BERICHT ZU KI-ARCHITEKTUR & UNTERNEHMENSSTRATEGIE

# Der Fliegeneffekt: Ereignisgesteuerte Agentenarchitekturen und der Kostenirrtum

LLM-Kosten im Unternehmen senken, ausgehend vom Trägheitsprinzip im fahrenden Auto

**Mehmet Keskinkılıç** · November 2026 **Kategorie:** Systemdesign & Finanzoptimierung

**Ansatz:** Deterministische Zustandsautomaten + gezielte Inferenz

***Zusammenfassung:*** *Viele Unternehmen verschieben die KI-Transformation und die Einführung autonomer Agenten wegen „unvorhersehbarer API-Kosten“. Ein wichtiger Grund dafür ist nicht fehlende Technologie, sondern eine Architektur, die bei jedem Arbeitsschritt große Modelle auslöst. Inspiriert von der Physik einer Fliege im geschlossenen Auto beschreibt diese Arbeit den Weg zu einem „Beschleunigungs-Agentenmodell“: Routinevorgänge bleiben bei null Token, das LLM läuft nur bei Anomalien. Die Zahlen in diesem Bericht sind Annahmen eines Beispielszenarios, keine Messwerte.*

## 1. Die physikalische Metapher: Warum wird die Fliege in der geschlossenen Kabine nicht herumgeschleudert?

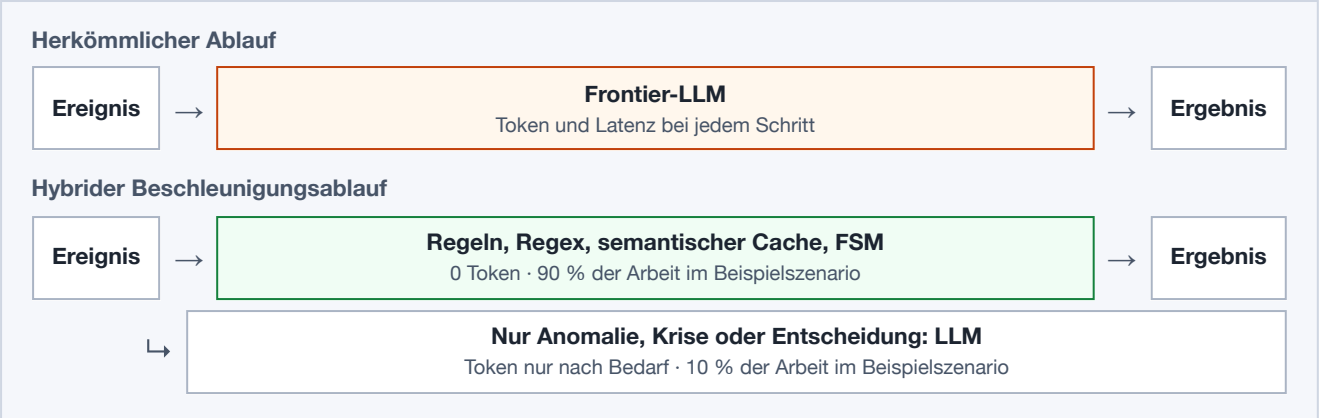
Eine Fliege in einem geschlossenen Auto, das mit 100 km/h fährt, braucht keine zusätzliche Anstrengung, um in der Luft zu stehen oder von der Windschutzscheibe zur Rückbank zu fliegen. Dafür gibt es zwei physikalische Gründe: *die mit gleicher Geschwindigkeit mitbewegte Luftmasse* und das *Trägheitsprinzip*. Die Fliege bewegt sich relativ zur ruhenden Luft in der Kabine, nicht relativ zur Straße. Viel Energie für ihr Gleichgewicht braucht sie erst, wenn das Auto scharf bremst, beschleunigt oder ein offenes Fenster Turbulenzen hereinlässt.

Herkömmliche Agentendesigns verhalten sich, als müsste die Fliege außerhalb des Autos gegen 100 km/h Fahrtwind fliegen: Jede routinemäßige Dateneingabe, jede Statusprüfung und jeder einfache Freigabeschritt löst ein großes Sprachmodell (LLM) von Anfang bis Ende aus, und der Tokenverbrauch gerät außer Kontrolle.

## 2. Der Fehler im Unternehmen: Der Reflex, an jeden Schritt ein LLM zu hängen

Hinter der Aussage „Die KI-Transformation sprengt unser Budget“ steht oft ein monolithisches Agentendesign, das jedes operative Ereignis direkt an ein teures Frontier-Modell weitergibt. Dabei lässt sich ein erheblicher Teil der Unternehmensprozesse (Kundenanfragen, Rechnungsprüfungen, ERP-Aktualisierungen, Ticket-Weiterleitung) mit vorhersehbaren, deterministischen Regeln abwickeln. Dieser Anteil ist je nach Prozess verschieden und muss gemessen werden; das Beispielszenario dieses Berichts nimmt an, dass 90 % der Arbeit abgeschlossen werden, ohne ein Modell zu erreichen.

