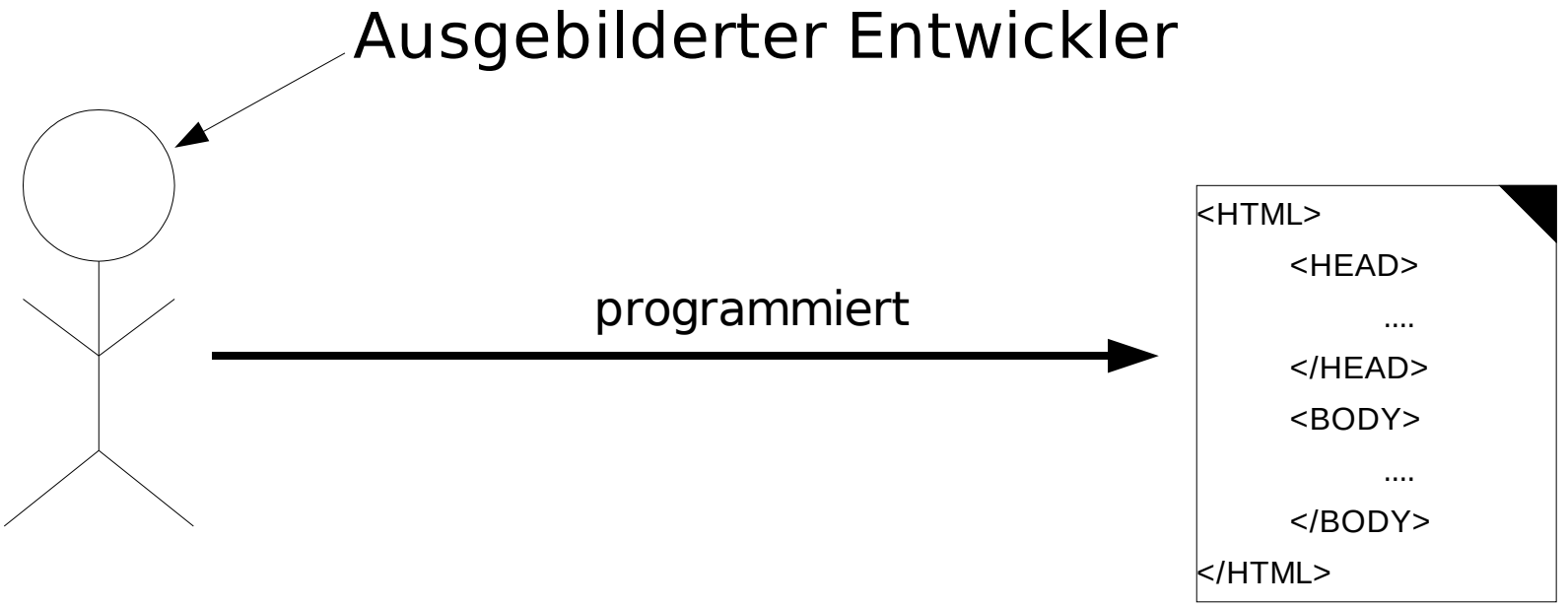
Existierende Systeme II: Generierende Ansätze

Jens Siebert
PG-WaVis - Universität Paderborn

Motivation

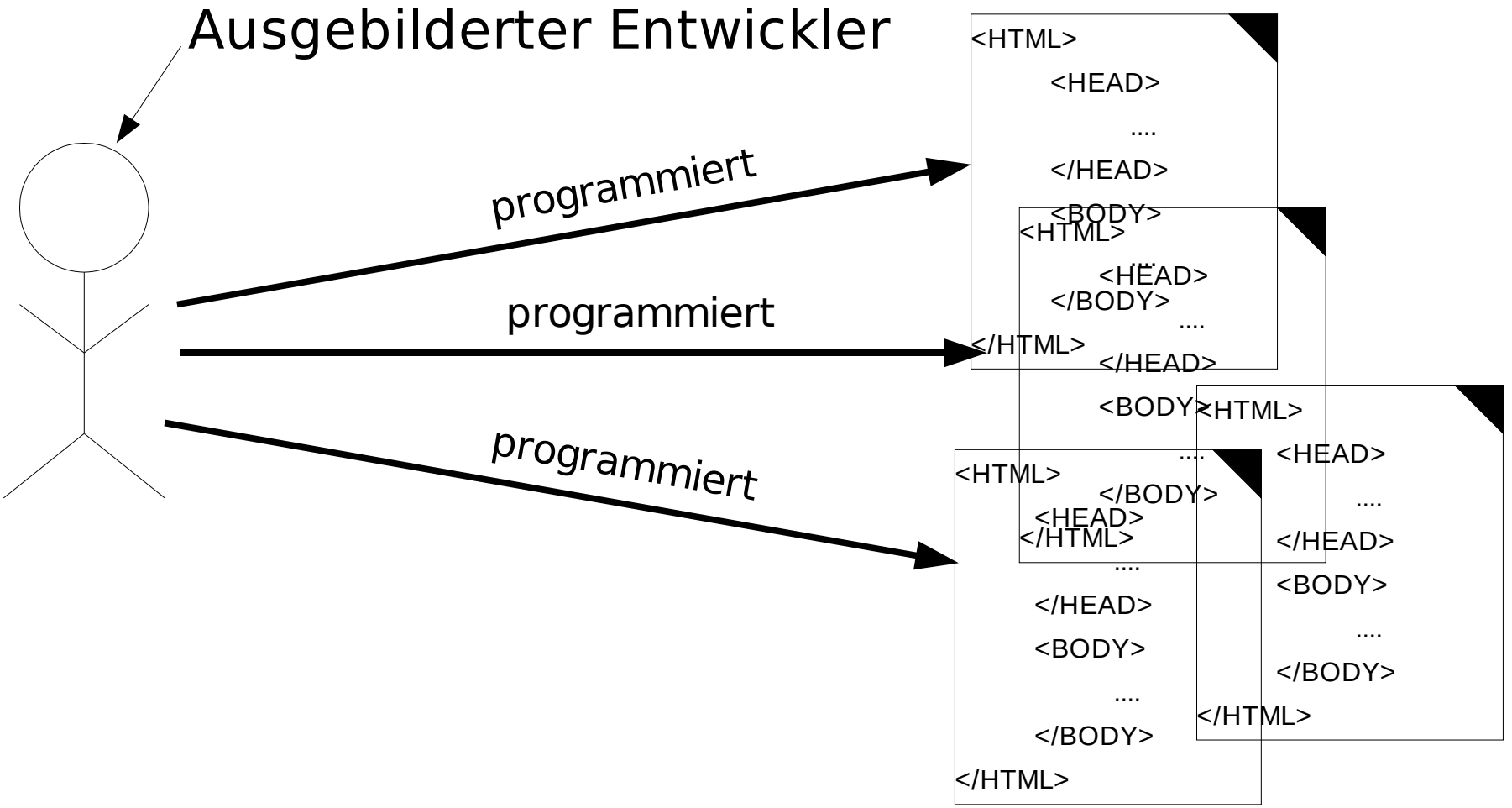
1. Warum überhaupt Web-Anwendungen generieren?
2. Wie generiert man Web-Anwendungen?
3. Welche Systeme existieren bereits?

