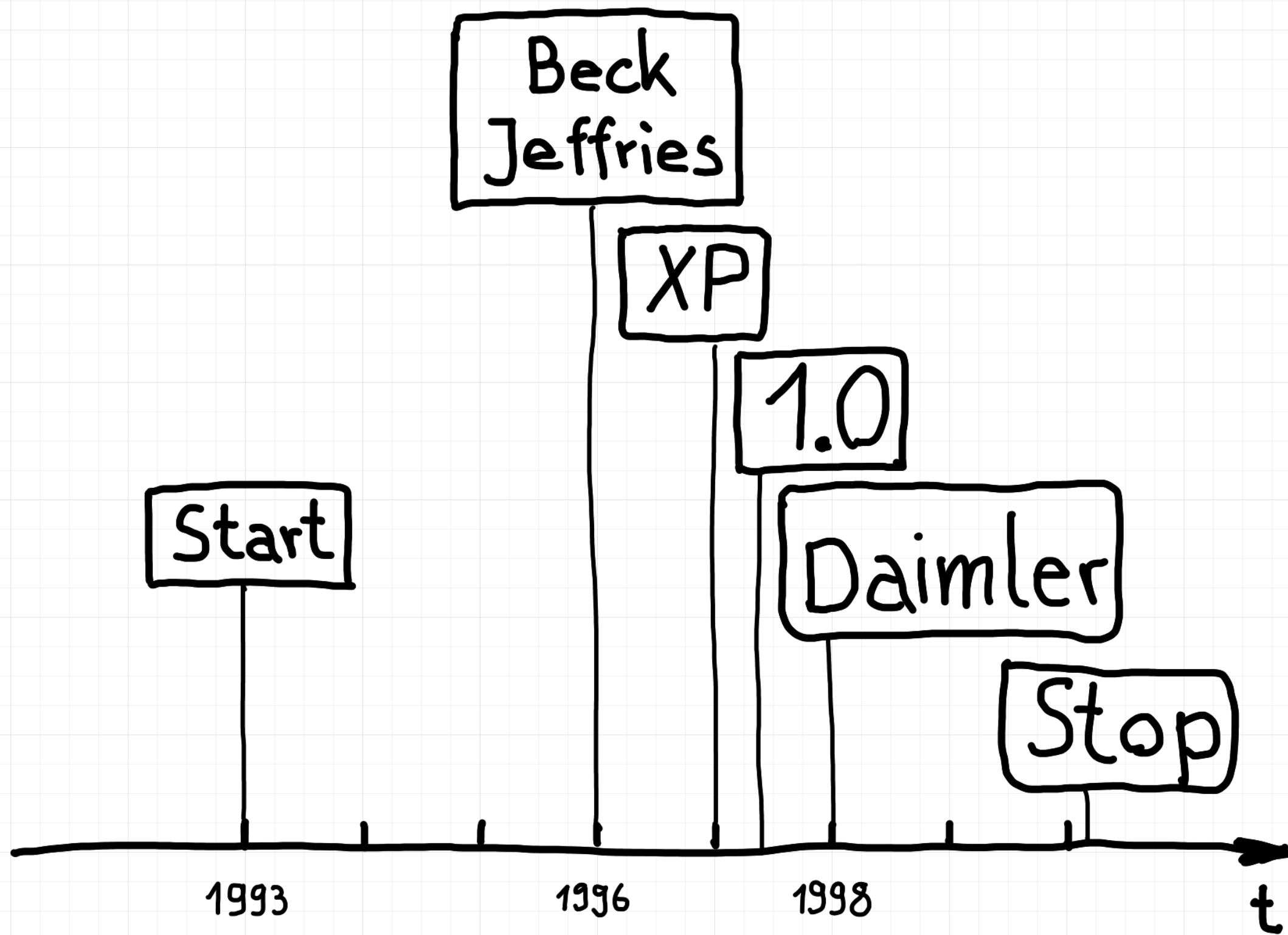












Regeln des Extreme Programming (Auszug)

Coding (Auszug)

- Unit-Test zuerst
- Pair-Programming
- Integriere oft
- Dedizierter Integrationscomputer

Testing (Auszug)

- Aller Code muss Unit-Tests haben
- Aller Code muss die Tests passieren vor den Release
- Bug -> Tests

Regeln des Extreme Programming (Auszug)

Coding (Auszug)

- **Unit-Test zuerst**
- Pair-Programming
- Integriere oft
- Dedizierter Integrationscomputer

Testing (Auszug)

