UNABHÄNGIGE FORSCHUNGSMONOGRAPHIE

Design, Implementierung und Evaluation eines AI-nativen Operating Systems für einen Solo-Operator

Eine Design-Science-Research-Studie zu Human-in-the-Loop-Governance, lokaler KI und autonomer Geschäftsprozessautomatisierung

Autor: Julien Marschall

Status: Unabhängige Forschungsmonographie · Public Research Edition · Review Candidate

Institutionelle Zugehörigkeit: Unabhängiger Forscher / Entwickler

Version: 1.0 · Public Research Edition · Review Candidate · August 2026

Erhebungsstand des Artefakts: 22. August 2026

Einordnung der Arbeit

Diese unabhängige Forschungsmonographie wurde außerhalb eines institutionellen Hochschulstudiengangs erstellt. Sie orientiert sich hinsichtlich Forschungsdesign, methodischer Struktur, wissenschaftlicher Dokumentation und Umfang an einer Masterarbeit im Bereich Wirtschaftsinformatik / Information Systems. Sie wurde jedoch weder im Rahmen eines Hochschulstudiums eingereicht noch stellt sie eine akademische Abschlussarbeit dar. Mit ihrer Veröffentlichung ist kein akademischer Grad verbunden.

Hinweis zur Public Research Edition

Diese öffentliche Fassung bewahrt Forschungsfragen, Methodik, empirische Kernergebnisse, Negativergebnisse und abgeleitete Designprinzipien. Wettbewerbs-, sicherheits- oder betriebsrelevante Implementierungsdetails werden bewusst abstrahiert oder ausgelassen. Die Kürzung verändert nicht die Richtung der wissenschaftlichen Befunde, soll aber vermeiden, dass die Publikation als Reproduktionsanleitung für produktive interne Systeme dient.

Publikations- und Transparenzhinweis

Diese Monographie dokumentiert die Entwicklung und Evaluation eines eigenständig entwickelten Forschungs- und Softwareartefakts. Sie wurde außerhalb eines Hochschulstudiengangs erstellt und begründet keinen akademischen Grad. Die Bezeichnung Forschungsmonographie beschreibt die methodisch strukturierte, quellenbasierte Untersuchung des Löwen Codex OS und ist ausdrücklich nicht als universitäre Abschlussbezeichnung zu verstehen.

Autorschaft und Doppelrolle

Julien Marschall ist Entwickler, Betreiber und Autor des untersuchten Artefakts. Diese Doppelrolle ermöglicht einen ungewöhnlich tiefen Zugriff auf Quellcode, Betriebszustände und Entwicklungsgeschichte, erzeugt aber zugleich ein Bias-Risiko. Die Arbeit begegnet diesem Risiko durch eine Evidenzhierarchie, explizite Negativergebnisse, die Trennung von Betreiberbeobachtung und maschineller Evidenz sowie eine zurückhaltende Formulierung kausaler Aussagen.

Einsatz generativer KI bei der Erstellung

Generative KI wurde als Werkzeug für Strukturierung, sprachliche Ausarbeitung, Zusammenfassung, Literaturorientierung und redaktionelle Überarbeitung eingesetzt. Die empirische Primärevidenz stammt aus Quellcode, Logs, Zustandsdateien, automatisch erzeugten Reports und der dokumentierten Entwicklungsgeschichte. KI-generierte Formulierungen werden nicht als eigenständige empirische Quelle behandelt. Verantwortung für Forschungsgegenstand, Datenbereitstellung, Freigabe der Argumentationslinie und Veröffentlichung liegt beim Autor.

Review- und Veröffentlichungsstatus

Version 1.0 ist als Public Research Edition · Review Candidate für eine externe fachliche Begutachtung vorgesehen. Eine spätere Kennzeichnung als extern begutachtet oder peer reviewed erfolgt ausschließlich nach tatsächlich dokumentierter Prüfung. Änderungen aus Reviews sollen versioniert und in einem Änderungsprotokoll festgehalten werden.

