

SOPHIST



STABLE

Ein Ansatz zur systematischen
Strukturierung von Anforderungen

SOPHIST GmbH

Vordere Cramergasse 13

90478 Nürnberg

Tel: +49 (0)911 40 900 - 0

Fax: +49 (0)911 40 900 - 99

www.sophist.de

heureka@sophist.de

Wie lange dauert's noch?

Der Kontext:
Templates wider das Chaos

Der Ansatz:
Analyse, Modelle und Regeln

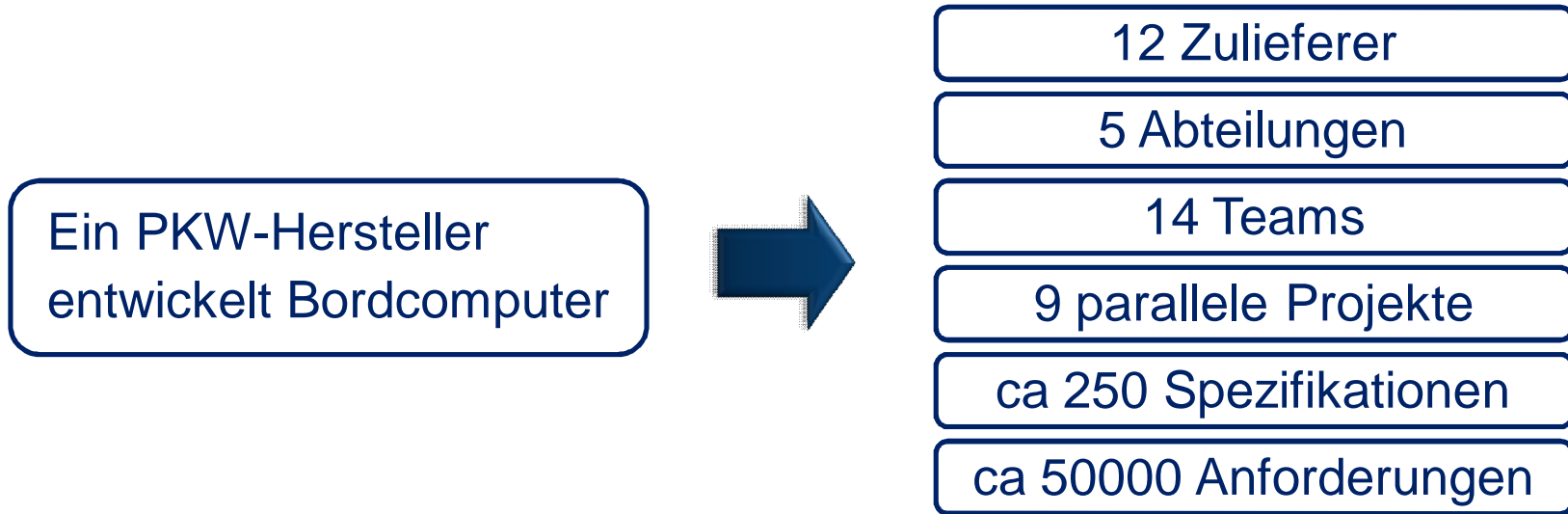