- Aller Code muss Unit-Tests haben
- Aller Code muss die Tests passieren vor den Release
- Bug -> Tests



The Rules of Extreme Programming

Planning

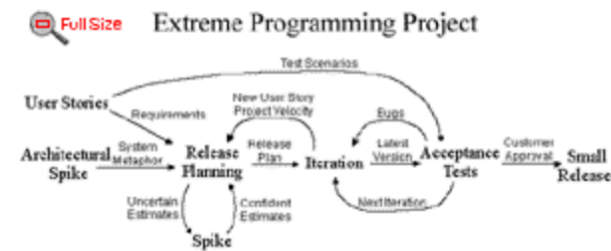
- User stories are written.
- Release planning creates the release schedule.
- Make frequent small releases.
- The project is divided into iterations.
- Iteration planning starts each iteration.

Managing

- Give the team a dedicated open work space.
- Set a sustainable pace.
- A stand up meeting starts each day.
- The Project Velocity is measured.
- Move people around.
- Fix XP when it breaks.

Designing

- Simplicity.
- Choose a system metaphor.
- Use CRC cards for design sessions.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.



Coding

- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.
- All production code is pair programmed.
- Only one pair integrates code at a time.
- Integrate often.
- Set up a dedicated integration computer.
- Use collective ownership.

Testing

- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.

Let's review the values of Extreme Programming (XP) next. ⌕ ↻

[ExtremeProgramming.org home](http://www.extremeprogramming.org) | [XP Map](#) | [XP Values](#) | [Test framework](#) | [About the Author](#)

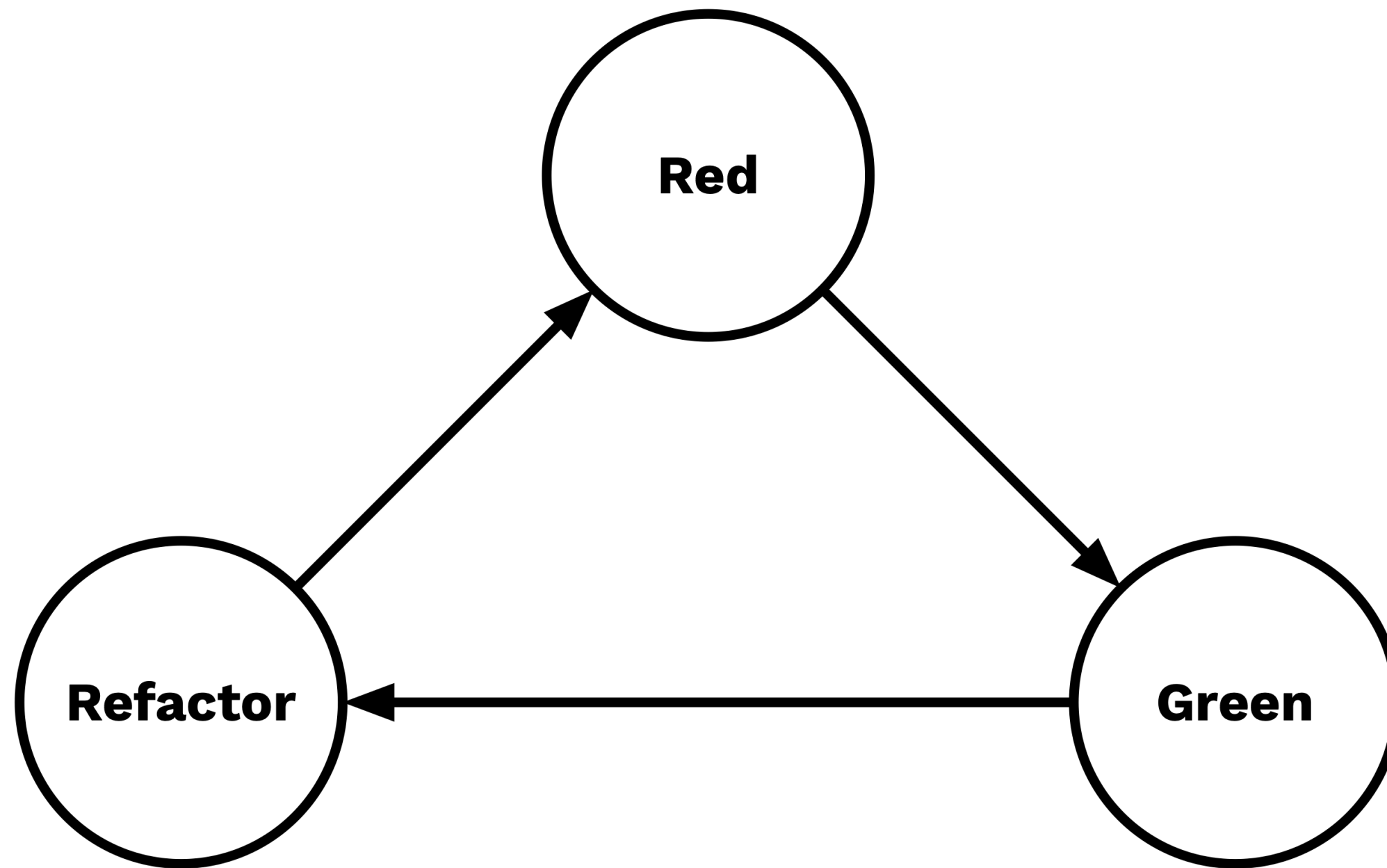
Copyright 1999 Don Wells all rights reserved