Abstract

Die vorliegende Arbeit untersucht den Entwurf, die Implementierung und die Evaluation des Löwen Codex OS, eines auf einem einzelnen Windows-Host betriebenen, AI-nativen Betriebssystems zur Unterstützung und Automatisierung heterogener unternehmerischer Prozesse. Methodisch folgt die Untersuchung dem Design-Science-Research-Paradigma. Das entwickelte System wird als IT-Artefakt analysiert, dessen Architektur, Automatisierungsgrad, Human-in-the-Loop-Mechanismen, Governance-Strukturen und operative Wirksamkeit anhand von Quellcode, Laufzeitprotokollen, Zustandsdateien, automatisch erzeugten Reports und einer dokumentierten Entwicklungshistorie bewertet werden.

Die Ergebnisse zeigen eine historisch gewachsene Architektur aus mehreren Prozessketten, die über eingebettete relationale Datenhaltung-Datenbanken, dateibasierte Zustandsdaten-Zustände, lokale HTTP-Schnittstellen, einen Reverse Proxy-Webroot und den Windows-Taskplaner gekoppelt sind. Lokale Sprachmodelle werden überwiegend als text- und dateibasierte Zustandsdaten-erzeugende Komponenten eingesetzt; die eigentliche Aktionsausführung liegt weitgehend in deterministischem Python- und JavaScript-Code. Der Automatisierungsgrad variiert stark zwischen den Prozessen. Cold-Mail-Versand, Follow-ups, Content-Publishing und Teile der Telefonie erreichen hohe Automatisierungsgrade, während individuelle Antwortkommunikation, Lead-Diagnose und verschiedene Entwicklungsfunktionen stärker menschlich kontrolliert bleiben.

Zentral ist die empirisch belegte Trennung von technischer Autonomie, betrieblicher Wirksamkeit und Governance. Ein hoher Automatisierungsgrad führt nicht automatisch zu wirksamen Ergebnissen: über 19.000 dokumentierte Mail-Sendungen erzeugten über 1.000 Klicks und mehreren Dutzend Antworten, jedoch keinen kampagnenzuordenbaren Termin. Im Voice-Pfad wurden über 1.500 Wahlversuche protokolliert, von denen nahezu allen als nicht erreicht klassifiziert wurden; ein Termin wurde nicht gebucht. Gleichzeitig zeigt die Governance-Analyse, dass technisch sauber implementierte Freigabe- und Sandbox-Mechanismen teilweise gerade in wenig

oder nicht genutzten Komponenten liegen, während produktive Prozesse mit Außenwirkung teilweise ohne menschliche Einzelbestätigung operieren.

Aus den Ergebnissen werden Designprinzipien für AI-native Operator-Systeme abgeleitet: deterministische Grenzen für kritische Aktionen, prozessbezogene statt globale Autonomiebewertung, Default-Deny-Governance, beobachtbare Zustände, getrennte Service-Healthchecks, aktuelle Wissensindizes, wirksame Kill-Switches und eine explizite Unterscheidung zwischen Systemaktivität und Geschäftswirkung. Die Arbeit zeigt damit, dass AI-native Systeme nicht primär anhand der Anzahl integrierter KI-Komponenten, sondern anhand ihrer kontrollierbaren, beobachtbaren und nachweisbar wirksamen Prozessketten bewertet werden sollten.

Schlüsselwörter: Design Science Research; AI-native Operating System; Human-in-the-Loop; AI Governance; lokale KI; Automatisierung; Agentic AI; Observability; Solo-Operator.

Inhaltsverzeichnis

Kapitel I – Einleitung und Forschungsdesign

1.1 Problemstellung und Motivation

Generative künstliche Intelligenz hat sich von einer isolierten Assistenztechnologie zu einem Baustein komplexer Informationssysteme entwickelt. In praktischen Unternehmensumgebungen werden Sprachmodelle nicht mehr ausschließlich für einzelne Texte oder Analysen eingesetzt, sondern mit Datenbanken, Automatisierungslogik, Kommunikationskanälen, Content-Systemen und operativen Benutzeroberflächen verbunden. Dadurch entsteht eine neue Systemklasse, die in dieser Arbeit als AI-native Operating System bezeichnet wird: ein Informationssystem, in dem KI-Funktionen nicht als nachträgliche Zusatzfunktion auftreten, sondern in zentrale Informations-, Entscheidungs- und Ausführungsprozesse eingebettet sind.

