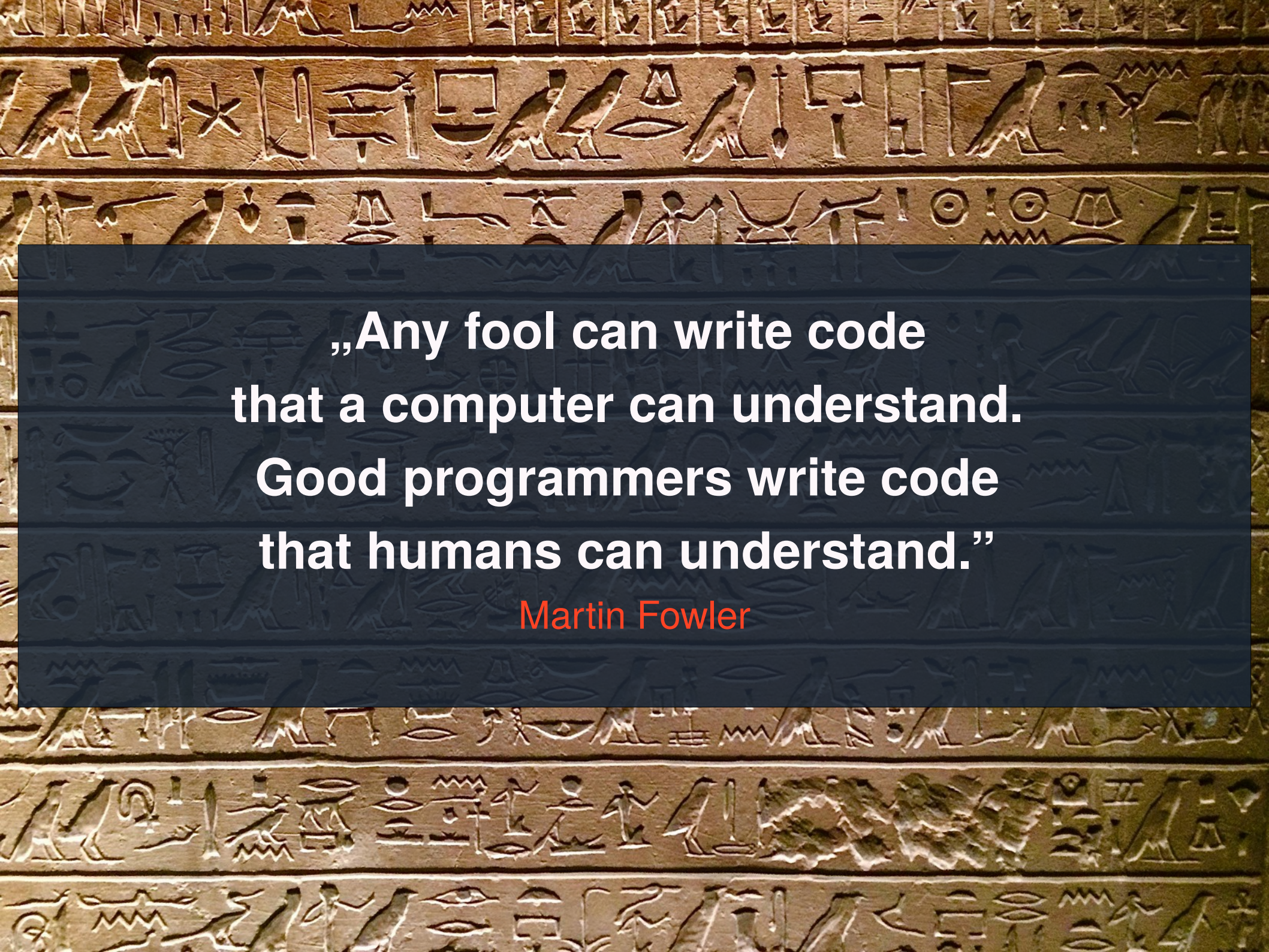
Clean Code mit Golang

Kristian Köhler

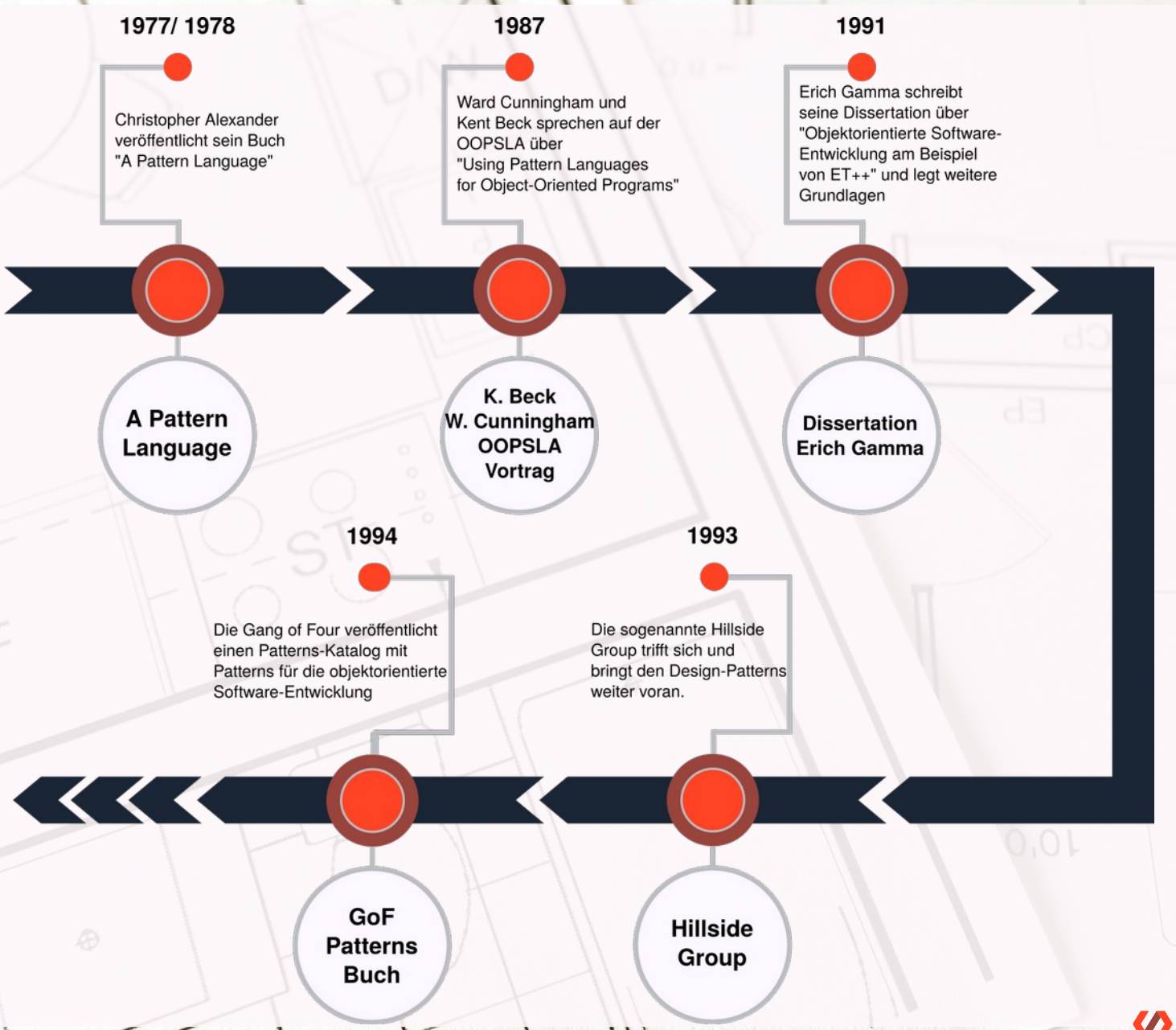
sourcefellows


„Any fool can write code
that a computer can understand.
Good programmers write code
that humans can understand.”
Martin Fowler