Generierung von Web-Anwendungen - Warum?



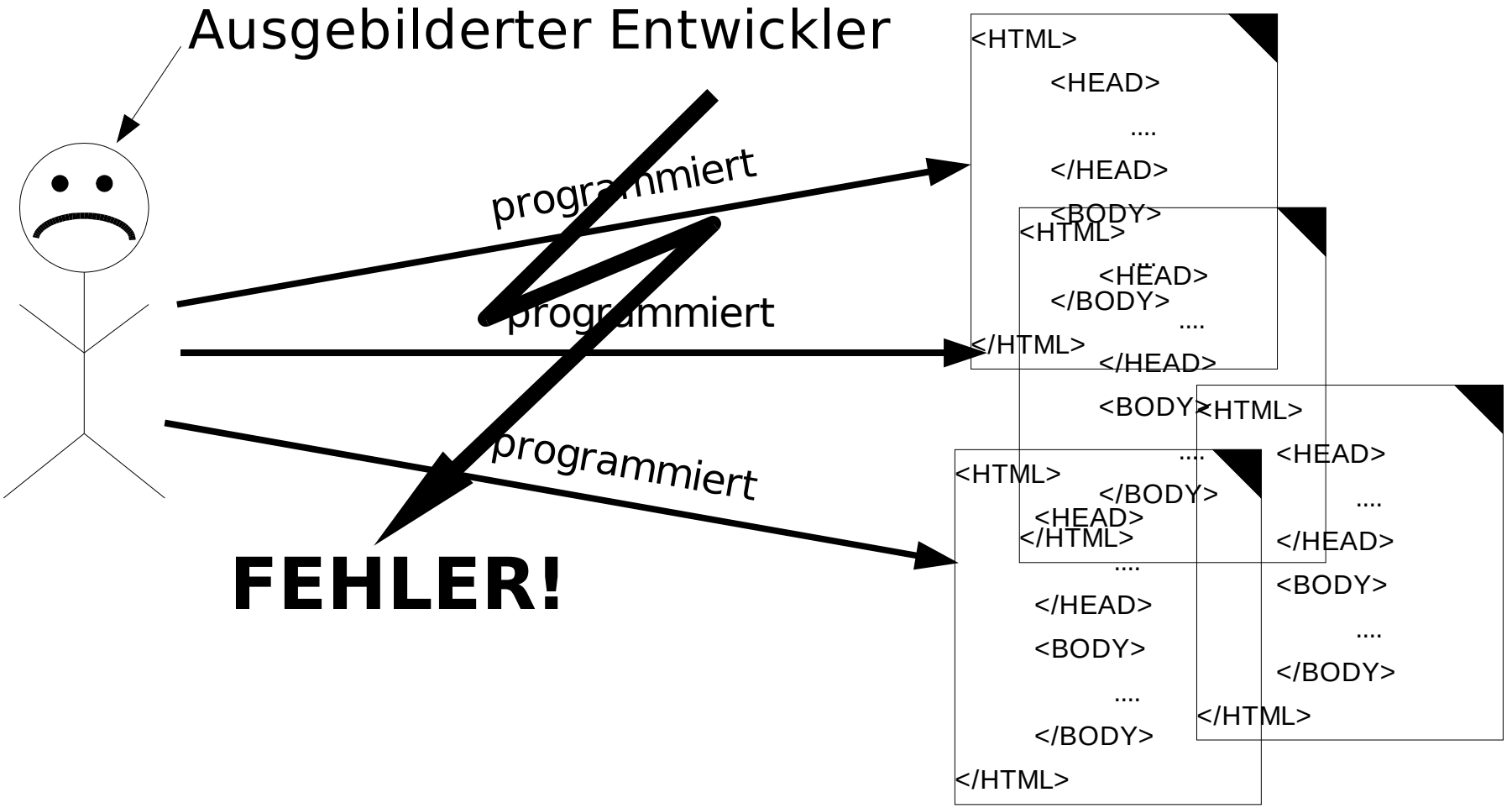
→ Für „kleine“ Projekte Generierung nicht nötig

Generierung von Web-Anwendungen - Warum?



→ „kleine“ Projekte können schnell komplex werden

Generierung von Web-Anwendungen - Warum?



→ Generierung von Web-Anwendungen ist gut....

Generierung von Web-Anwendungen - Warum?

