

Existierende Systeme II: Generierende Ansätze

Jens Siebert

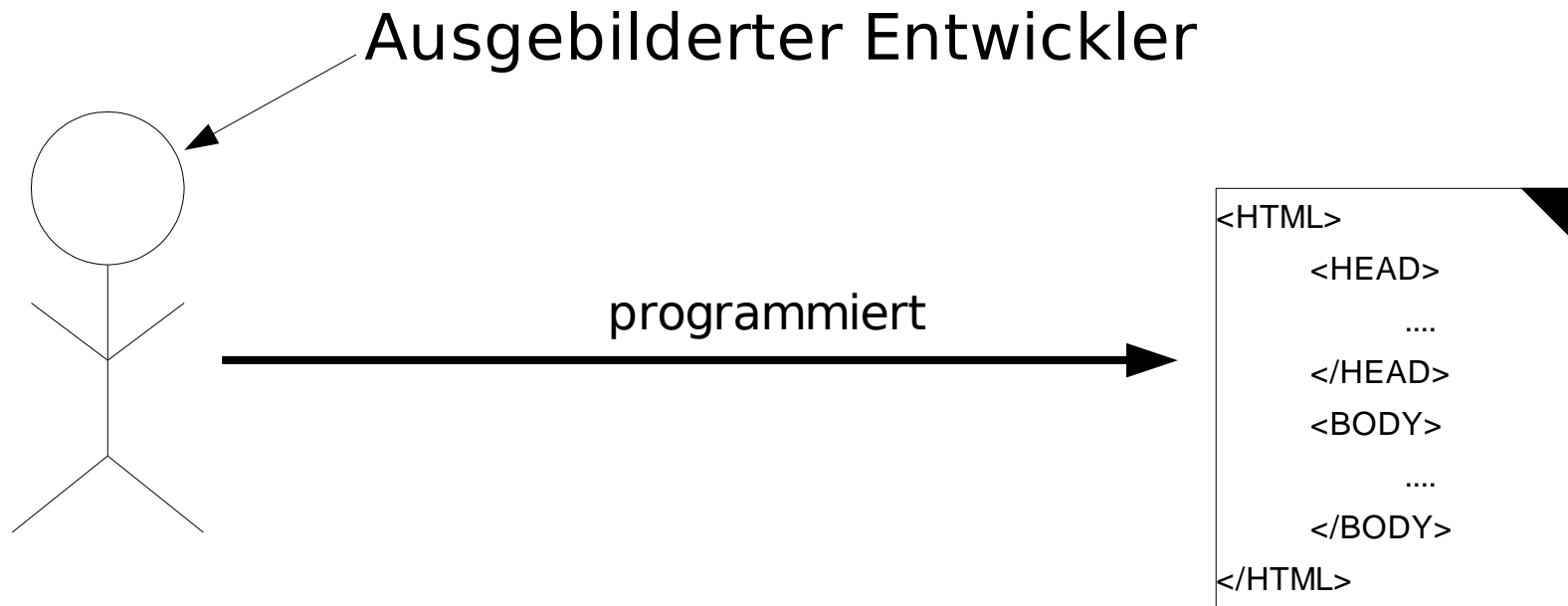
PG-WaVis - Universität Paderborn

Motivation

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

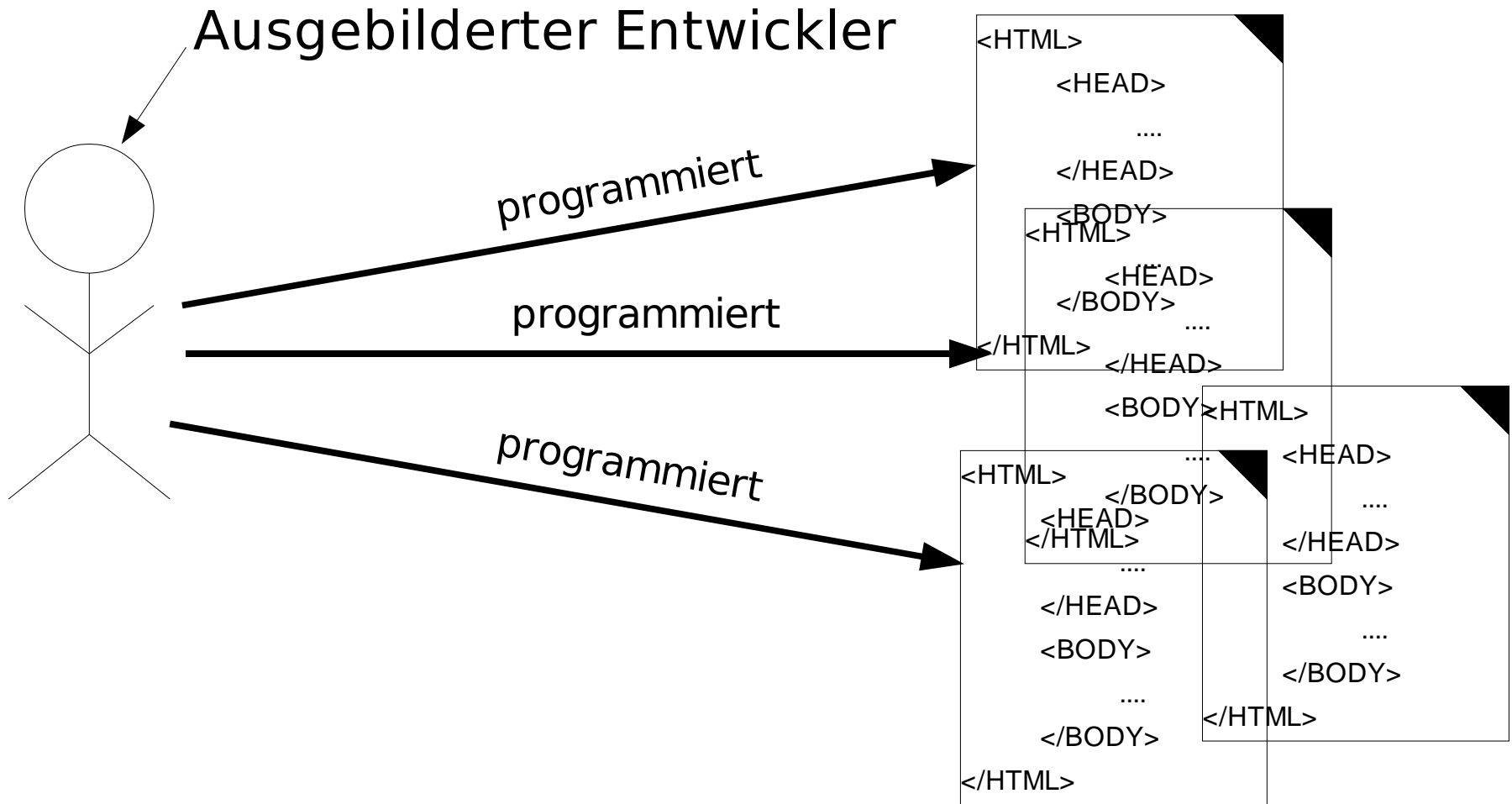


Generierung von Web-Anwendungen - Warum?