STABLE

Das Beispiel:
Strukturieren von Zuständen

Die Zukunft:
Umsetzung und Folgeprojekte

Templates wider das Chaos

Ein Fallbeispiel



Prozesstemplates

Dokumentvorlagen

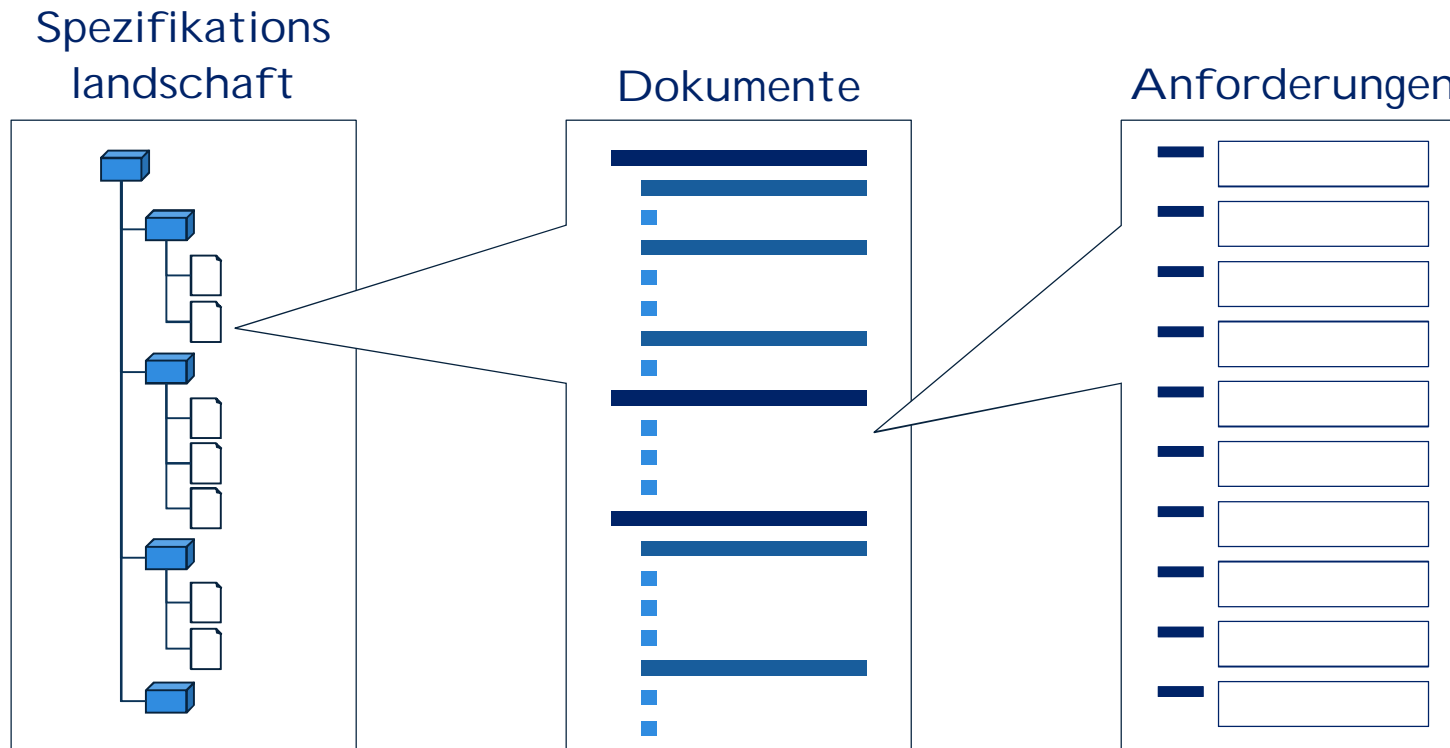
Standardgliederungen

Anforderungsschablonen

Spezifikationsleitfäden

Templates wider das Chaos

Wer strukturiert die Anforderungen?



strukturiert durch ...

