virtauto Governance Layer – Zusammenfassung

Diese Zusammenfassung beschreibt den Governance-Layer von virtauto sowie seine Verbindung zu den drei strategischen Prinzipien aus dem „AI Investment Trap“-Framework: Continuous Learning, Optionality und Autonomy Portfolio Management.

---

1. Überblick: Existierende Governance-Schichten

---

virtauto besitzt bereits heute einen mehrschichtigen Governance-Layer:

A) Strategy / Intent

- Vision, Blog, Architekturprinzipien
- langfristige Ausrichtung agentischer Systeme

B) Governance / Guardrails

- guardian.yaml (Security Monitoring, Risk Flags, Auto-Stop-Rules)
- policies/ (Compliance-/Safety-Regeln, Investitionslogik möglich)
- emergency_lock.json (Not-Aus)
- george_rules.yaml (Orchestrierungs- und Routingregeln)

- HealthState & Health-Metrics in george_orchestrator_v2.py
- persistente Entscheidungslogs in ops/decisions/

C) Execution Layer

- Deploy-, Content-, Monitoring-, Compliance- und Guardian-Agent
- GEORGE Orchestrator
- Events, Routing & autonome Aktionserzeugung

---

2. Verbindung zu den drei Governance-Prinzipien

2.1 Continuous Learning (Kontinuierliches Lernen)

AI wird als dynamisches Asset betrachtet. Governance soll sicherstellen:

- Modelle werden regelmäßig aktualisiert.
- Datenalter („Data Staleness“) wird überwacht.
- RAG- und Wissenssysteme werden periodisch erneuert.
- Deployments können blockiert werden, wenn die Datenbasis veraltet ist.

Umsetzung:

- learning_policy.yaml mit Parametern wie max_data_staleness_days
- Guardian prüft vor Deploys die Einhaltung
- HealthState kann einen freshness_score integrieren

2.2 Optionality (Optionen ermöglichen)

Agentic AI erzeugt oft Wert außerhalb geplanter Use Cases.
Governance ermöglicht bewusst:

- Safe-to-Explore-Zonen
- Budgetierter Raum für Experimente
- Logging von emergent_value_events

Umsetzung:

- Erweiterungen in autonomy.json:
  * investment_mode: production / optionality / experiment
  * budget_slots pro Agent
- Guardian regelt, wie viele explorative Aktionen erlaubt sind
- Decisions-Persistenz dokumentiert emergenten Wert

2.3 Portfolio of Autonomy Levels (Autonomie-Portfolio)

Nicht alle Agenten laufen auf voller Autonomie.
Governance verwaltet ein Portfolio:

- guided → semi_autonomous → autonomous
- Risk-Tiers bestimmen maximale Autonomie
- Guardian reguliert Übernutzung oder riskante Eskalationen

Umsetzung:

- autonomy.json Felder:

* autonomy (0–1)

* mode (guided / semi_autonomous / autonomous)

* requires_human_confirm

* max_daily_actions

- Guardian prüft Autonomiegrenzen, eskaliert bei Verstößen

- Autonomy Changes laufen über Requests als Governance-Events

---

3. Konkrete Governance-Artefakte (Vorschlag)

A) governance_core.yaml

Definiert die drei Prinzipien + messbare Regeln

B) learning_policy.yaml

Stellt sicher, dass Wissenspflege budgetiert und überwacht wird

C) autonomieabhängige Guardian-Regeln

Regeln für:

- Autonomieänderungs-Requests

- Blocking Conditions (z. B. HealthScore zu niedrig)

- Übernutzungsgrenzen

D) Erweiterte autonomy.json

Dokumentiert Fähigkeiten, Risiken und Verantwortungen aller Agenten

---

4. Governance Summary

Der virtauto-Governance-Layer existiert bereits in Form eines funktionierenden Sicherheits-, Orchestrierungs- und Entscheidungsrahmens. Um den „AI Investment Trap“ aktiv zu adressieren, wird dieser Layer nun erweitert, sodass er:

- kontinuierliche Lerninvestitionen sicherstellt,
- Emergenz und ungeplanten Wert systematisch erfasst,
- ein steuerbares Autonomieportfolio ermöglicht.

Damit wird virtauto zu einem realen praktischen Beispiel agentischer Governance auf Systemebene.