, denn:

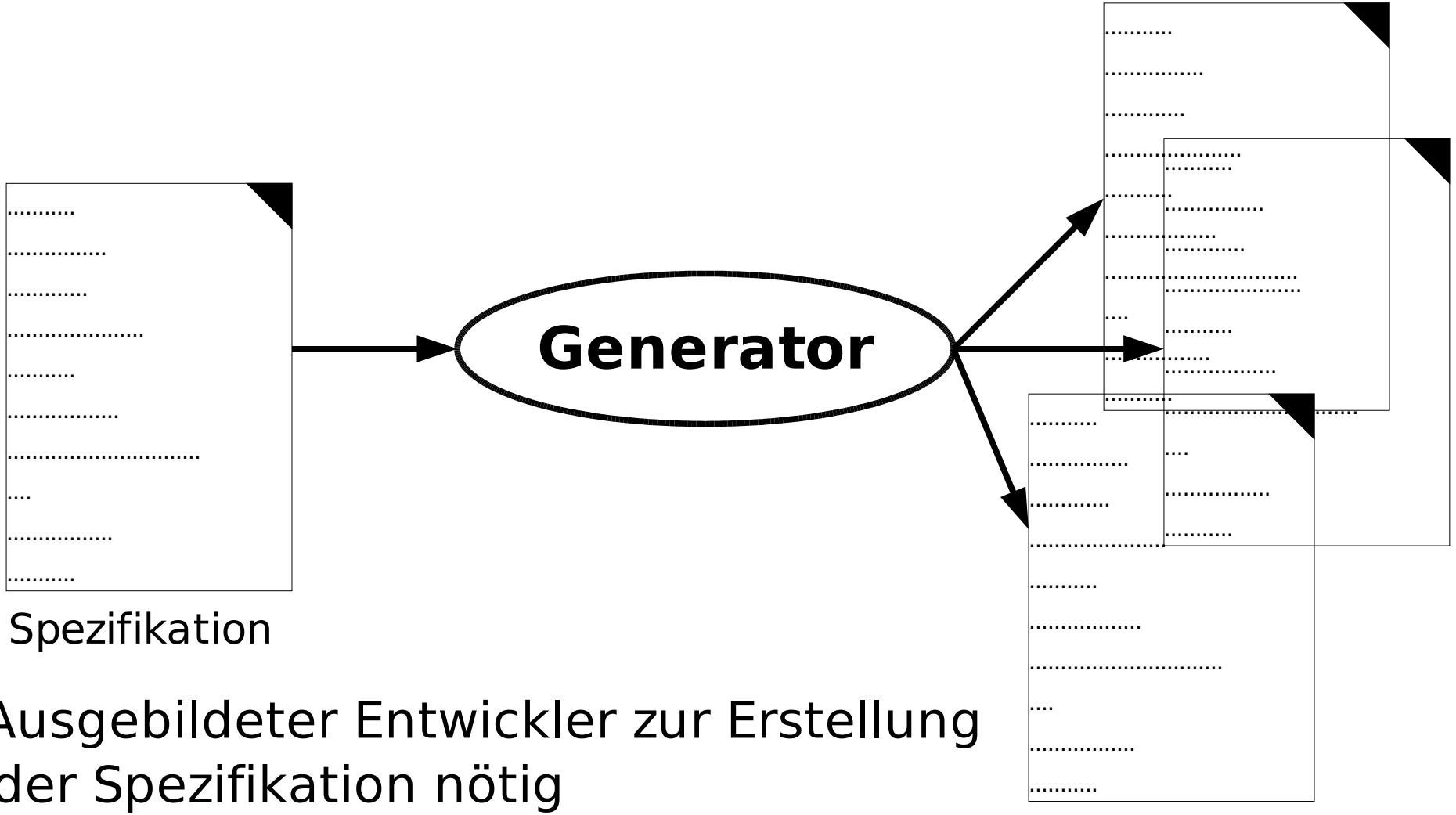
- Reduzierung der Komplexität

- Einfachere Wartbarkeit

- Generierung weniger fehlerbehaftet

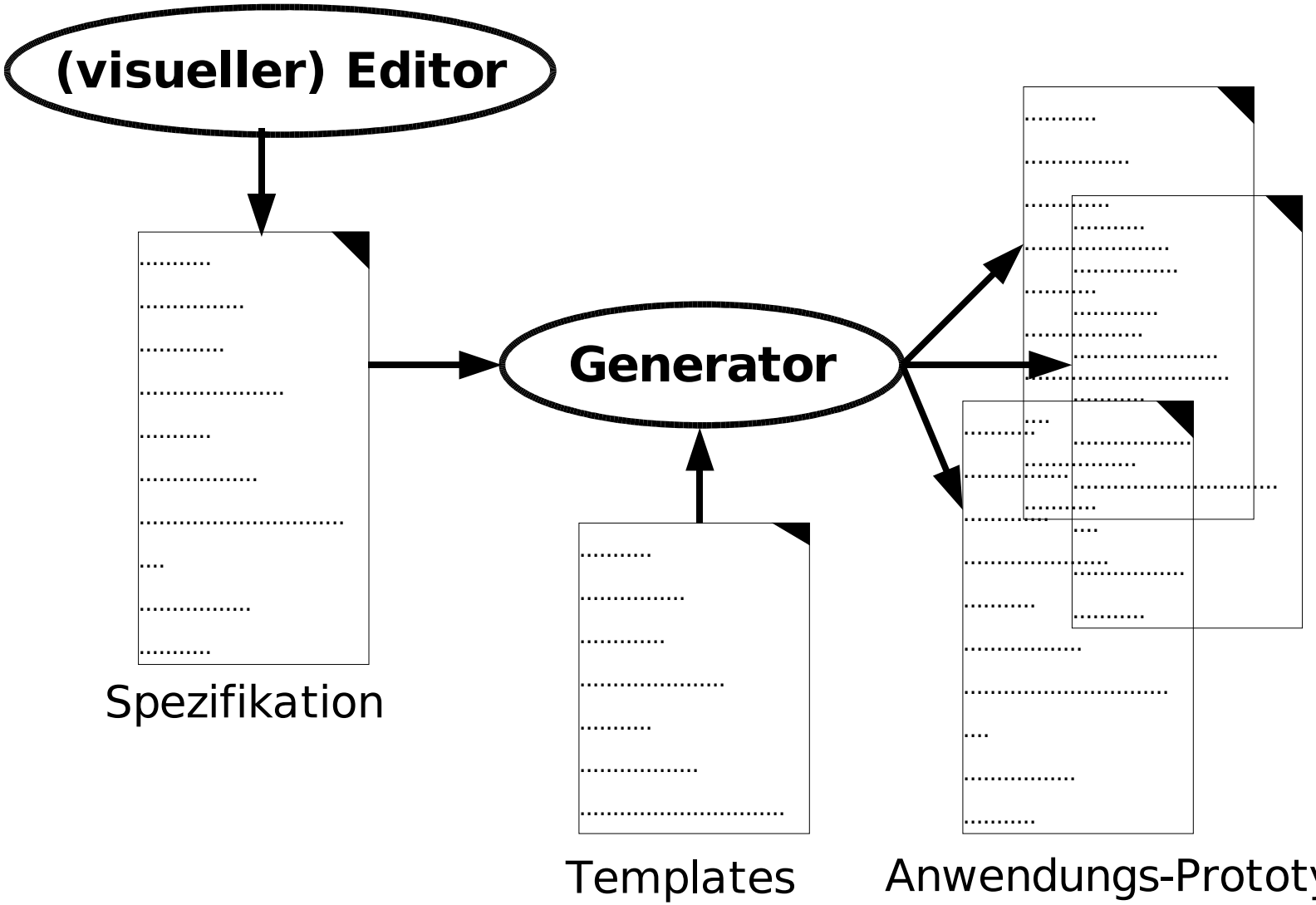
- Entwicklung von Anwendungen durch „nicht-Entwickler“