Für kleine Unternehmen und insbesondere Solo-Operatoren ist diese Entwicklung besonders relevant. Während größere Organisationen spezialisierte Rollen für Vertrieb, Marketing, Softwareentwicklung, Datenanalyse, Kundenservice und Governance einsetzen können, muss ein Einzeloperator dieselben Funktionsbereiche mit begrenzter Zeit und begrenzten personellen Ressourcen koordinieren. Automatisierung verspricht hier eine Hebelwirkung, verschiebt jedoch zugleich die Kontrollfrage: Je mehr Prozessschritte ohne menschliche Einzelentscheidung ausgeführt werden, desto wichtiger werden technische Grenzen, Zustandsbeobachtung, Fehlermanagement und nachvollziehbare Freigabelogiken.

Das in dieser Arbeit untersuchte Löwen Codex OS entstand aus einer mehrjährigen praktischen Entwicklung. Der ursprüngliche Forschungsentwurf aus dem Jahr 2025 war deutlich breiter angelegt und verband unter anderem Persona-Architekturen, Prompt-Stacks, Growth-Marketing, Neuromarketing, Responsible AI sowie spekulative Quantum- und Hawkins-Bezüge. Mehrere Abschnitte waren noch als offene [WRITE]-Blöcke markiert; konkrete Forschungsfragen und der methodische Rahmen waren nicht ausformuliert. Diese frühe Fassung wird deshalb nicht als empirische Evidenz verwendet, sondern als Dokument der explorativen Konzeptionsphase. Der aktuelle Forschungsgegenstand ist demgegenüber ein implementiertes System mit Quellcode, Logs, Zustandsdateien und realen Betriebsdaten.

Die wissenschaftliche Relevanz entsteht gerade aus der Differenz zwischen Entwurfsintention und Betriebsrealität. Das System enthält lokale Sprachmodelle, eine Mail- und Outreach-Engine, einen Voice-Stack, eine Content-Engine, CRM- und Lead-Funktionen, Wissens- und Memory-Komponenten, Dashboards, Backups und verschiedene Governance-Mechanismen. Gleichzeitig dokumentieren die Exporte erhebliche Negativergebnisse: fehlende Versionskontrolle, unvollständige Healthchecks, veraltete Wissensindizes, Governance-Gates ohne Nutzung, automatisierte Prozesse ohne Einzelbestätigung sowie hohe Aktivität ohne nachweisbare Conversion. Diese Kombination erlaubt eine Evaluation, die nicht nur Funktionsumfang, sondern tatsächliche Wirksamkeit und Kontrollierbarkeit untersucht.

1.2 Forschungslücke

Die Forschung zu Design Science, Human-Automation Interaction, Human-AI Interaction und AI Governance liefert etablierte Perspektiven auf Artefaktentwicklung, Automatisierungsgrade und menschliche Aufsicht. Hevner et al. (2004) verstehen Design Science als problemorientierte Forschung, in der Wissen durch die Konstruktion und Anwendung zweckgerichteter Artefakte entsteht. Peffers et al. (2007) operationalisieren diesen Ansatz als sechsstufigen DSRM-Prozess von Problemidentifikation über Design und Demonstration bis Evaluation und Kommunikation. Parasuraman, Sheridan und Wickens (2000) zeigen, dass Automatisierung nicht eindimensional betrachtet werden sollte, sondern unterschiedliche Informations- und Aktionsfunktionen auf unterschiedlichen Stufen automatisiert werden können. Amershi et al. (2019) betonen für Human-AI-Systeme insbesondere Korrigierbarkeit, Transparenz, vorsichtige Anpassung und globale Kontrollmöglichkeiten. Das NIST AI RMF ordnet Governance als querschnittliche Funktion über den Lebenszyklus von KI-Systemen ein; auch der EU AI Act formuliert für Hochrisiko-KI das Prinzip wirksamer menschlicher Aufsicht.