Quelle: <http://www.extremeprogramming.org/rules.html>

Nicht nur Sonnenschein

- Ein paar Monate nach Release kündigte der "Customer Representative" wegen Burnout und Stress.
- Nach Einstellung des C3-Projektes wurde XP (Extreme Programming) bei DaimlerChrysler verbannt.

Test Driven Development



Red

Schreibe einen Test, der fehlschlägt

- Kompilierfehler -> Testfehler
- Test muss **echt** fehlschlagen
- Sobald der Test fehlschlägt sollte Produktivcode geschrieben werden

Green

Schreibe gerade so viel Code, dass Test durchläuft

- Schreibe den simpelsten Code
- You Ain't Gonna Need It (YAGNI)

Refactor

Verbessere den Produktiv- und den Test-Code

- Test-Code ist auch Code
- Tests helfen dabei nichts kaputt zu machen

Simple Beispiel

Eine Wallet, in die Nutzerin einzahlen kann.

Simple Beispiel (Red)

```
@MainActor
struct WalletTests {
    let wallet: Wallet
}
```

Simple Beispiel (Red)

```
$> swift test
```

```
...
```

```
6 | @MainActor
```

```
7 | struct WalletTests {
```

```
8 |     let wallet: Wallet
```

```
    |         - error: cannot find type 'Wallet' in scope
```

```
9 | }
```

```
10 |
```

```
...
```

Simple Beispiel (Green)

```
class Wallet {}
```

Simple Beispiel (Green)

```
$> swift test
```

```
...
```

```
✔ Test run with 0 tests in 0 suites passed after 0.001 seconds.
```

Simple Beispiel (Red)

```
@MainActor
struct WalletTests {
    let wallet: Wallet

    init() {
        wallet = Wallet()
    }

    @Test
    func deposit_increasesBalance() {
        wallet.deposit(20)
    }
}
```

Simple Beispiel (Red)

```
$> swift test
```

```
...
```

```
14 | @Test
```

```
15 | func deposit_increasesBalance() {
```

```
16 |     wallet.deposit(20)
```

```
    |         - error: value of type 'Wallet' has no member 'deposit'
```

```
17 | }
```

```
18 | }
```

```
...
```

Simple Beispiel (Green)

```
class Wallet {  
    func deposit(_ value: Int) {  
  
    }  
}
```

Simple Beispiel (Green)

```
$> swift test
```

```
...
```

```
✔ Test run with 1 test in 1 suite passed after 0.001 seconds.
```

Simple Beispiel (Red)

```
@Test
func deposit_increasesBalance() {
    wallet.deposit(20)

    #expect(wallet.balance == 20)
}
```

Simple Beispiel (Red)

```
$> swift test
20 |     wallet.deposit(20)
21 |
22 |     #expect(wallet.balance == 20)
    |           ^- error: value of type 'Wallet' has no member 'balance'
23 | }
24 | }
```

Simple Beispiel (Red)

```
class Wallet {  
    var balance = 0  
  
    func deposit(_ value: Int) {  
  
    }  
}
```

Simple Beispiel (Red)

```
$> swift test
```

```
...
```

```
✖ Test deposit_increasesBalance() recorded an issue at WalletTests.swift:24:5:
```

```
Expectation failed: wallet.balance == 20
```

```
↳ wallet.balance == 20 → false
```

```
↳ wallet.balance → 0
```

```
...
```