Verständnis macht Software wartbar


„The goal of software architecture
is to minimize the human ressources
required to build
and maintain required systems.“
Robert C. Martin / Uncle Bob

Software Design-Patterns - Entstehung



Clean Code and Clean Architecture

- **Vorgehen um „saubere“ Software zu erstellen**

  Software soll stabiler und besser wartbar werden

  Bücher von Robert C. Martin prägen Begriffe

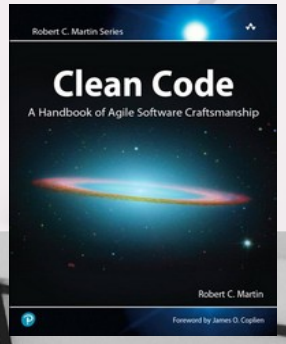
- **Clean Code**

  Vorgehen im Code (Formatierung, Kommentare, Namen, ...)

- **Clean Architecture**

  Strukturierung von Anwendungen

  Design principles / Architecture



Wer ich bin

Kristian Köhler

Source Fellows GmbH

https://www.source-fellows.com

https://www.linkedin.com/in/kristian-köhler/

20+ Jahre in Softwareentwicklung

Java Enterprise Hintergrund

Javascript, Python, C#, etc etc

Golang – Brutalismus in Code?

„Reaktion auf die Tendenz zur Über-Verfeinerung und den trockenen akademisch-abstrakten Geometrien, die sich im International Style verbargen.“ Anthony Vidler

„kompakte, disziplinierte Architektur“ Alison und Peter Smithson

#Funktional

#Gradlinig

#SichtbareStrukturen

#SimpleForm

sourcefellows

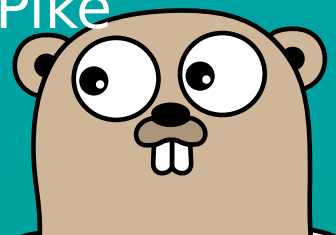
Programmiersprache Go

- **Entstanden ab 2007 bei Google**
  - Robert Griesemer, Rob Pike und Ken Thompson
  - Seit 2009 als OpenSource verfügbar

https://go.dev/

- **Entwickelt für “heutige” Anwendungen**
  - Nebenläufig, Netzwerk, ...

- **Einfach, Performant, Effizient, Sicher**
  - „Go was designed by and for people who write—and read and debug and maintain—large software systems.“ - Rob Pike



Keep it simple and stupid!