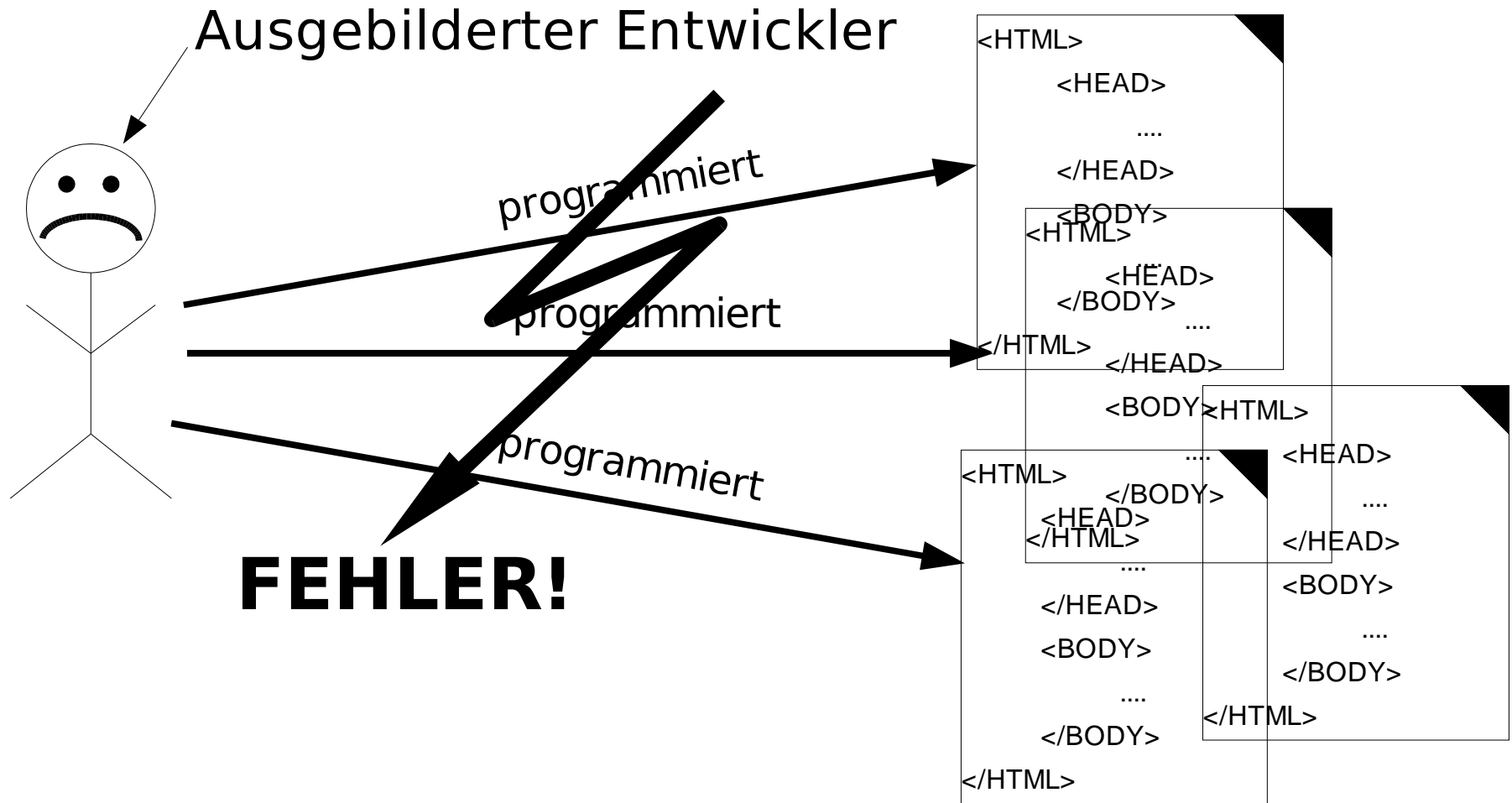
→ Für „kleine“ Projekte Generierung nicht nötig

Generierung von Web-Anwendungen - Warum?



→ „kleine“ Projekte können