Simple Beispiel (Green)

```
class Wallet {  
    var balance = 0  
  
    func deposit(_ value: Int) {  
        balance += value  
    }  
}
```

Simple Beispiel (Green)

```
$> swift test
```

```
...
```

```
✔ Test run with 1 test in 1 suite passed after 0.001 seconds.
```

Simple Beispiel (Refactor)

```
@Test
func deposit_increasesBalance() {
    let value = 20

    wallet.deposit(value)

    #expect(wallet.balance == value)
}
```

Simple Beispiel (Refactor)

```
$> swift test
```

```
...
```

```
✔ Test run with 1 test in 1 suite passed after 0.001 seconds.
```



Relevanz in 2026

- Agentic Codeing: Schreibe mindestens die Tests selbst!
- Schutz vor unerwarteten Änderungen.
- Bleibe handlungsfähig.
- YAGNI -> Code Coverage
- Besser: Schreibe auch den Code selbst! Aus Gründen...

SwiftUI

- In UIKit: keine Sorgen :)
- ViewModel: 😎
- Views: ViewInspector

Warum ist TDD entstanden?

**Warum ist TDD entstanden?
Wegen der Schmerzen!**

Der Elefant im Raum: AI

Wenn uns die AI nun alle Schmerzen und auch die Zeit nimmt, tief in etwas einzutauchen, werden wir dann weiterhin innovativ sein?

Werden wir weiter stolz sein auf unseren Code (um uns 4 Monate später dafür schämen)?

Ist Code nur Mittel zum Zweck?

Ist Code nur Mittel zum Zweck?