KISS and Go

- **Einfacher Syntax / Sprachstabilität**

  Sprache ist „langweilig“

- **Umfangreiche Standardbibliothek und Tooling**

  nicht sofort externe Bibliothek und Tools nötig

- **Keine komponentenbasierte Entwicklung**

  Entscheidung was genutzt wird

  minimale Abhängigkeiten

  Frameworks sind Bibliotheken

- **Keine „Magic“ mit Annotations**

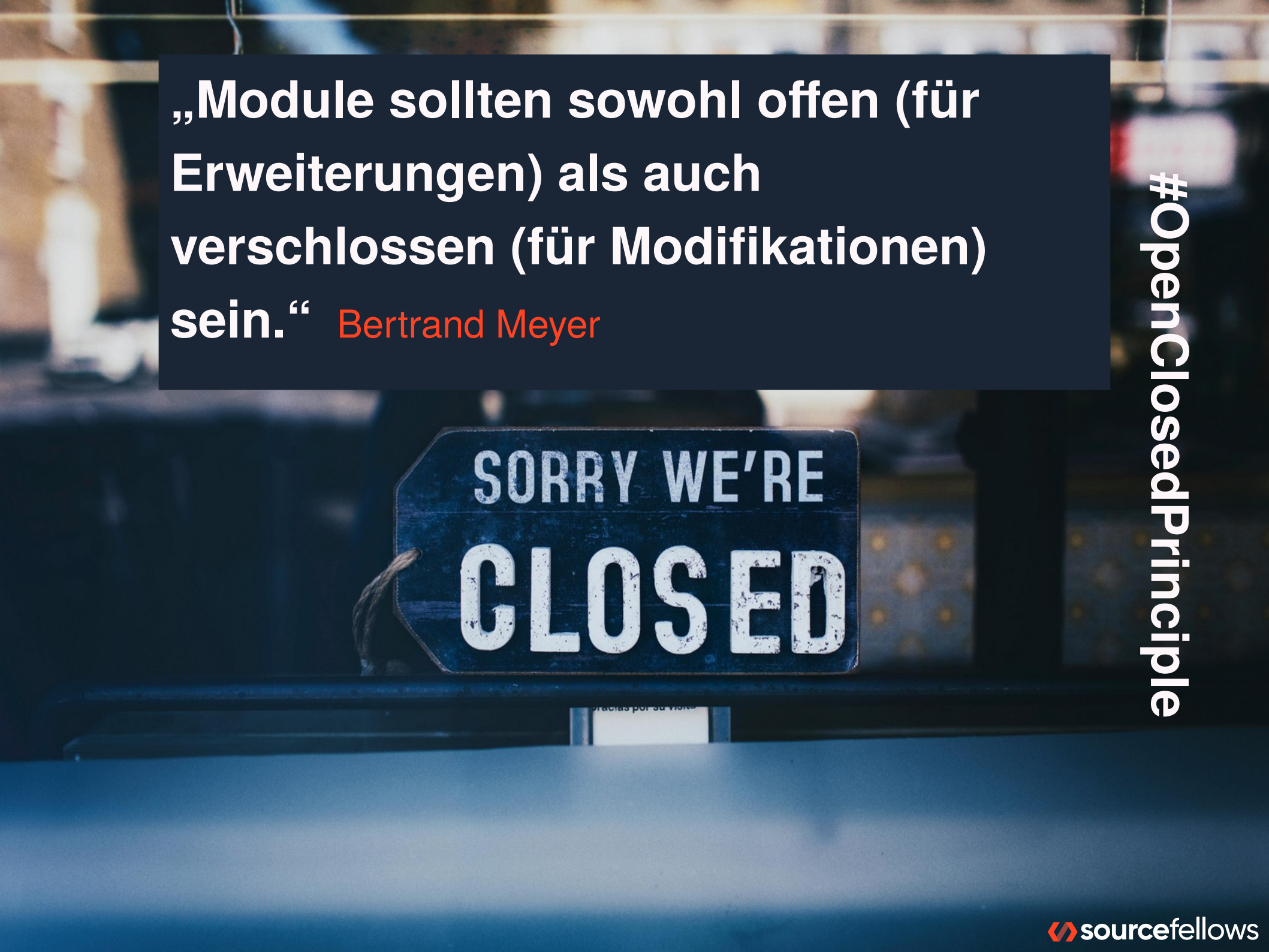
SOLID principles

- **S**ingle Responsibility
- **O**pen/Closed
- **L**iskov subsititution
- **I**nterface segregation
- **D**ependency inversion