schnell komplex werden

Generierung von Web-Anwendungen - Warum?



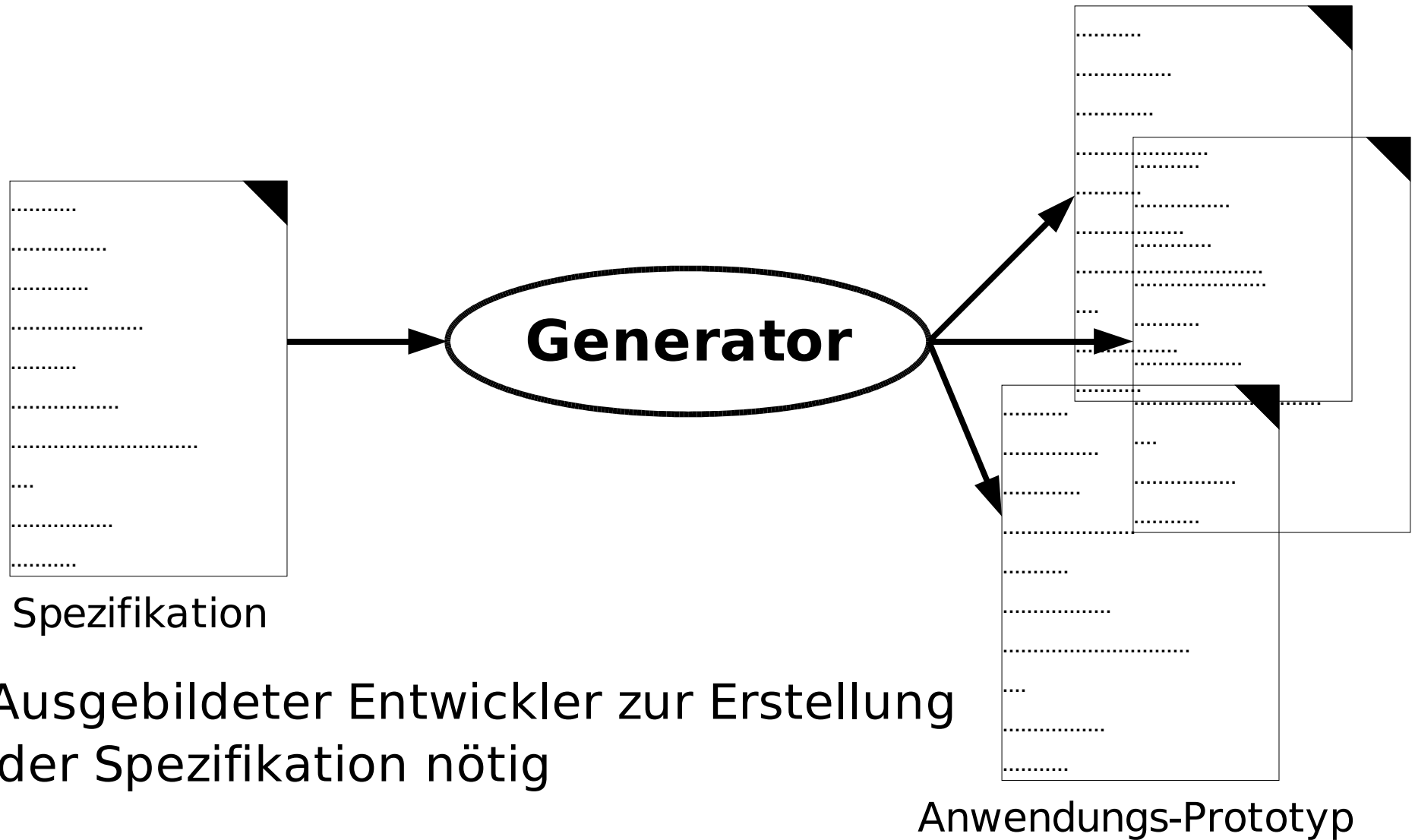
→ Generierung von Web-Anwendungen ist gut....