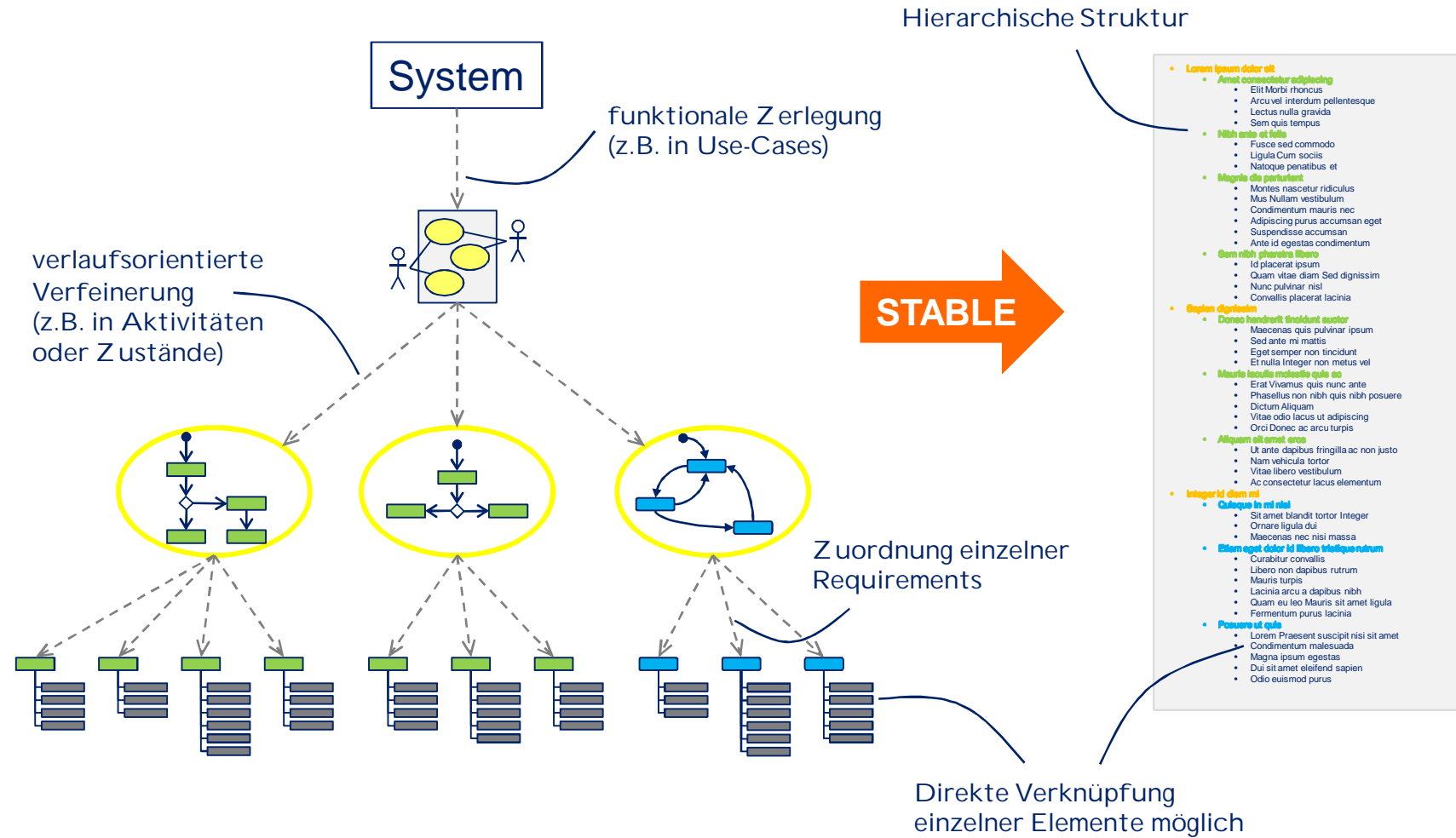
... Projektvorgaben
... Prozesse

... Standardgliederungen
... Dokumentvorlagen

... die Autoren?

Systematische Strukturierung

Use-Case Analyse als Grundlage für die Struktur



Das STABLE-Regelwerk

Schrittweise vom Modell zur Struktur

Aus den Köpfen ...

bewährte Prinzipien
aus Projekten und
Trainings



... in lesbare Form


Beispielmodell

Beispielstruktur



Automatisierung?

deterministisches
Regelwerk

Umsetzung von Akteuren ins Rollen- und Benutzerkonzept
SOPHIST 

Bedingung: Falls in der Spezifikationsstruktur noch kein Kapitel für Anforderungen an Benutzer und Rollen vorhanden ist

Beispiel

Akteure

Langschläfer

Zeitschaltuhr an Steckdose

Struktur **Variante**

„Halbautomatische“ Kaffeemaschine

1. Funktionale Anforderungen

1.1 Kaffeekocher vorbereiten

1.1.1 Kaffeefilter einsetzen

1.1.2 Kaffeepulver in Kaffeefilter geben

1.1.3 Kaffeemaschine mit Wasser befüllen

1.3 Kaffee entnehmen

1.4 Kaffee kochen

3. Rollen- und Benutzerkonzept

3.1 Langschläfer

3.2 Zeitschaltuhr an Steckdose

Regeln

- Für jeden Akteur, der in einem Use-Case Diagramm auftritt, muss ein Unterkapitel „<Name des Akteurs>“ angelegt werden.
- Dieses Kapitel enthält alle Anforderungen, die diesen Akteur betreffen.

