ZERYON · PREGA DESIGN

WHITEPAPER

Beweisbar statt plausibel

Governance-by-Design für kontrollierte KI-Systeme — Architektur und Kontroll-Plane von ZERYON Control Studio

Produkt	ZERYON Control Studio
Version	2.4.0
Dokumentstand	30. Juli 2026
Herausgeber	Predrag Gasic · Prega Design

Inhaltsverzeichnis

Management Summary 3

Der Problemraum: Wenn KI plausibel, aber nicht beweisbar ist 4

- Die Vertrauenslücke 4
- Woran KI-Vorhaben in der Praxis scheitern 4
- Warum der Zeitpunkt zählt 4

Die Doktrin: KI schlägt vor, das System entscheidet 5

Architektur des Kontroll-Plane 6

- Schichtenmodell 6
- Backend-autoritative Autorisierung 6
- Lokal zuerst, minimale Angriffsfläche 6

Die fünf Kontrollmechanismen im Detail 7

- 1. Deterministische Policy-Engine 7
- 2. State Machine mit erzwungenen Übergängen 7
- 3. Freigabe-Workflow mit erzwungener Funktionstrennung 7
- 4. Manipulationssicherer Audit-Ledger 8
- 5. Rollenbasierte Zugriffskontrolle (RBAC) 8

Governance-by-Design & EU AI Act 9

Sicherheit, Betrieb und Reifegrad 10

- Sicherheit als Grundhaltung 10
- Ehrliche Abgrenzung des Reifegrads 10

Handlungsempfehlung 11

Management Summary

Künstliche Intelligenz ist in Unternehmen angekommen — aber das Vertrauen in ihre Entscheidungen hinkt hinterher. Der Grund ist selten die Modellqualität. Der Grund ist, dass die meisten KI-Systeme **plausibel** wirken, ohne **beweisbar** zu sein: Sie liefern ein Ergebnis, aber keinen belastbaren Nachweis, *warum* es zustande kam, *wer* es hätte verhindern können und *ob* eine verbindliche Regel es überhaupt zugelassen hätte.

Für unkritische Aufgaben ist das akzeptabel. Für Vorgänge mit Geld-, Rechts- oder Compliance-Folgen ist es ein Haftungsrisiko. Genau hier verläuft die Grenze zwischen einem KI-Experiment und einem einsetzbaren KI-System.

ZERYON Control Studio schließt diese Lücke mit einer einfachen, aber konsequenten Doktrin: **KI schlägt vor, das System entscheidet.** Zwischen den Vorschlag eines KI-Agenten und die Ausführung einer Aktion schiebt ZERYON eine deterministische Kontrollschicht: Richtlinienprüfung, menschliche Freigabe bei Hochrisiko-Aktionen, Funktionstrennung und ein manipulationssicheres Audit-Protokoll. Jede sicherheitsrelevante Aktion ist damit einer Identität, einer Rolle, einer Berechtigung, einer Richtlinie und einem lückenlosen Nachweis zuordenbar.

Das Leitprinzip ist **beweisbar statt plausibel**: nicht die Wahrscheinlichkeit einer guten Entscheidung, sondern der überprüfbare Nachweis, dass jede Entscheidung durch geprüfte, deterministische Regeln gedeckt und vollständig protokolliert ist.

Dieses Whitepaper richtet sich an zwei Leserkreise. Der erste Teil (Kapitel 1–3, 7–9) adressiert Entscheider, Compliance- und Risikoverantwortliche: Er ordnet das Governance-Problem ein und zeigt, wie ZERYON regulatorische Anforderungen — insbesondere den EU AI Act — praktisch bedient. Der zweite Teil (Kapitel 4–6) adressiert Architekten und Security-Engineering: Er beschreibt das Kontroll-Plane, die deterministische Policy-Engine, den kryptografisch verketteten Audit-Ledger und das Sicherheitsmodell im Detail.

Der Problemraum: Wenn KI plausibel, aber nicht beweisbar ist

Die Vertrauenslücke

Ein modernes Sprachmodell kann eine Rechnung analysieren, eine Zahlung vorschlagen und die Begründung dazu formulieren — flüssig, überzeugend, in Sekunden. Was es nicht kann: garantieren, dass diese Zahlung eine verbindliche Richtlinie einhält, dass ein zweiter Mensch sie bei Bedarf freigegeben hat und dass der gesamte Vorgang später revisionssicher nachvollziehbar bleibt.