Generierung von Web-Anwendungen - Wie?



→ Ausgebildeter Entwickler zur Erstellung der Spezifikation nötig

Generierung von Web-Anwendungen - Wie?



Generierung von Web-Anwendungen - Wie?

- Erstellung der Spezifikation durch visuellen Editor
  - → weniger Komplexität, da abstrakter
  - → einfacher für „nicht-Entwickler“
  - → nahezu visuelle Darstellung der fertigen Anwendung

- Einbindung von Templates durch den Generator
  - → Trennung von Aufgaben (Entwickler/Designer)
  - → bessere Wartbarkeit

- Erstellung eines Anwendungs-Prototyps durch Generator
  - → schnelle Ergebnisse
  - → weniger Fehler
  - → kein Prototyp zum Wegwerfen

Existierende Systeme

- ProGUM-Web
  - → Entwickelt an der Universität Paderborn
  - → Verwendet UML als visuelle Sprache

- T-WEB-System
  - → Entwickelt am Tokyo Institute of Technology
  - → Verwendet Web Transition Diagrams

- BioPro-System
  - → Entwickelt an der Universität Tokushima
  - → Verwendet Bild-/Komponenten-basierten Ansatz

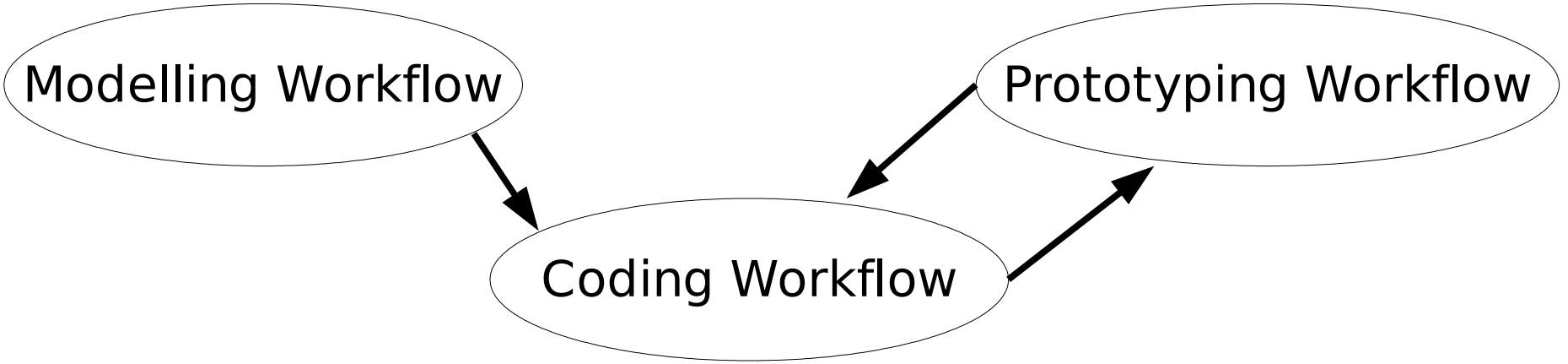
... und viele weitere Generatoren: www.codegeneration.net