SOPHIST GmbH

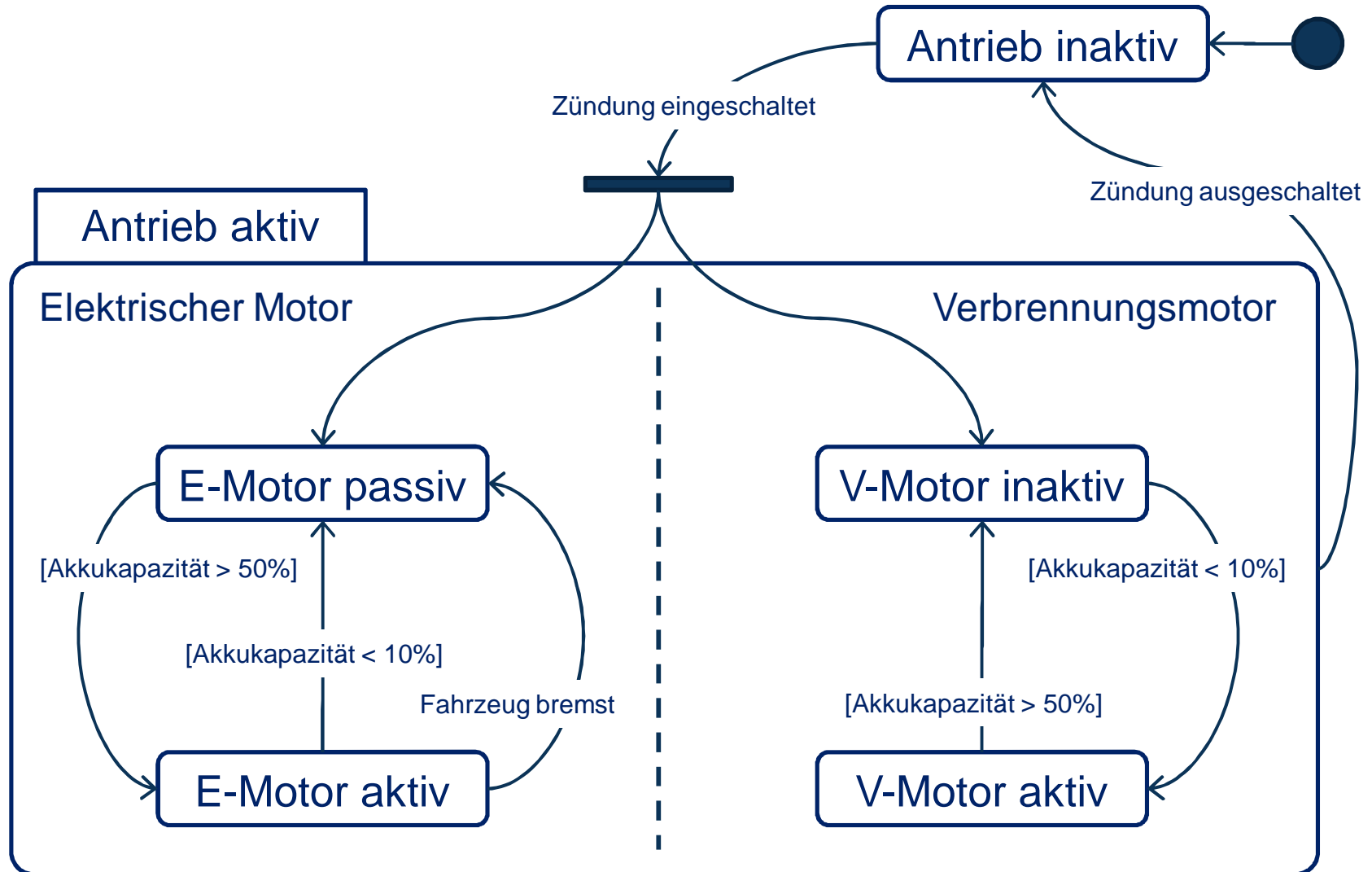
STABLE - Regelwerk

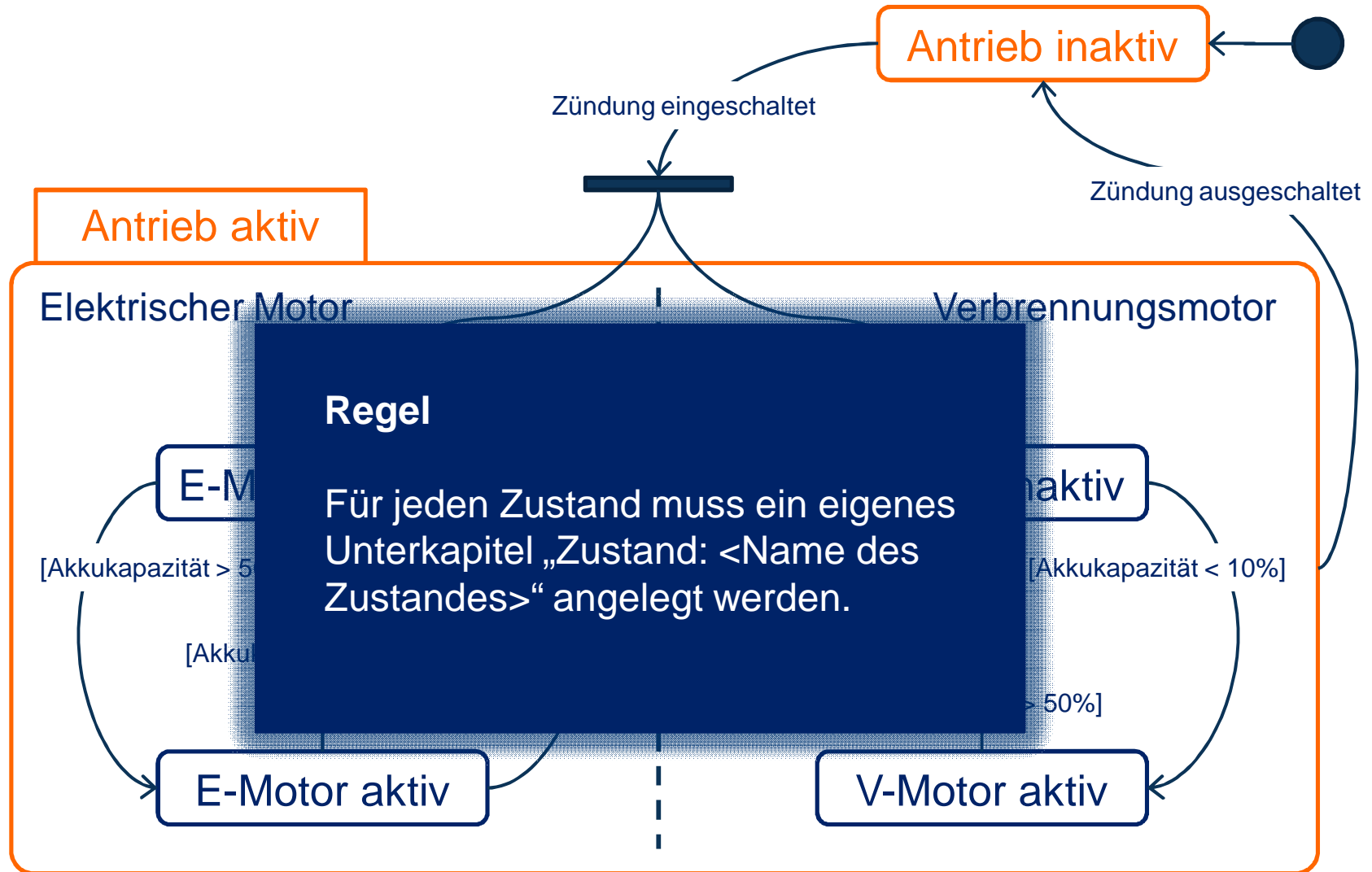
Regeln

Hybrid-Antrieb