Diese Lücke ist strukturell, nicht temporär. Sie verschwindet nicht mit dem nächsten, größeren Modell. Ein KI-System, das seine eigene Ausführung kontrolliert, bleibt eine Blackbox mit Handlungsvollmacht — und genau das ist in regulierten oder haftungsrelevanten Prozessen nicht tragbar.

Woran KI-Vorhaben in der Praxis scheitern

Nicht am Modell, sondern an der fehlenden Kontrollarchitektur drumherum:

- **Keine erzwingbare Regelbindung.** Prompts und Systemanweisungen sind Bitten, keine Garantien. Ein Modell kann eine Regel ignorieren; ein deterministisches Gate kann es nicht.
- **Keine belastbare Freigabe.** „Der Mensch schaut noch mal drüber" ist kein Kontrollmechanismus, solange nicht technisch erzwungen ist, *wer* freigeben darf und *dass* er nicht sich selbst freigibt.
- **Kein manipulationssicherer Nachweis.** Ein Log, das sich nachträglich ändern lässt, ist im Streitfall wertlos. Auditierbarkeit entsteht nicht durch mehr Logging, sondern durch *unveränderliche* Protokolle.
- **Unklare Verantwortlichkeit.** Wenn nicht jede Aktion einer Identität und Rolle zugeordnet ist, gibt es im Ernstfall niemanden, der belegen kann, was warum geschah.

Warum der Zeitpunkt zählt

Mit dem EU AI Act, NIS2 und den etablierten Anforderungen aus DSGVO und Revisionssicherheit verschiebt sich die Frage von „Kann unsere KI das?" zu „Können wir belegen, wie unsere KI zu Entscheidungen kommt — und wer sie kontrolliert?". Wer diese Frage nicht mit Nachweisen beantworten kann, trägt das Risiko. Governance wird damit vom Nice-to-have zur Voraussetzung für den produktiven Einsatz.

> **Kernaussage:** Das Problem ist nicht, dass KI Fehler macht. Das Problem ist, dass ungoverntes KI-Handeln nicht *beweisbar* kontrolliert und nicht *belastbar* nachvollziehbar ist. Genau das lässt sich mit Architektur lösen — nicht mit einem besseren Prompt.

Die Doktrin: KI schlägt vor, das System entscheidet

ZERYON Control Studio ist bewusst **kein Chat**. Es ist eine Kontroll- und Nachweisumgebung für KI-gestützte Vorgänge. Der KI-Agent verliert dabei nichts von seiner analytischen Stärke — er verliert nur die Vollmacht, seine eigenen Vorschläge unkontrolliert auszuführen.

Der zentrale Ablauf ist immer derselbe und immer sichtbar:

```
Agent → Workflow → Policy-Prüfung → Approval → Ausführung → Audit-Ledger
```

Ein Agent analysiert einen Vorgang und schlägt eine Aktion vor. Ein Workflow bringt diesen Vorschlag in einen definierten Ablauf. Eine deterministische Policy-Prüfung entscheidet, ob die Aktion erlaubt ist, blockiert wird oder eine menschliche Freigabe erfordert. Erst nach bestandener Kontrolle — und, wo nötig, nach unabhängiger Freigabe — wird die Aktion (simuliert) ausgeführt. Jeder einzelne Schritt wird in einem manipulationssicheren Audit-Ledger festgeschrieben.

Vier Prinzipien tragen diese Doktrin:

Determinismus. Die Kontrollentscheidungen fallen in regelbasiertem, reproduzierbarem Code — nicht im Modell. Dieselbe Eingabe unter denselben Richtlinien führt zur selben Entscheidung. Das ist die Grundlage von Prüfbarkeit.

Fail-closed. Im Zweifel blockiert das System, statt durchzuwinken. Fehlt eine Information, versagt ein Baustein oder ist eine Zuordnung nicht eindeutig, ist die sichere Antwort „Nein/Stopp", nicht „Weiter".