ProGUM – Webanwendungen mit UML

- Modell-basierte Entwicklung

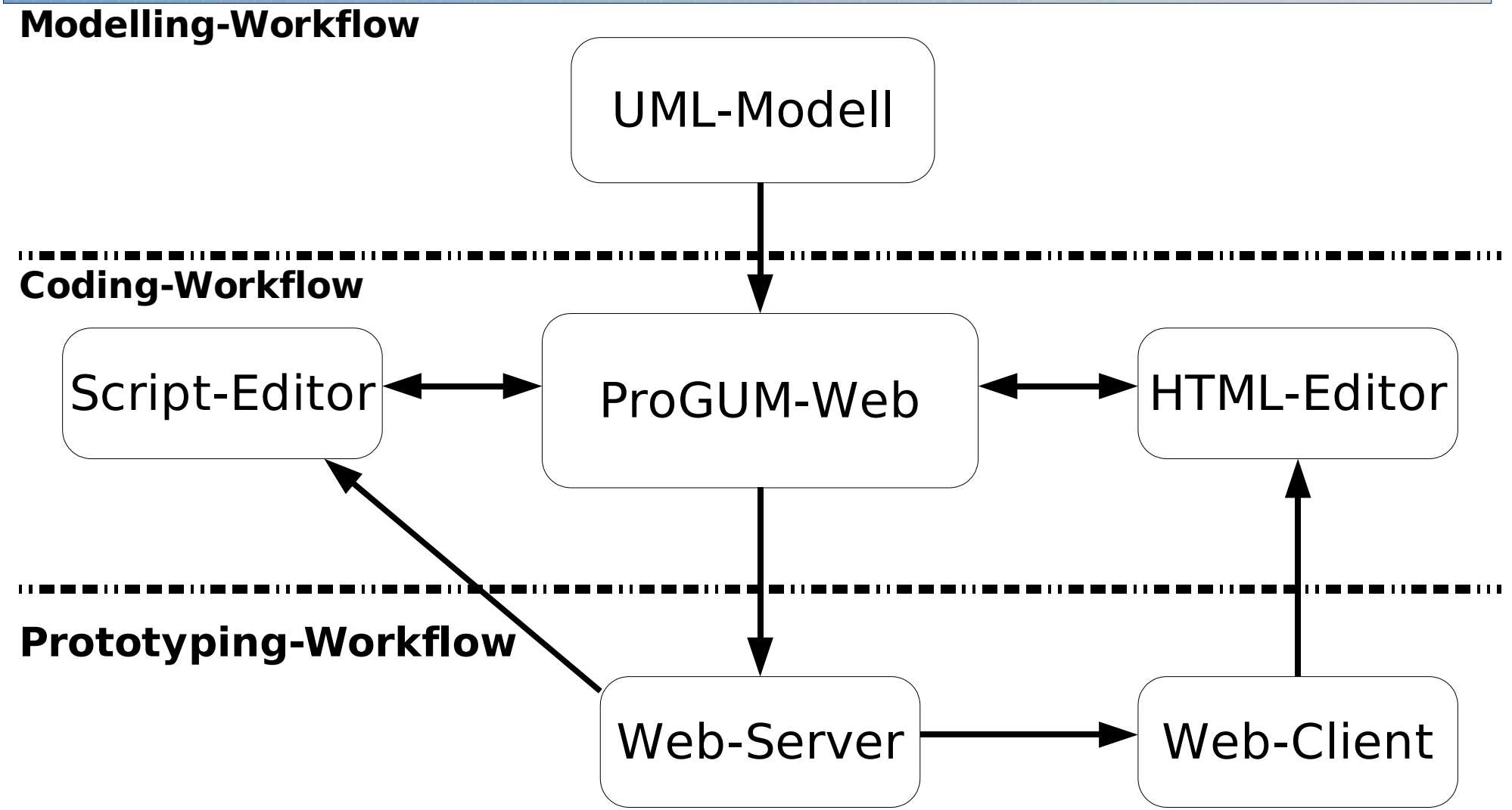
  → Struktur und Verhalten werden mit UML modelliert