Der (vereinfachte) Zustandsautomat

Das Beispiel

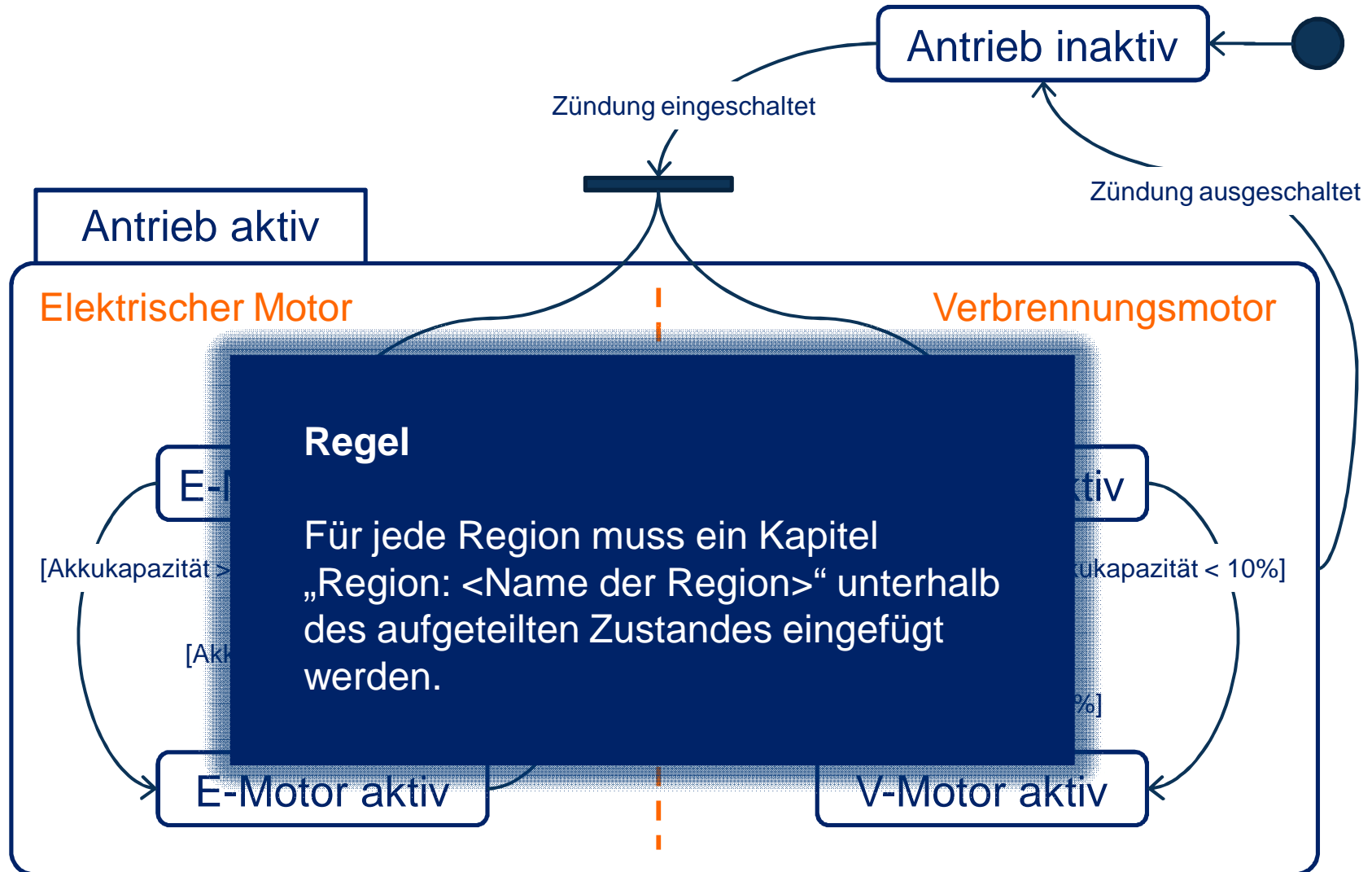




Zustände

Die Struktur der Anforderungen