0x5f3759df

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;
    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y; // evil floating point bit level hacking
    i = 0x5f3759df - ( i >> 1 ); // what the fuck?
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
    return y;
}
```

Quelle: <https://www.heise.de/hintergrund/Zahlen-bitte-0x5f3759df-merkwuerdige-Mathematik-im-Egoshoooter-3208927.html>

Zurück zum Thema!

TDD - Vorteile

- Refaktorisierbarkeit (Agilität)
- Schnelles Feedback
- Dokumentation
- Gedankenstütze
- Regressionsschutz
- Doppelte Buchführung

Eigenschaften guter Unit-Tests²

- Sensitiv: Verhalten
- Insensitiv: Struktur
- Lesbar
- Schreibbar
- Schnell
- Deterministisch
- Automatisiert
- Isoliert
- Komponierbar
- Spezifisch
- Vorhersagend
- Inspirierend

² Quelle: <https://tidyfirst.substack.com/p/desirable-unit-tests>

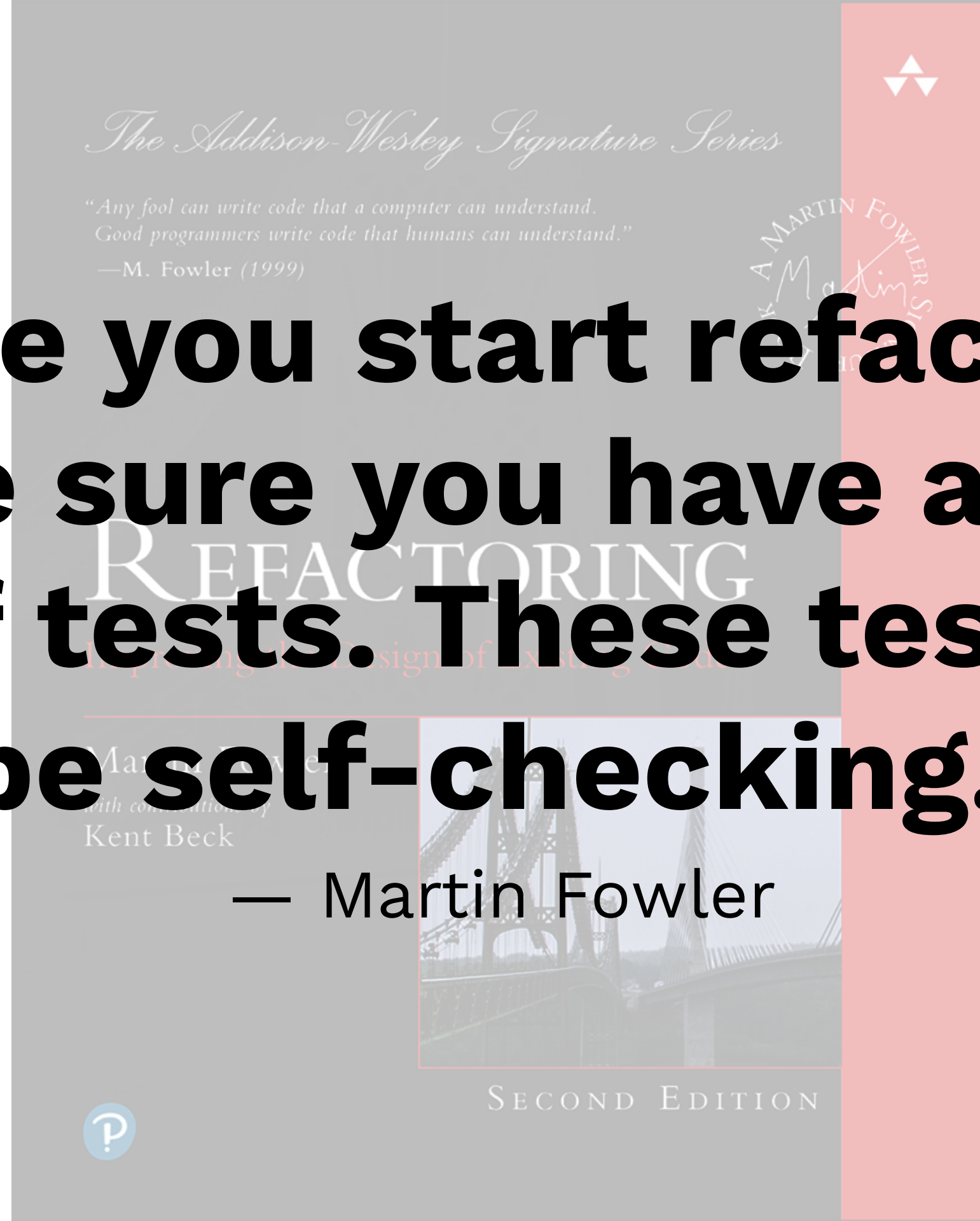
Test Doubles³

- **Dummy**: Platzhalter
- **Fake**: spielt dem Test Funktionen vor; nutzt Abkürzungen (In-Memory-DB...)
- **Stub**: liefert kontrollierte Antworten für den Test (API-Calls...)
- **Spy**: ein Stub, das auch Anfragen aufzeichnet
- **Mock**: vorprogrammiert mit Erwartungen

³ Quelle: <https://www.martinfowler.com/articles/mocksArentStubs.html>

**"Before you start refactoring,
make sure you have a solid
suite of tests. These tests must
be self-checking."**

— Martin Fowler

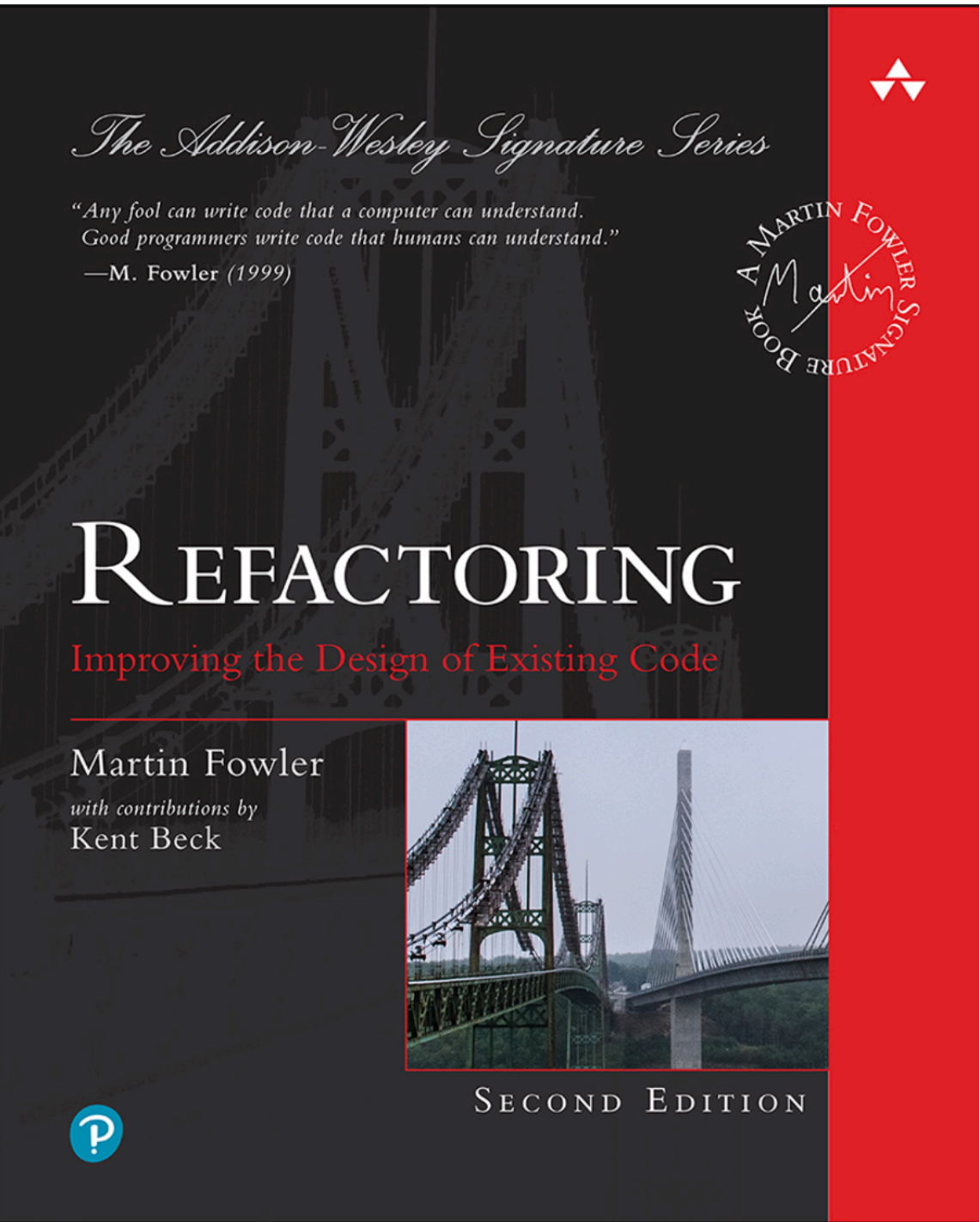


"Although it takes time to build the tests, I end up saving that time, with considerable interest, by spending less time debugging."

— Martin Fowler

Refactoring

List of Refactorings	
Change Function Declaration (124)	Pull Up Field (353)
Change Reference to Value (252)	Pull Up Method (350)
Change Value to Reference (256)	Push Down Field (361)
Collapse Hierarchy (380)	Push Down Method (359)
Combine Functions into Class (144)	Remove Dead Code (237)
Combine Functions into Transform (149)	Remove Flag Argument (314)