© 2026 Mehmet Keskinkılıç · mehmetkeskinkilic.com




## 3. Die drei tragenden Säulen der Architektur

### A. Deterministische Torwächter-Schicht (Gating Layer)

Jede Eingabe durchläuft zuerst Prüfungen ohne oder mit minimalen Kosten. Passt sie zu einem Standardmuster, einer festgelegten API-Route oder einer regelbasierten Validierung, wird der Vorgang direkt im Code abgeschlossen. Es gibt keinen LLM-API-Aufruf und keinen Tokenverbrauch.

### B. Arbeitsablauf auf Basis eines Zustandsautomaten (Finite State Machine, FSM)

Die Prozesse des Agenten werden mit zustandsorientierten Frameworks wie LangGraph modelliert. Beim Übergang zwischen Zuständen laufen die Entscheidungsregeln im Code. Nur bei einer unerwarteten Ausnahme, einer unstrukturierten Datenanomalie oder einer strategischen Verzweigung mit mehreren Optionen löst das System einen „Beschleunigungsalarm“ aus und verzweigt zum LLM-Knoten.

### C. Semantisches Caching und differenzbasierte Inferenz (Diff)

Wiederkehrende Anfragen werden mit leichten lokalen Embedding-Modellen erkannt und direkt aus dem Cache beantwortet. Auch wenn ein LLM-Aufruf nötig ist, wird nicht jedes Mal der gesamte Dokumentenbestand gesendet; Prompt Caching und die Übertragung nur der geänderten Zustandsparameter (Delta) senken die Ausgaben.

## 4. Kosten- und Leistungsvergleich (Beispielszenario)

Die folgende Tabelle ist keine Messung einer echten Installation. Sie ist ein Beispielszenario für ein mittelgroßes Support- oder Bearbeitungszentrum mit 10.000 operativen Ereignissen pro Tag.

| Kennzahl / Phase | Monolithischer LLM-Agent | Hybrides Beschleunigungsmodell | Einsparung & Wirkung |
|---|---|---|---|
| **Routineablauf (75 %)** | ~1.500 Token pro Aufruf | 0 Token (Ausführung auf Regel- und API-Ebene) | 100 % LLM-Token gespart |
| **Ähnliche / wiederholte Eingabe (15 %)** | Modell denkt von vorn | 0 Token (Treffer im semantischen Cache) | 100 % LLM-Token gespart |
| **Anomalie & Entscheidung (10 %)** | Läuft mit teurem Kontext | Kleines oder optimiertes Modell + minimaler Diff-Kontext | 70 % Token-Optimierung |
| **Beispielhafte Antwortzeit** | 1.800 ms – 3.500 ms | < 40 ms (Routine) / 900 ms (Anomalie) | Über 97 % weniger bei Routine, etwa 50–75 % weniger bei Anomalien |
| **Geschätzte Monatskosten** | 4.500 $ – 7.000 $ / Monat | 350 $ – 600 $ / Monat | **Etwa 87 % – 95 % geschätzte Einsparung** |

**Rechenannahme:** Rund 300.000 Ereignisse im Monat × ~1.500 Token = ~450 Millionen Token, im Schnitt etwa 10–16 $ pro Million Token. Damit kostet der 10-%-Anomalieanteil im Hybridmodell etwa 135–215 $ an Token; die Spanne in der Tabelle liegt als vorsichtige Schätzung darüber. Regelerstellung, Infrastruktur, Wartung und Cache-Betrieb sind in beiden Spalten nicht enthalten.

## 5. Zusammenfassung für den Vorstand und Aktionsplan

> **Strategisches Ergebnis:** KI-Transformation heißt nicht, „alle Prozesse einem LLM zu übergeben“. Eine erfolgreiche Transformation trennt die deterministische Verlässlichkeit der Softwaretechnik und die Problemlösungsfähigkeit von LLMs an den richtigen Schwellen. Wenn wie im Beispielszenario 90 % der Arbeit ohne Tokenverbrauch laufen und KI nur in den übrigen 10 % unsicherer Momente einspringt, kann aus der Kostenhürde ein Vorteil werden.

### Aktionsplan

1. **Messen:** Erfassen Sie aktuelle LLM-Aufrufe nach Ereignistyp mit Volumen, Token, Latenz und Fehlerquote. Ersetzen Sie die Anteile dieses Berichts durch eigene Daten.
2. **Trennen:** Verlagern Sie volumenstarke Routinearbeit mit überprüfbarem Ergebnis in Regeln und Zustandsautomaten.
3. **Cache kontrolliert öffnen:** Betreiben Sie den semantischen Cache mit Ähnlichkeitsschwelle, prüfen Sie Stichproben und verhindern Sie, dass personenbezogene oder sensible Daten zwischen Antworten wechseln.
4. **Eskalationsregel festlegen:** Senden Sie Anomalien zuerst an ein kleines Modell. Bei geringer Sicherheit eskalieren Sie an ein großes Modell oder einen Menschen.
5. **Laufend überwachen:** Verfolgen Sie den ohne Modell erledigten Arbeitsanteil, falsche Regel- und Cache-Antworten und die monatlichen Gesamtkosten einschließlich Entwicklung.

**Mehmet Keskinkılıç**
Luftfahrt-Operations-Spezialist & Indie-iOS-Entwickler
© 2026 Mehmet Keskinkılıç. Weitergabe mit Quellenangabe erlaubt. Aktuelle Fassung: mehmetkeskinkilic.com/de/writing/