Generierung von Web-Anwendungen - Warum?

, denn:

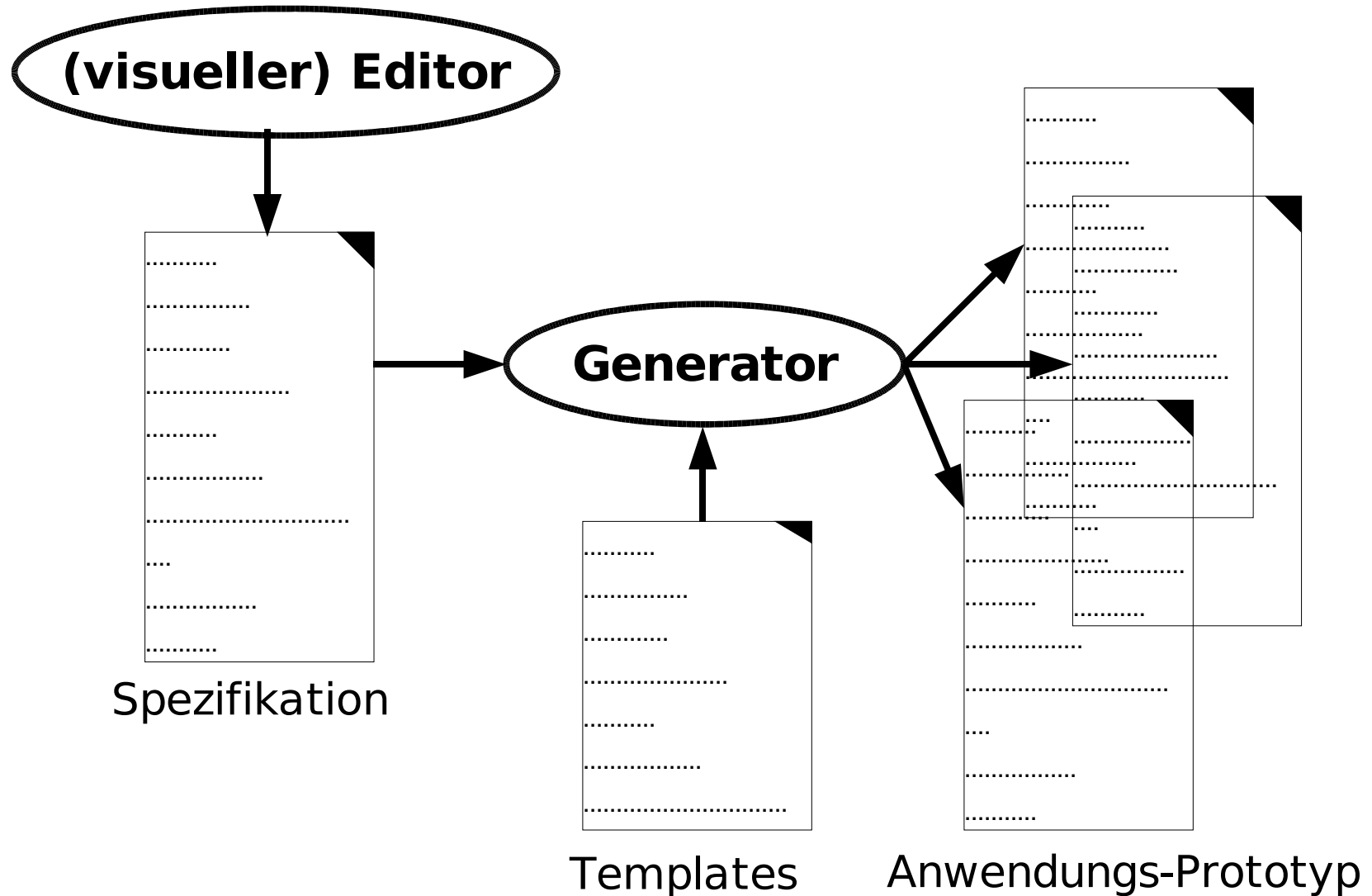
- Reduzierung der Komplexität
- Einfachere Wartbarkeit
- Generierung weniger fehlerbehaftet
- Entwicklung von Anwendungen durch „nicht-Entwickler“