Funktionstrennung. Wer einen Vorgang startet, gibt ihn nicht selbst frei. Wer eine Richtlinie verfasst, gibt die dadurch betroffene Hochrisiko-Aktion nicht frei. Diese Trennung ist technisch erzwungen, nicht nur organisatorisch empfohlen.

Lückenloser Audit-Trail. Jede sicherheitsrelevante Aktion erzeugt einen unveränderlichen, kryptografisch verketteten Nachweis. Nachträgliche Manipulation ist erkennbar.

> **Beweisbar statt plausibel** heißt: Der Wert des Systems liegt nicht darin, dass es meistens richtig liegt, sondern darin, dass jede Entscheidung durch geprüfte Regeln gedeckt und jederzeit belegbar ist.

Architektur des Kontroll-Plane

Für Architekten und Security-Engineering: ZERYON ist als lokale Desktop-Anwendung mit strikt geschichteter Architektur aufgebaut. Die Kontrolllogik liegt vollständig im Backend; die Oberfläche kann keine Sicherheitsentscheidung treffen oder umgehen.

Schichtenmodell

```
React-Oberfläche      Anzeige, Formulare, Navigation — keine Regel-Logik
   ↓ typisierte Kommandos
Application Services   Transaktionen, Orchestrierung
   ↓
Domain-Schicht        Policy-Engine · State Machine · Model-Provider (reine Regeln)
   ↓
Repository-Schicht    SQL-Abbildung, keine Richtlinienentscheidungen
   ↓
SQLite                Persistenz (Foreign Keys erzwungen)
```

Die Grenzen sind bewusst hart gezogen: Die Oberfläche schreibt **nicht** direkt in die Datenbank und wertet **keine** Richtlinien aus. Autorisierung, Richtlinienprüfung, Zustandsübergänge und die Ledger-Fortschreibung passieren ausschließlich im Backend. Clientseitige Prüfungen existieren nur als Komfort für die Bedienung, nie als Sicherheitsgrenze.

Backend-autoritative Autorisierung

Jedes sicherheitsrelevante Kommando durchläuft einen zentralen Autorisierungsdienst, der Rollen und Berechtigungen bei **jedem** Aufruf frisch aus der Datenbank liest. Der Entzug einer Rolle wirkt dadurch sofort — ohne dass sich der betroffene Nutzer neu anmelden muss. Sessions sind serverseitig gültig, mit Leerlauf- und Gesamt-Ablauf sowie sofortigem Widerruf.

Lokal zuerst, minimale Angriffsfläche

ZERYON läuft lokal (Tauri, Rust-Backend, SQLite) und benötigt für den Kernbetrieb keine Cloud-Anbindung. Das reduziert die Angriffsfläche und hält sensible Daten im Haus. Die Anwendung nutzt bewusst nur minimale Systemrechte, keine Shell-Ausführung und keinen unbeschränkten Dateizugriff. Für den Direktvertrieb wird sie signiert und von Apple notarisiert ausgeliefert.

Die fünf Kontrollmechanismen im Detail

Die Doktrin wird von fünf ineinandergreifenden Mechanismen getragen. Jeder ist für sich prüfbar; zusammen bilden sie das Kontroll-Plane.

1. Deterministische Policy-Engine

Die Policy-Engine bewertet jede vorgeschlagene Aktion gegen die aktiven Richtlinien. Richtlinien bestehen aus Bedingungen (Operatoren wie `eq`, `gt`, `contains`, `in`, kombiniert über `all` / `any` / `not`) und einem Effekt: `allow`, `deny` oder `require_approval`.

Entscheidend ist die feste Rangfolge bei mehreren Treffern:

```
deny  >  require_approval  >  allow
```

Ein `deny` gewinnt gegen jede höherpriorisierte Erlaubnis und erzeugt **niemals** eine Freigabeanfrage — eine Blockade lässt sich also nicht durch eine nachgelagerte Genehmigung aushebeln. Für nicht abgedeckte Aktionen ist das Verhalten explizit und dokumentiert; ein optionaler Fail-closed-Modus eskaliert un-geregelte Hochrisiko-Aktionen zur Freigabepflicht, statt sie stillschweigend durchzulassen.