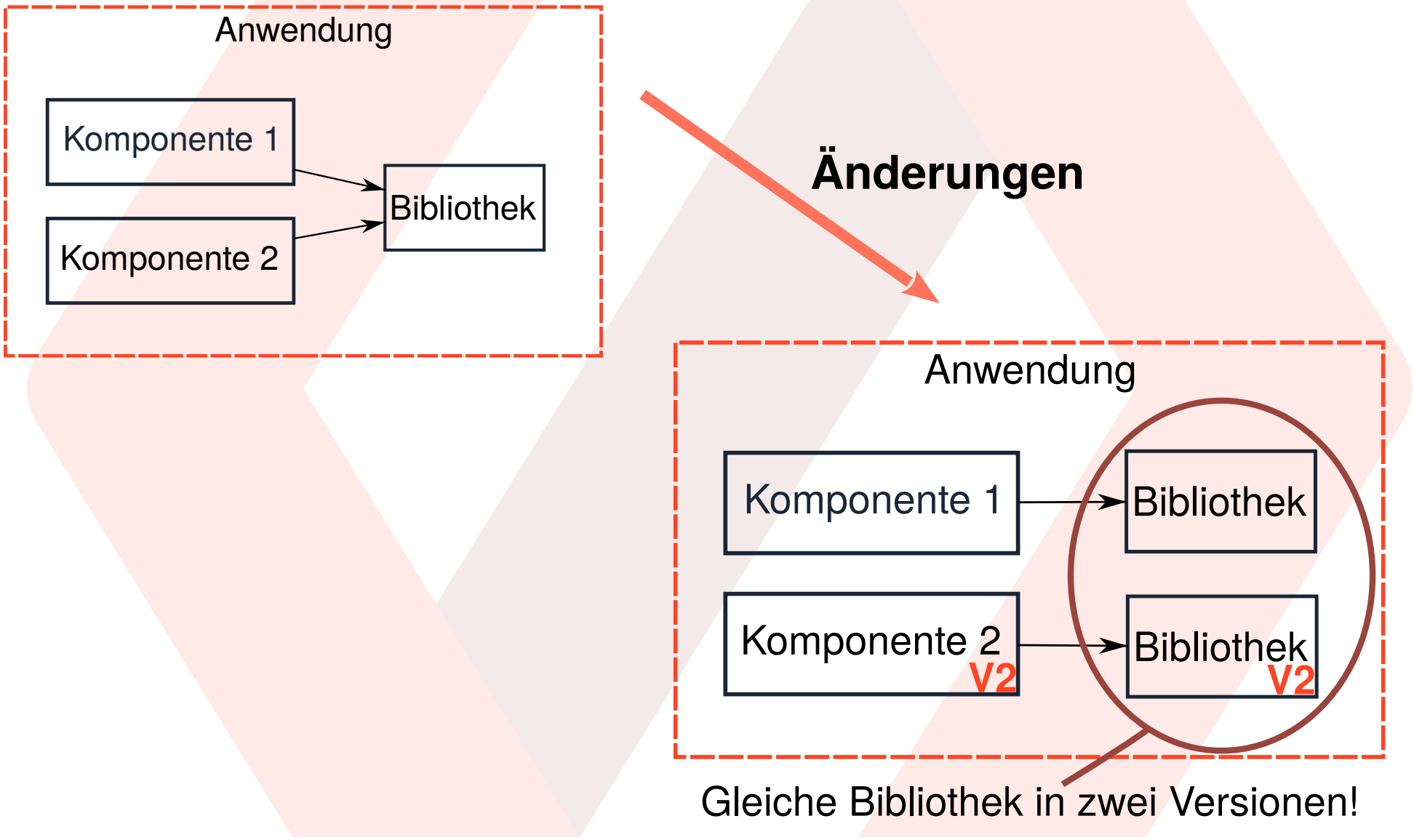
Single Responsibility

„Es sollte nie mehr als einen Grund geben, eine Klasse zu ändern.“
Robert C. Martin (Uncle Bob)

“Gather together the things that change for the same reasons. Separate those things that change for different reasons.” Robert C. Martin (Uncle Bob)


„Module sollten sowohl offen (für Erweiterungen) als auch verschlossen (für Modifikationen) sein.“ Bertrand Meyer
SORRY WE'RE
CLOSED
#OpenClosedPrinciple
sourcefellows

Kompatibilitätsversprechen



Interface segregation principle

#RoleInterface

“Clients should not be forced to depend upon interfaces that they do not use.”

Robert C. Martin

Vielen Dank für Ihre Aufmerksamkeit!

sourcefellows