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

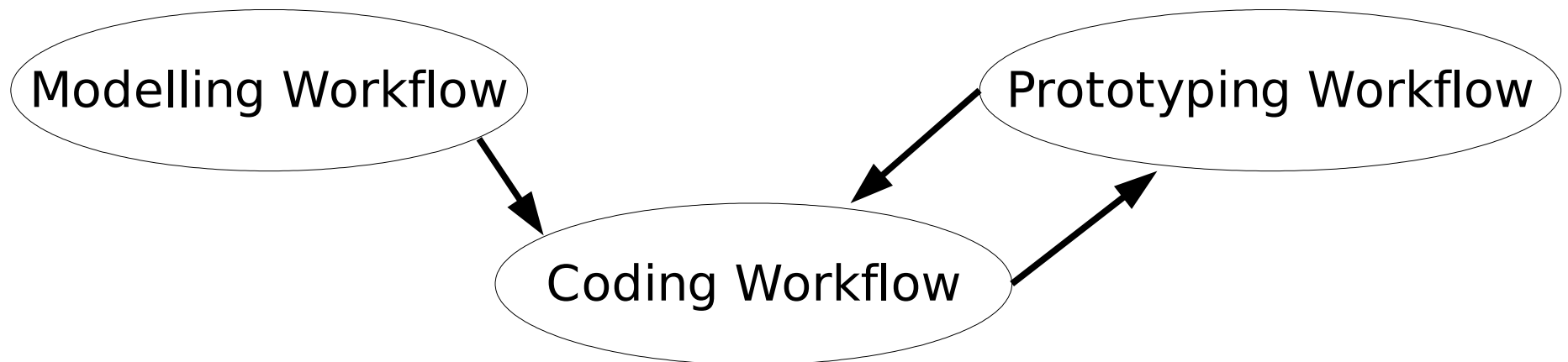
- Entwickelt an der Universität Tokushima
- Verwendet Bild-/Komponenten-basierten Ansatz