2. State Machine mit erzwungenen Übergängen

Jede Ausführung durchläuft eine explizite Zustandsmaschine (`erstellt → laufend → wartet auf Freigabe → abgeschlossen / blockiert / abgelehnt / fehlgeschlagen / abgebrochen`). Ungültige Übergänge werden im Backend hart abgewiesen. Terminale Zustände haben keine ausgehenden Übergänge — eine bereits abgeschlossene, abgelehnte oder blockierte Ausführung kann nicht reaktiviert werden. Ein Freigabeschritt lässt sich nicht überspringen.

3. Freigabe-Workflow mit erzwungener Funktionstrennung

Verlangt eine Richtlinie eine Freigabe, pausiert die Engine vor der Aktion und legt eine Freigabeanfrage an. Zwei Trennungsregeln sind technisch erzwungen: Der Starter eines Vorgangs kann ihn **nicht selbst** freigeben, und der Autor oder Publisher einer auslösenden Richtlinie kann die dadurch betroffene Hochrisiko-Aktion **nicht** freigeben. Ist die Identität des Starters nicht eindeutig zuordenbar, wird die Freigabe fail-closed abgelehnt, statt die Prüfung zu überspringen. Abgelehnte Vorgänge enden terminal; die Aktion wird nicht ausgeführt. Ausstehende Freigaben verfallen nach einer Frist und sind danach nicht mehr aktionierbar.

4. Manipulationssicherer Audit-Ledger

Jedes sicherheitsrelevante Ereignis wird in einen append-only Ledger geschrieben — eine kryptografisch verkettete Kette (jeder Eintrag bindet den Hash seines Vorgängers). Drei Eigenschaften machen ihn belastbar:

- **Append-only auf Datenbankebene:** Änderungen und Löschungen an den Ledger-Tabellen werden durch Datenbank-Trigger unterbunden, nicht nur durch Anwendungslogik.
- **Verschlüsselt geschlüsselte Kette:** Der Ereignis-Hash ist mit einem Schlüssel (HMAC) gesichert, der außerhalb der Datenbank in der Schlüsselverwaltung des Betriebssystems liegt. Schreibzugriff auf die reine Datenbankdatei genügt damit nicht, um die Kette unbemerkt neu zu berechnen.
- **Prüfbar:** Eine Verifikationsfunktion rechnet die gesamte Kette nach und meldet Sequenz- oder Hash-Brüche. Bestehende Ketten bleiben ohne Migration gültig.

5. Rollenbasierte Zugriffskontrolle (RBAC)

ZERYON kennt abgestufte Rollen — u. a. Workspace-Administrator, KI-Systemarchitekt, Operator, Freigabeverantwortlicher, Auditor und Betrachter — mit fein granularen Berechtigungen. Die Anmeldung ist lokal (Argon2id-Passwort-Hashing); Passwörter liegen nie im Klartext in Datenbank, Ledger oder Export. Sensible Kommandos sind an konkrete Berechtigungen gebunden und werden serverseitig erzwungen.

Governance-by-Design & EU AI Act

Für Compliance- und Risikoverantwortliche: ZERYON ist so gebaut, dass regulatorische Kernanforderungen nicht nachträglich „aufgesetzt", sondern in die Architektur eingebaut sind. Das ist der Unterschied zwischen Governance-by-Design und Governance-as-Documentation.

Die folgende Zuordnung ist eine architektonische Einordnung, keine Rechtsberatung. Die konkrete Einstufung eines KI-Systems und die daraus folgenden Pflichten hängen vom Einsatzkontext ab und sind mit den zuständigen Stellen zu bewerten.

Regulatorische Anforderung (u. a. EU AI Act)	Entsprechung in ZERYON
Menschliche Aufsicht (Human Oversight)	Erzwungener Freigabe-Workflow mit Funktionstrennung; KI schlägt vor, der Mensch entscheidet bei Hochrisiko
Protokollierung / Record-Keeping	Append-only, kryptografisch verketteter Audit-Ledger über jeden sicherheitsrelevanten Schritt
Transparenz & Nachvollziehbarkeit	Deterministische, regelbasierte Entscheidungen mit dokumentierter Rangfolge; jede Aktion einer Identität/Rolle/Richtlinie zuordenbar
Risikomanagement	Richtlinien-gesteuerte Klassifikation; fail-closed als Grundverhalten; Deny nicht umgehbar
Robustheit & Sicherheit	Backend-autoritative Autorisierung, erzwungene Zustandsmaschine, minimale Angriffsfläche, signierte/notarisierte Auslieferung
Datenschutz (DSGVO)	Lokaler Betrieb, keine Cloud-Pflicht; Secrets nie im Klartext; kontrollierter Export

Der praktische Mehrwert: Nachweise entstehen **während** des Betriebs, nicht erst zur Audit-Vorbereitung. Statt Protokolle zu rekonstruieren, exportiert man eine bereits vollständige, verifizierbare Kette.