Consolidate Conditional Expression (263)	Remove Middle Man (192)
Decompose Conditional (260)	Remove Setting Method (331)
Encapsulate Collection (170)	Remove Subclass (369)
Encapsulate Record (162)	Rename Field (244)
Encapsulate Variable (132)	Rename Variable (137)
Extract Class (182)	Replace Command with Function (344)
Extract Function (106)	Replace Conditional with Polymorphism (272)
Extract Superclass (375)	Replace Constructor with Factory Function (334)
Extract Variable (119)	Replace Derived Variable with Query (248)
Hide Delegate (189)	Replace Function with Command (337)
Inline Class (186)	Replace Inline Code with Function Call (222)
Inline Function (115)	Replace Loop with Pipeline (231)
Inline Variable (123)	Replace Nested Conditional with Guard Clauses (266)
Introduce Assertion (302)	Replace Parameter with Query (324)
Introduce Parameter Object (140)	Replace Primitive with Object (174)
Introduce Special Case (289)	Replace Query with Parameter (327)
Move Field (207)	Replace Subclass with Delegate (381)
Move Function (198)	Replace Superclass with Delegate (399)
Move Statements into Function (213)	Replace Temp with Query (178)
Move Statements to Callers (217)	Replace Type Code with Subclasses (362)
Parameterize Function (310)	Separate Query from Modifier (306)
Preserve Whole Object (319)	Slide Statements (223)
Pull Up Constructor Body (355)	Split Loop (227)
	Split Phase (154)
	Split Variable (240)
	Substitute Algorithm (195)



Wie starten?

- Neues Feature
- Bug -> Test
- Dokumentation
- Einfach anfangen :)

Changes



Quelle: <https://x.com/KentBeck/status/250733358307500032>

test && commit || revert⁴

Oddmund Strømme, the first programmer I've found as obsessed with symmetry as I am, suggested that if the tests failed the code should be reverted. I hated the idea so I had to try it.

— Kent Beck

⁴ Quelle: https://medium.com/@kentbeck_7670/test-commit-revert-870bbd756864

Vielen Dank!

Fragen?