... 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

Modelling-Workflow

UML-Modell

Coding-Workflow

Script-Editor

ProGUM-Web

HTML-Editor

Prototyping-Workflow

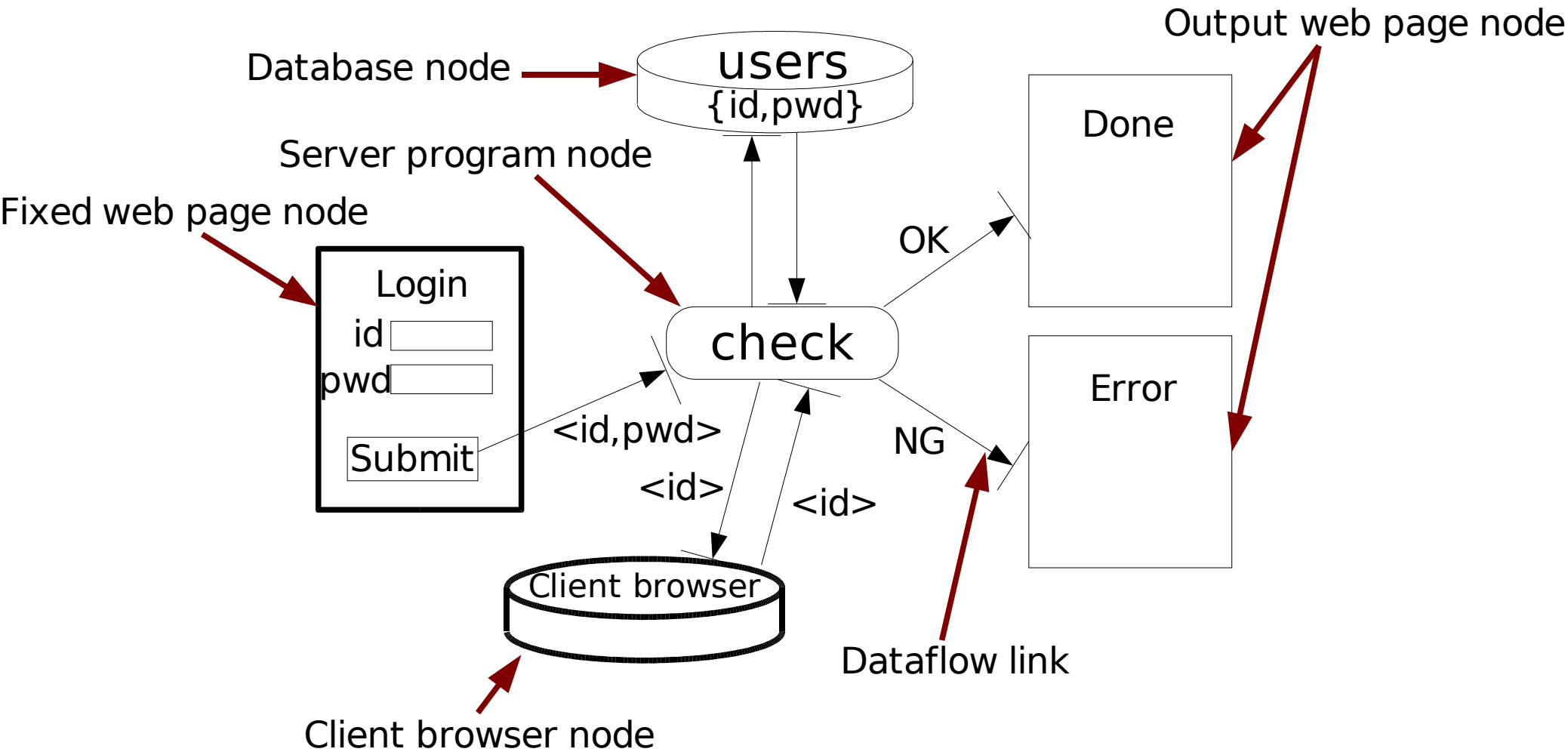
Web-Server

Web-Client

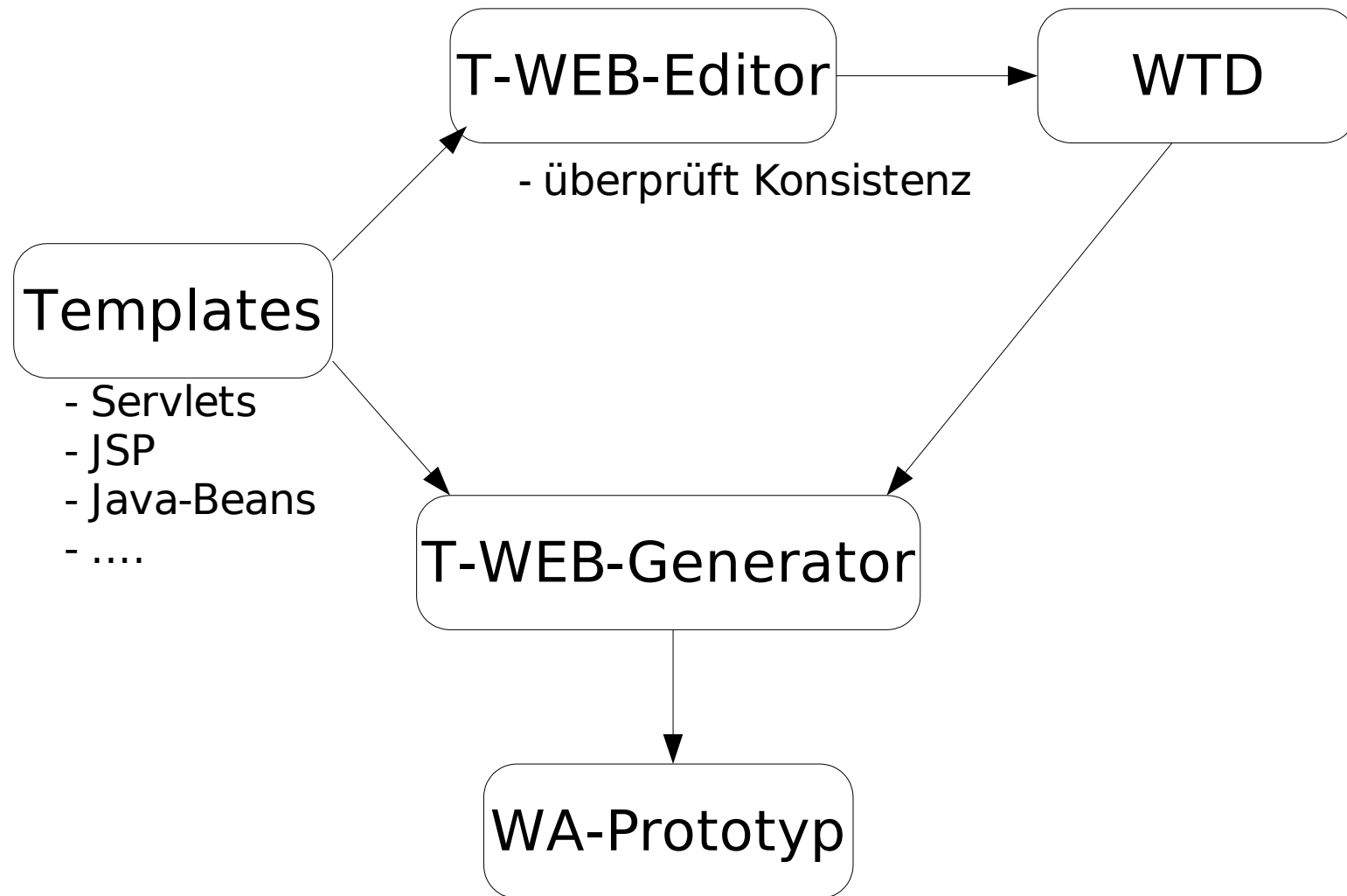
ProGUM - Überblick

- UML-Methodologie nutzbar
 - Modell-basierte Entwicklung
 - Workflow-basierter Entwicklungsprozess
- Teamwork durch zentrales Repository

T-WEB – Web-Transition-Diagrams



T-WEB - System



- 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

The screenshot shows the BioPro IDE interface for an online shop. It features several panels and components:

- entry: (Web page) [entry]:** A login form with fields for Name and Password, and buttons for Join, Enter, and shop: (Web ...).
- cake: (DB table): [Exists in DB]:** A table with columns: Field name, Type, and Sample. The data is as follows:

Field name	Type	Sample
name	String	Strawberry