Sicherheit, Betrieb und Reifegrad

Sicherheit als Grundhaltung

ZERYON folgt durchgängig dem Fail-closed-Prinzip: Kontrollen fallen im Fehlerfall zu, nicht auf. Die Oberfläche ist keine Sicherheitsgrenze; alle Entscheidungen sind backend-autoritativ. Der Audit-Ledger ist auf Datenbankebene unveränderlich und kryptografisch geschlüsselt. Diese Eigenschaften wurden in einem gezielten Sicherheits- und Korrektheits-Review verifiziert und gehärtet.

Ehrliche Abgrenzung des Reifegrads

Glaubwürdigkeit entsteht durch Präzision, nicht durch Superlative. Der aktuelle Stand:

- **Lokaler Einzel-Workspace.** ZERYON ist als lokale, revisionssichere Kontrollumgebung ausgelegt. Multi-Tenancy und cloudseitige Identitätsföderation (OIDC/Entra ID) sind bewusst nicht Teil des aktuellen Funktionsumfangs.
- **Ausführungsmodell.** Aktionen werden **simuliert** ausgeführt — es entstehen keine realen Zahlungen, E-Mails oder externen Seiteneffekte. Das macht ZERYON zur sicheren Kontroll- und Nachweisumgebung, in der Governance geprüft wird, bevor reale Wirkungen angebunden werden.
- **Model-Provider.** Ein deterministischer Referenz-Provider ist ausführbar; weitere Provider (lokales LLM, governter externer Adapter) sind vorbereitet und bleiben an dieselben Governance-Gates gebunden — es gibt keine Umgehungspfade um die Kontrollen.

Diese Abgrenzung ist kein Manko, sondern Ausdruck der Doktrin: Erst die Kontrolle beweisbar machen, dann die Wirkung anbinden.

Handlungsempfehlung

Wer KI über das Experimentieren hinaus in wert- oder risikorelevante Prozesse bringen will, braucht kein größeres Modell, sondern eine beweisbare Kontrollschicht. Drei konkrete Schritte:

1. **Governance-Anforderungen vor Use-Cases definieren.** Legen Sie fest, welche KI-Aktionen eine erzwungene Freigabe, welche eine harte Blockade und welche einen lückenlosen Nachweis erfordern — bevor Sie den ersten produktiven Use-Case anbinden.
2. **Kontrolle von Modell trennen.** Behandeln Sie das Modell als Vorschlagsgeber und die Kontroll- und Nachweisschicht als eigenständige, deterministische Instanz. Nur so bleibt Ihre Governance unabhängig vom jeweils eingesetzten Modell.
3. **Nachweisführung mitlaufen lassen.** Setzen Sie auf Protokolle, die während des Betriebs entstehen und manipulationssicher sind — nicht auf nachträglich zusammengetragene Logs.

ZERYON Control Studio setzt diese drei Schritte als Produkt um: als lokale, deterministische Kontrollumgebung, die aus KI-Vorschlägen beweisbar kontrollierte, revisionssichere Vorgänge macht.

> **Beweisbar statt plausibel.** Determinismus. Fail-closed. KI schlägt vor, das System entscheidet. Lückenloser Audit-Trail.

Kontakt: Predrag Gasic · Prega Design — für eine Demonstration des Kontroll-Plane, eine Architektur-Vertiefung oder eine Einordnung entlang Ihrer konkreten Governance-Anforderungen.

Dieses Whitepaper beschreibt Architektur und Funktionsweise von ZERYON Control Studio 2.4.0. Regulatorische Zuordnungen sind Einordnungen, keine Rechtsberatung; die konkrete Bewertung eines KI-Systems hängt vom Einsatzkontext ab.