1. Zustand: Antrieb inaktiv
2. Zustand: Antrieb aktiv



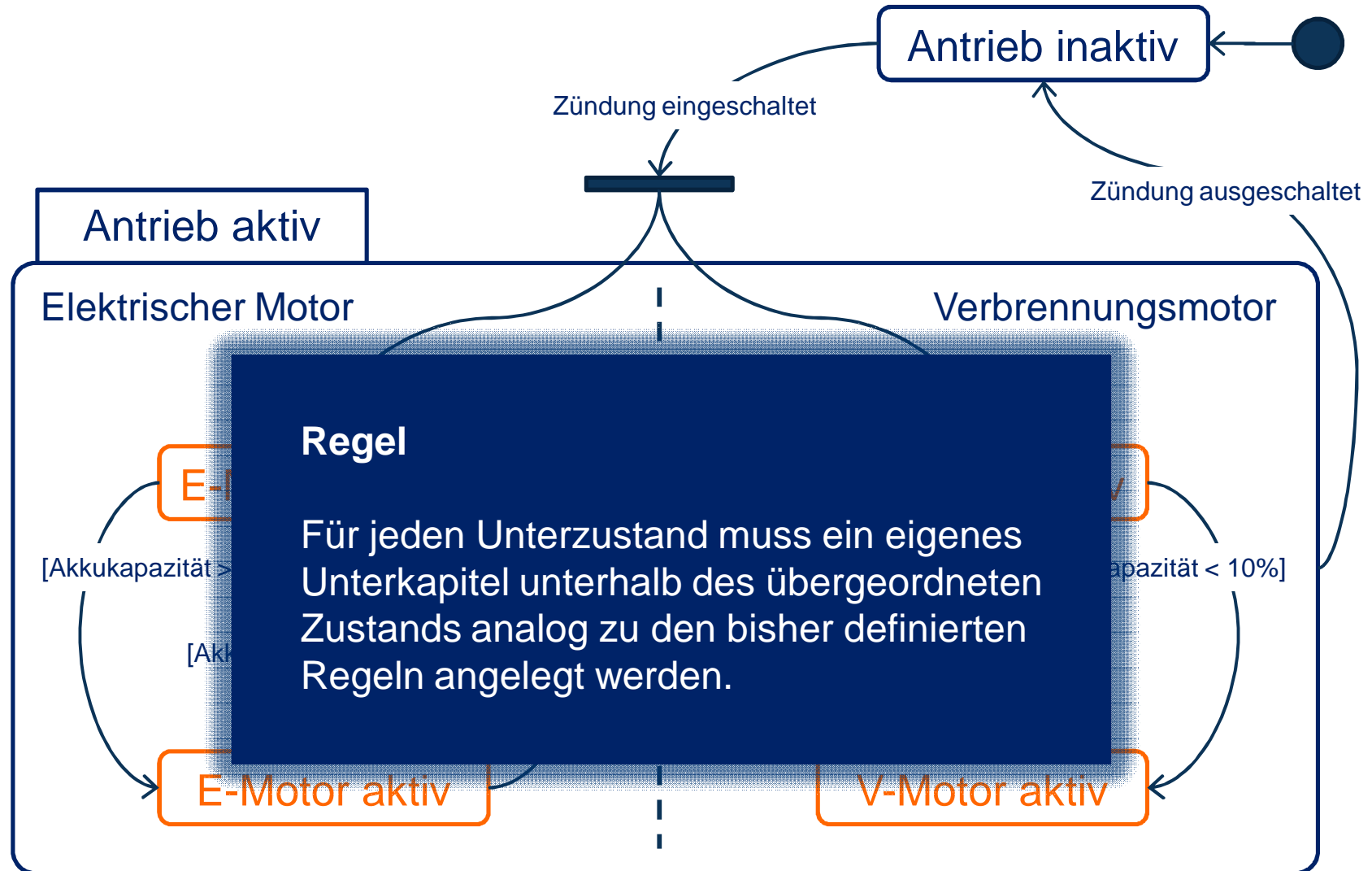
Regionen

Die Struktur der Anforderungen

1. **Zustand: Antrieb inaktiv**
2. **Zustand: Antrieb aktiv**
 1. **Region: Elektrischer Motor**
 2. **Region: Verbrennungsmotor**