image	String	Grape
price	Integer	100
1		Strawberry Shortcake
2		Cheesecake
3		MontBlanc
4		Chocolate Cake
- order: (Web page) [order]:** A table with columns: Cake, Image, Price, and Number. The content includes: `<%= cake.name %>`, `<%= html.img(cake.image) %>`, `<%= fmt.cur(cake.price) %>`, and `<%= Cart.num %>`. Buttons for Join, Change, and check: (Web ...) are visible.
- Cart: (Program table): [Code Generated]:** A table with columns: Field name, Type, and Sample. The data is as follows:

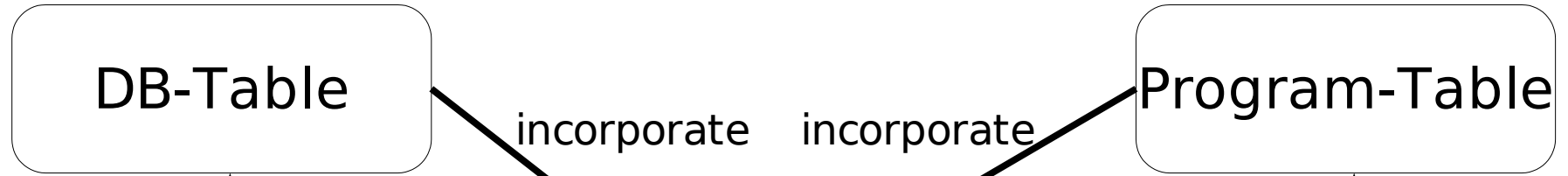
Field name	Type	Sample
name	String	Cheesecake
num	Integer	20

Annotations with red arrows point to various elements:

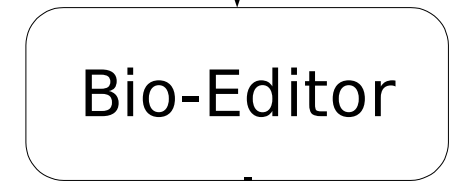
- Komponente:** Points to the 'entry' web page component.
- Web-Page:** Points to the 'order' web page component.
- „Platzhalter“:** Points to the `<%= cake.name %>` placeholder in the order page.
- Program-Table:** Points to the 'cake' and 'Cart' program tables.