- Workflow-basierter Entwicklungsprozess



- ProGUM dient als zentrales Repository

ProGUM - System

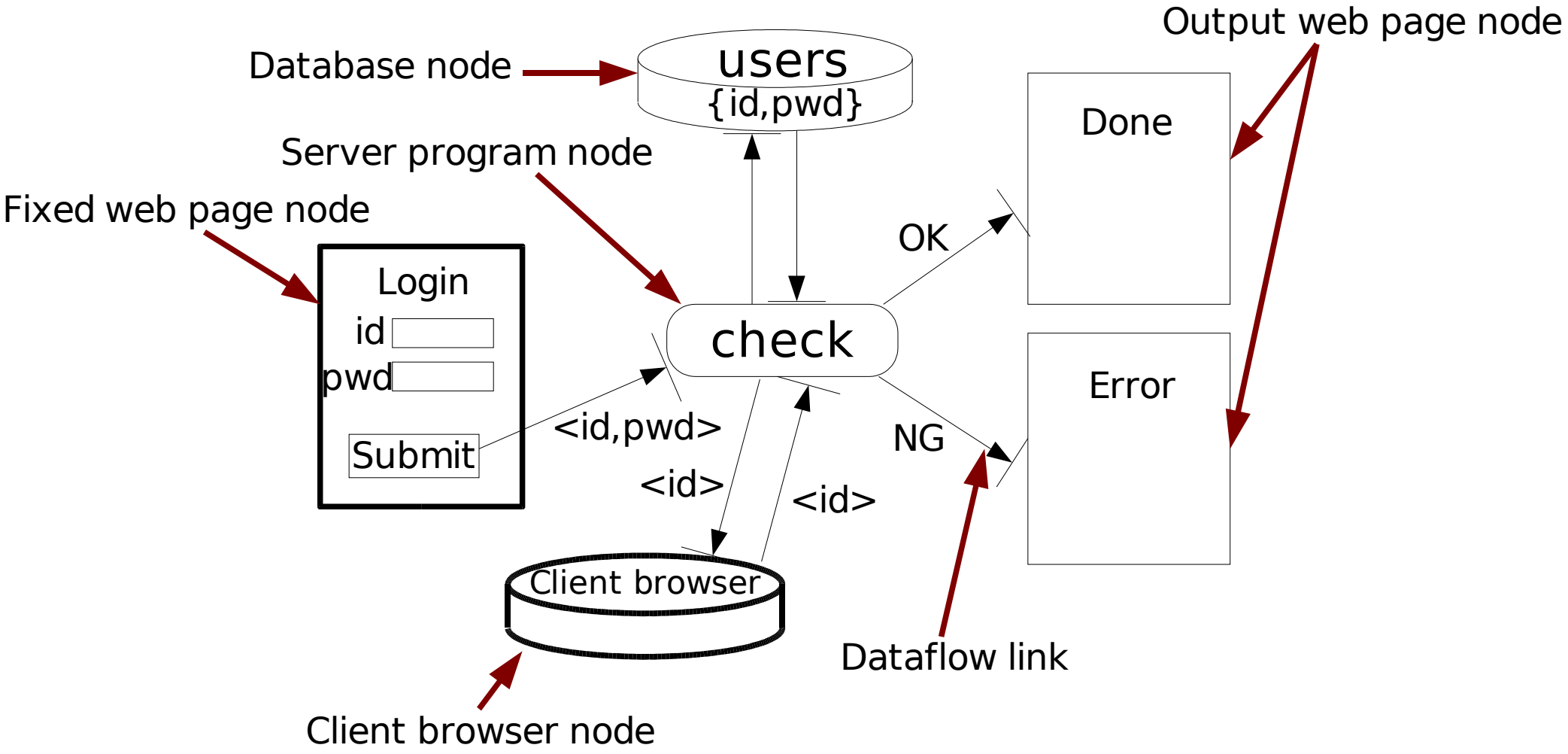


ProGUM - Überblick

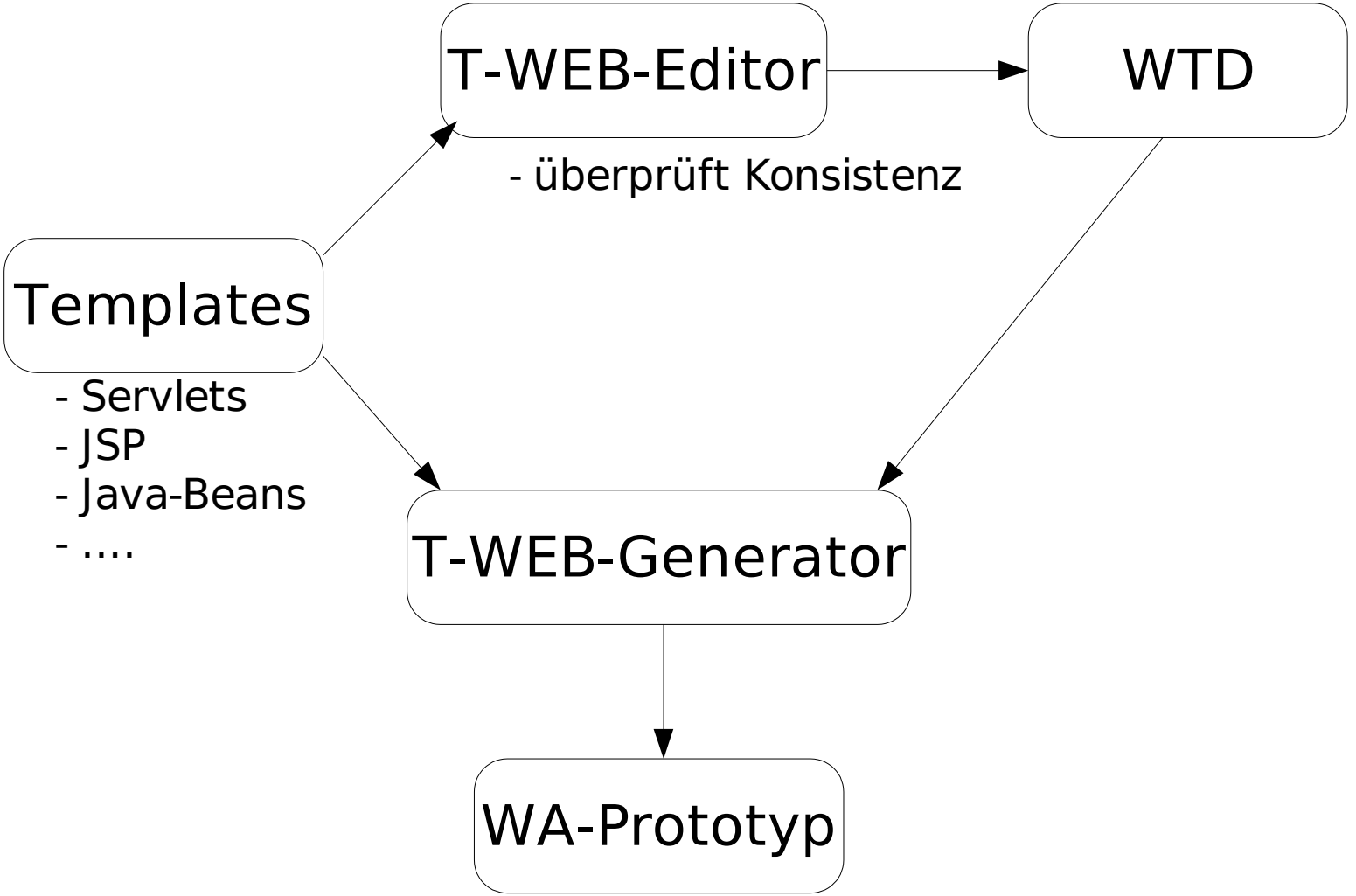
- UML-Methodologie nutzbar
  - → Modell-basierte Entwicklung
  - → Workflow-basierter Entwicklungsprozess

- Teamwork durch zentrales Repository

T-WEB – Web–Transition-Diagrams



T-WEB - System


T-WEB-Editor
WTD
- überprüft Konsistenz
Templates
- Servlets
- JSP
- Java-Beans
- ....
T-WEB-Generator
WA-Prototyp