Untertzustände

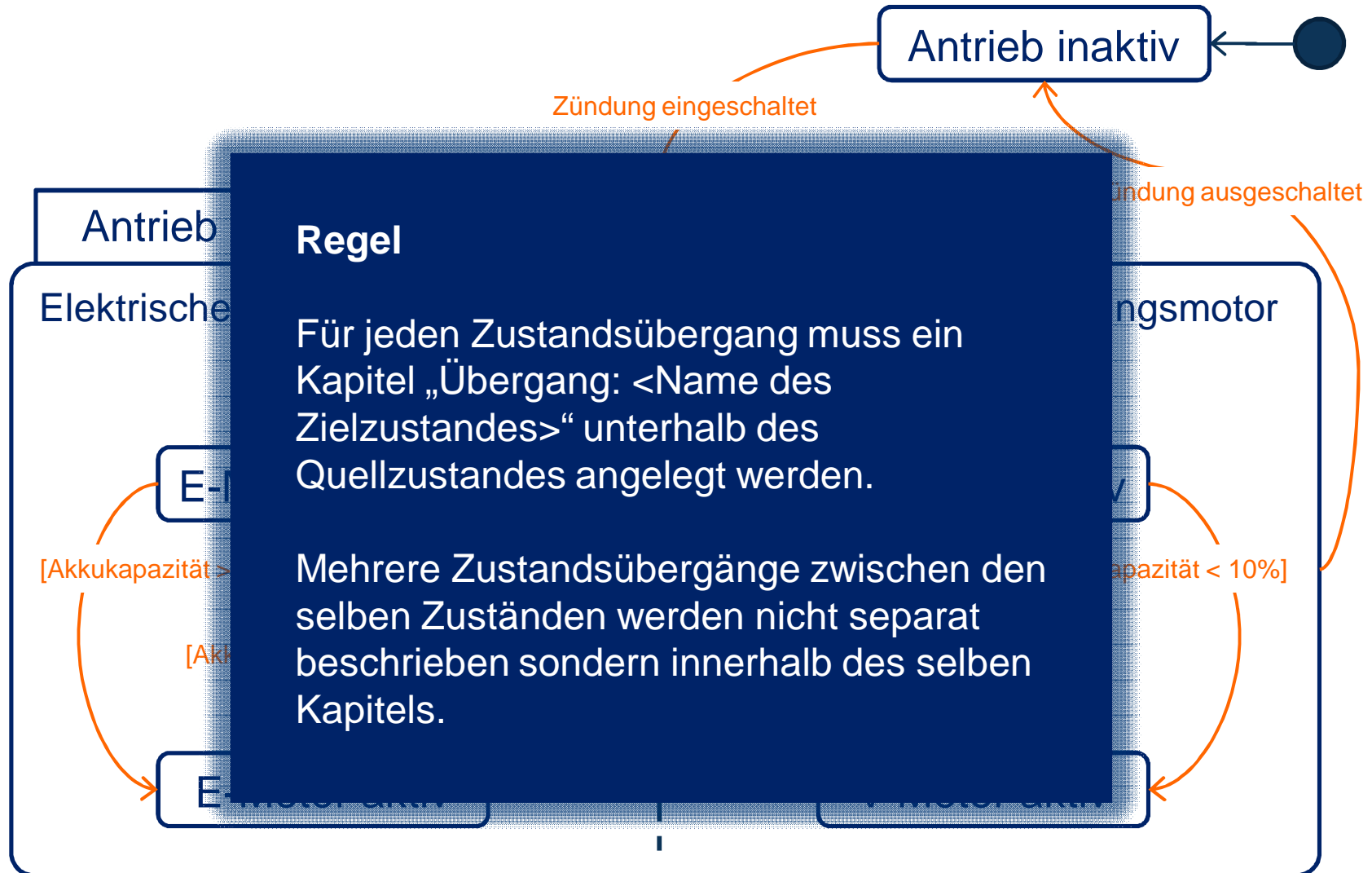
Das Beispiel



Untierzustände

Die Struktur der Anforderungen

1. **Zustand: Antrieb inaktiv**
2. **Zustand: Antrieb aktiv**
 1. Region: Elektrischer Motor
 1. **Zustand: E-Motor passiv**
 2. **Zustand: E-Motor aktiv**
 2. Region: Verbrennungsmotor
 1. **Zustand: V-Motor inaktiv**
 2. **Zustand: V-Motor aktiv**



Zustandsübergänge

Die Struktur der Anforderungen

1. **Zustand: Antrieb inaktiv**
 1. Übergang: E-Motor passiv
 2. Übergang: E-Motor aktiv
2. **Zustand: Antrieb aktiv**
 1. Übergang: Antrieb inaktiv
 2. Region: Elektrischer Motor
 1. **Zustand: E-Motor passiv**
 1. Übergang: E-Motor aktiv
 2. **Zustand: E-Motor aktiv**
 1. Übergang: E-Motor passiv
 3. Region: Verbrennungsmotor
 1. **Zustand: V-Motor inaktiv**
 1. Übergang: V-Motor aktiv
 2. **Zustand: V-Motor aktiv**
 1. Übergang: V-Motor inaktiv

Hybrid-Antrieb

Die Struktur der Anforderungen

- 1. Zustand: Antrieb inaktiv**
 1. Übergang: E-Motor passiv
 2. Übergang: E-Motor aktiv
- 2. Zustand: Antrieb aktiv**
 1. Übergang: Antrieb inaktiv
 2. Region: Elektrischer Motor
 - 1. Zustand: E-Motor passiv**
 1. Übergang: E-Motor aktiv
 - 2. Zustand: E-Motor aktiv**
 1. Übergang: E-Motor passiv
 3. Region: Verbrennungsmotor
 - 1. Zustand: V-Motor inaktiv**
 1. Übergang: V-Motor aktiv
 - 2. Zustand: V-Motor aktiv**
 1. Übergang: V-Motor inaktiv

Was kommt als nächstes?

Mittelfristige Pläne mit STABLE

Feedback einholen:

Idee publizieren (z.B. hier)
und Anregungen sammeln

Automatisierung umsetzen:

In der Projektrealität den
Determinismus belegen



STABLE 2

Regeln verfeinern:

Regeln für weitere
Notationselemente
entwickeln

Regeln erweitern:

Regeln für statische Modelle
(z.B. Klassendiagramm)
entwickeln