BioPro – MVC-Architektur

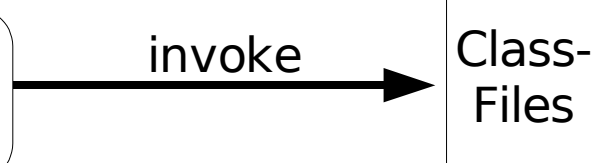
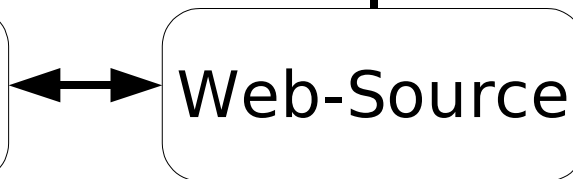
Model



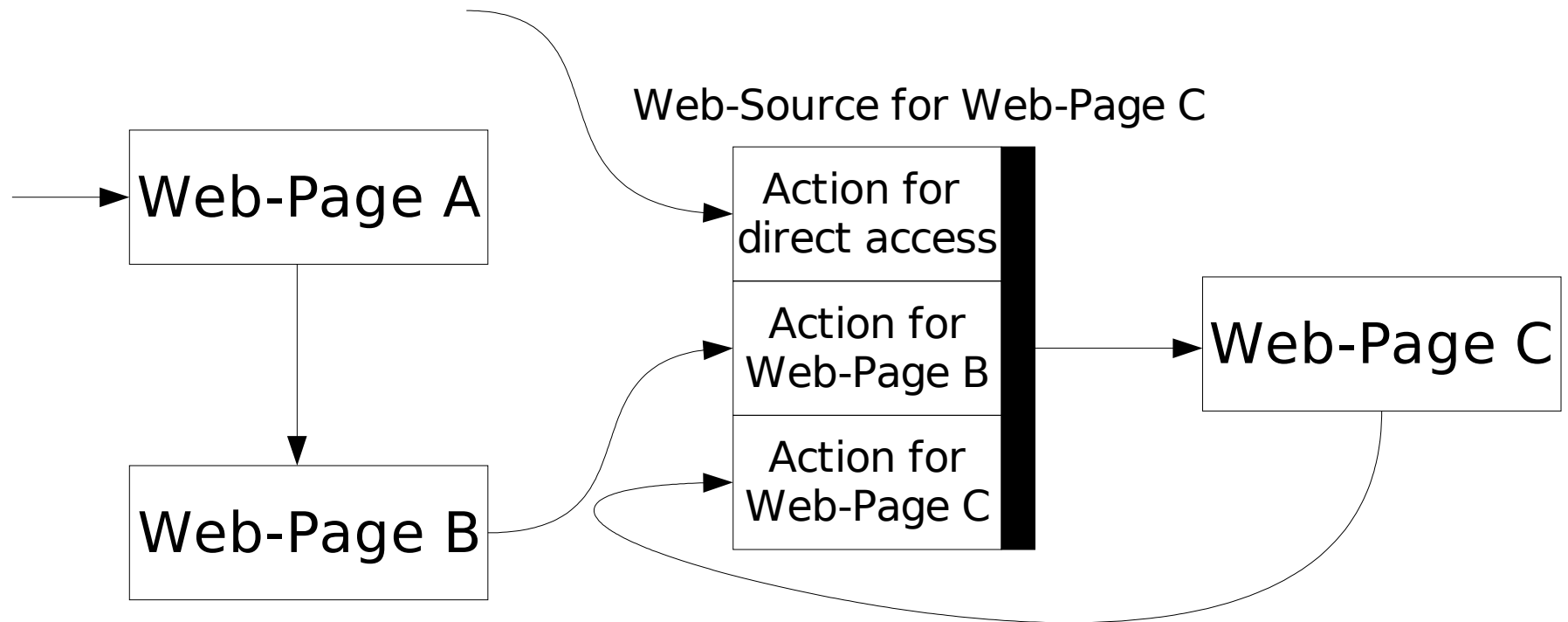
View



Controller



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 (Konsistenzchecks, Actions)
 - für den Aufbau des Editors (T-WEB, BioPro)
 - für den Generator (Templates, Repository)

Schluss

Löchern Sie den Vortragenden bitte JETZT mit Fragen ;-)