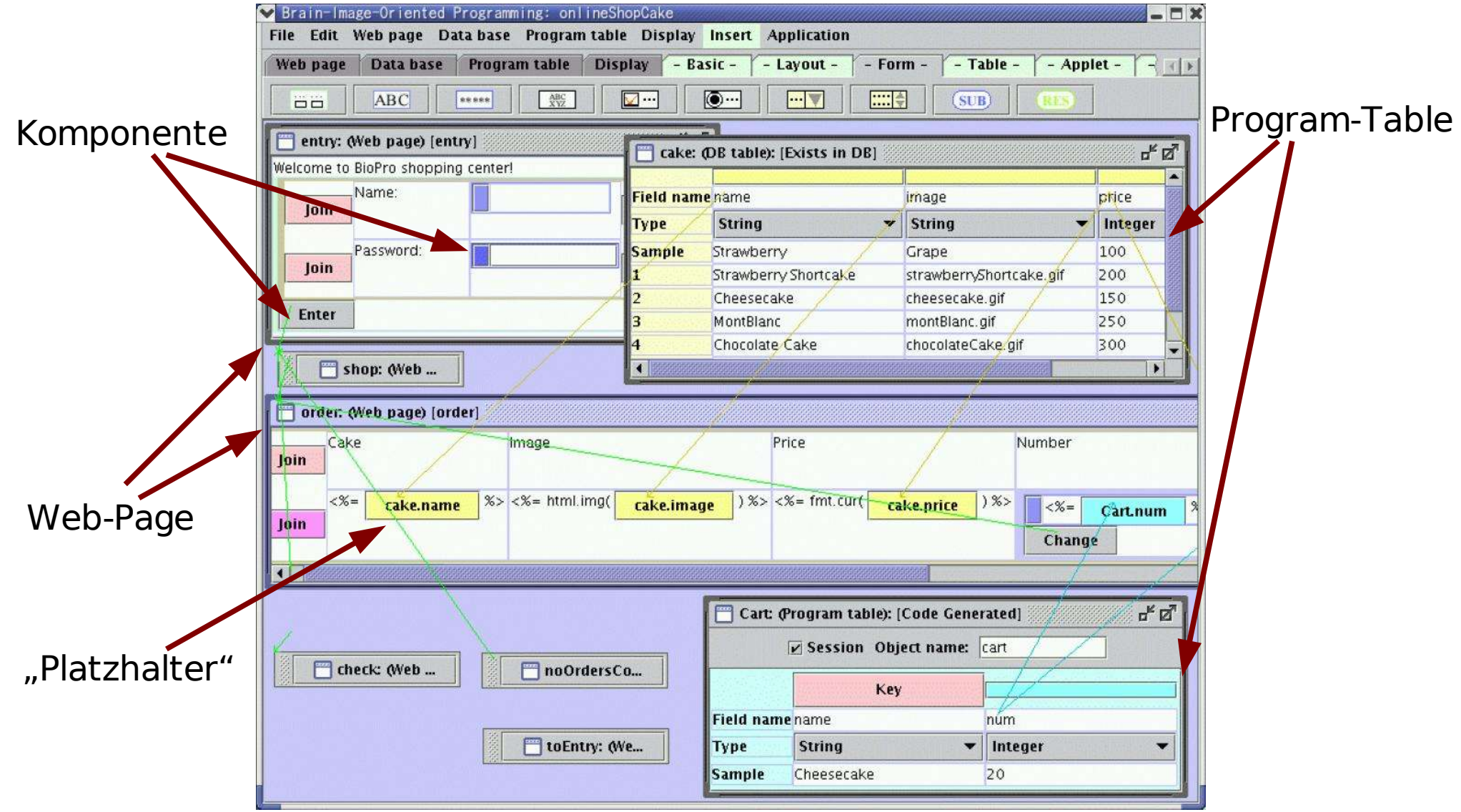
T – WEB - Überblick

- Modellierung durch Web-Transition-Diagrams
  - → Page-Flow und Data-Flow in einem Diagramm
  - → Bessere Übersicht über die Anwendung

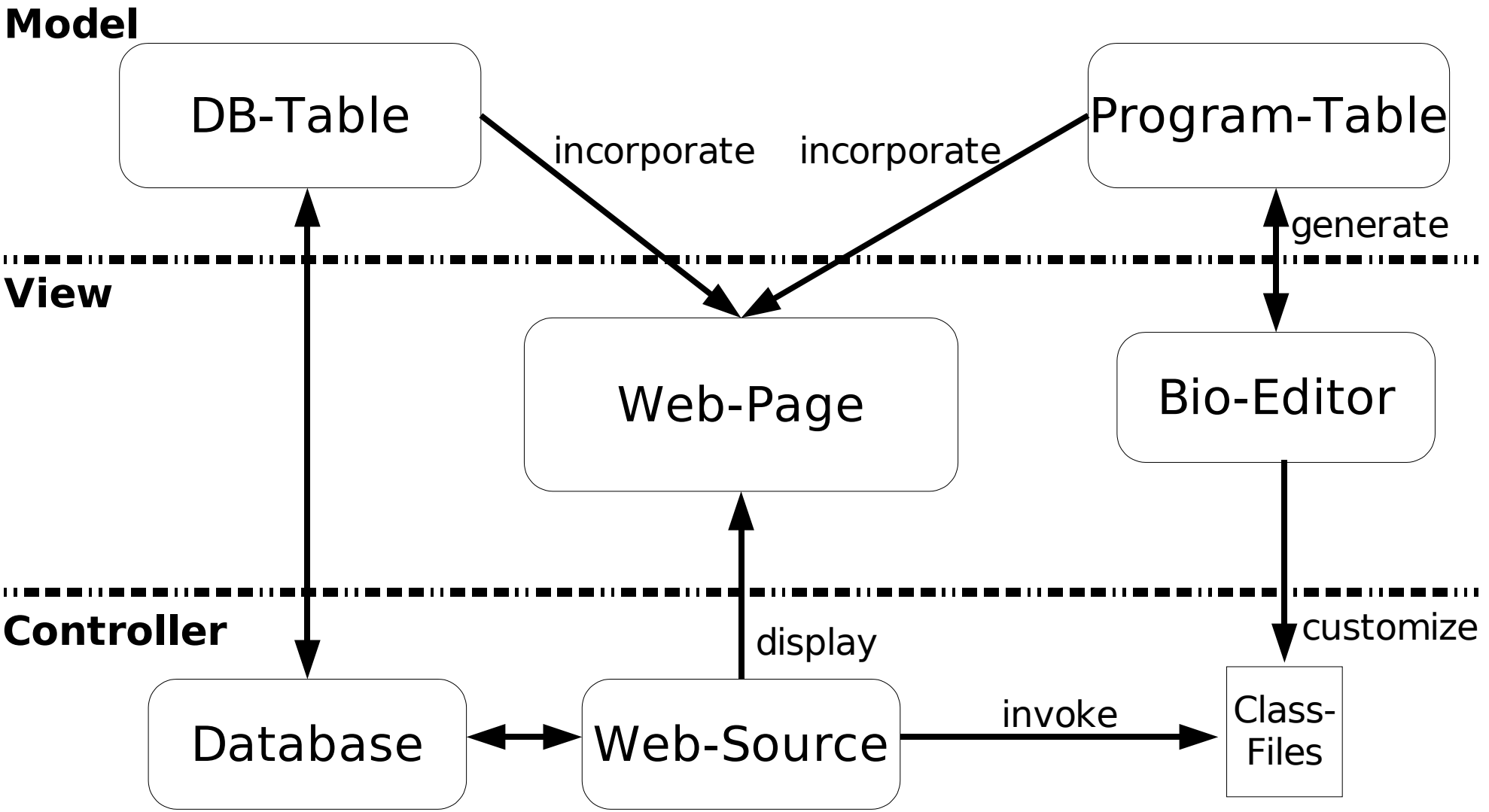
- Visueller Editor für WTDs
  - → Einfach zu bedienen (auch für „nicht-Entwickler“)
  - → Ständige Überprüfung der Konsistenz

- Generator erstellt aus WTDs und Templates WA-Prototyp
  - → Trennung von Anwendungsentwurf, Entwicklung und (Grafik-)Design

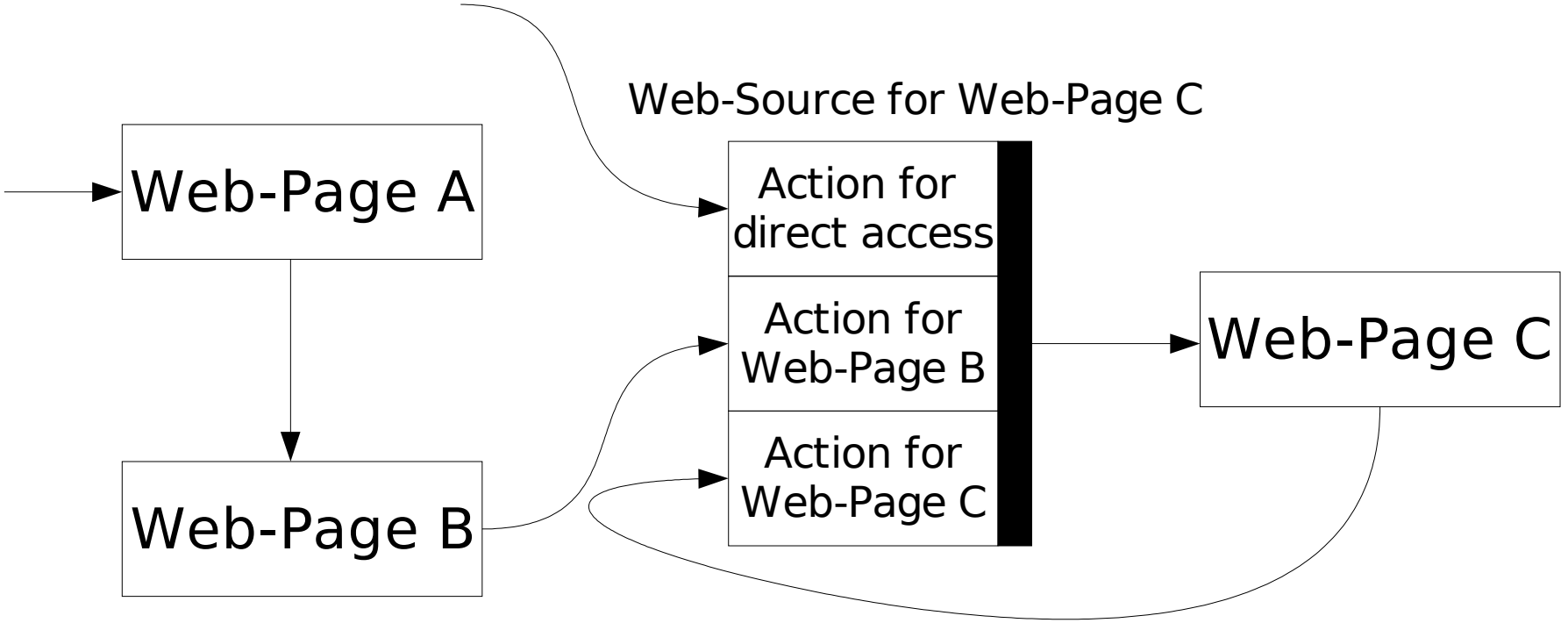
BioPro – Image based development



BioPro – MVC-Architektur



BioPro – Actions



BioPro – Überblick

- Bild- und Transitionsorientierter Ansatz
  - → Bildliche Darstellung der Web-Seite durch Komponenten
  - → Transitionen modellieren Page- und Data-Flow
- Datenmodellierung durch Programm-Tabellen
  - → Speicherung der Daten wie in DB-Tabellen
  - → Verknüpfung der Daten mit Web-Seiten durch „Platzhalter“
  - → „Join“ mit DB-Tabellen möglich
- MVC-basierte Architektur der Web-Anwendung
  - → Trennung von Code und Design
  - → Bessere Wartbarkeit
- Komponenten anpassbar

Zusammenfassung

- Es muss nicht immer UML sein

- Existierende Systeme liefern gute Ideen für PG
  - → für die visuelle Sprache
    (WTDs, image-based development)
  - → für die Funktionalität (Konsitenzchecks, Actions)
  - → für den Aufbau des Editors (T-WEB, BioPro)
  - → für den Generator (Templates, Repository)

Schluss

Löchern Sie den Vortragenden bitte JETZT mit Fragen ;-)