Weniger klar ist, wie diese Perspektiven in einem realen, heterogenen, von einer einzelnen Person betriebenen AI-nativen Gesamtsystem zusammenspielen. Insbesondere fehlt im konkreten Untersuchungsfall eine belastbare Antwort auf vier Fragen: Wie wird ein solches System tatsächlich architektonisch integriert? Welche

Prozesse sind nur KI-unterstützt und welche operieren autonom? Wo ist Human-in-the-Loop technisch erzwungen und wo lediglich dokumentierte Absicht? Und wie unterscheiden sich Systemaktivität, technische Autonomie und Geschäftswirkung im realen Betrieb? Die Arbeit adressiert diese Lücke als tiefgehende Artefakt- und Fallstudie, ohne daraus eine statistische Generalisierbarkeit für alle AI-Systeme abzuleiten.

1.3 Zielsetzung und Forschungsfragen

Ziel der Arbeit ist die systematische Rekonstruktion und Evaluation des Löwen Codex OS als Design-Science-Artefakt. Untersucht werden Architektur, lokale KI, Prozessautomatisierung, Human-in-the-Loop-Governance, Observability und operative Resultate. Auf dieser Basis sollen übertragbare Designprinzipien für vergleichbare AI-native Operator-Systeme abgeleitet werden.

Forschungsfrage	Formulierung
RQ1	Welche Architektur- und Designprinzipien ermöglichen die Integration heterogener KI-gestützter Geschäftsprozesse in ein von einem einzelnen Operator betriebenes AI-native Operating System?
RQ2	Welche Geschäftsprozesse lassen sich innerhalb eines solchen Systems zuverlässig automatisieren, und an welchen Stellen entstehen technische oder operative Grenzen zunehmender Autonomie?
RQ3	Wie wirksam sind Human-in-the-Loop-, regelbasierte und technische Governance-Mechanismen bei der Begrenzung autonomer KI-Aktionen in einem produktiven AI-native System?
RQ4	Welche Erkenntnisse über Zuverlässigkeit, Wirksamkeit und Weiterentwicklung AI-nativer Betriebssysteme lassen sich aus Betriebsdaten, Fehlerfällen, Negativergebnissen und menschlichen Eingriffen ableiten?

1.4 Wissenschaftlicher und praktischer Beitrag

Der wissenschaftliche Beitrag liegt nicht in der Behauptung eines allgemein überlegenen Automatisierungsmodells, sondern in der empirisch dokumentierten Zerlegung eines realen AI-nativen Systems. Die Arbeit verbindet DSR mit einer

technischen Fallstudie und zeigt, dass Autonomie, Wirksamkeit und Governance analytisch getrennt werden müssen. Der praktische Beitrag besteht in konkreten Designprinzipien für Systeme, die lokale KI, deterministische Automatisierung und menschliche Freigabe kombinieren.

1.5 Aufbau der Arbeit

Kapitel II entwickelt die theoretischen Grundlagen. Kapitel III beschreibt Methodik und Evidenzmodell. Kapitel IV rekonstruiert Entwicklung und Architektur des Artefakts. Kapitel V analysiert die produktiven Prozessketten. Kapitel VI evaluiert Automatisierungsgrad, Governance, Zuverlässigkeit und operative Resultate. Kapitel VII beantwortet die Forschungsfragen, leitet Designprinzipien ab und diskutiert Limitationen und zukünftige Forschung.

Kapitel II – Theoretische Grundlagen und Forschungsstand

2.1 Design Science Research

Design Science Research (DSR) ist für die vorliegende Untersuchung geeignet, weil der Forschungsgegenstand nicht lediglich beobachtet, sondern als zweckgerichtetes Informationssystem entwickelt wurde. Hevner et al. (2004) stellen die Konstruktion und Evaluation innovativer Artefakte in den Mittelpunkt. Der Erkenntnisgewinn entsteht durch das Bauen und Anwenden des Artefakts in einem relevanten Problemkontext. Peffers et al. (2007) konkretisieren DSR in sechs Aktivitäten: Problemidentifikation und Motivation, Definition der Lösungsziele, Design und Entwicklung, Demonstration, Evaluation und Kommunikation. Die vorliegende Arbeit rekonstruiert diese Aktivitäten teilweise retrospektiv, da die technische Entwicklung vor der formalen wissenschaftlichen Endauswertung begonnen hat.

2.2 Automatisierung als mehrdimensionale Eigenschaft

Parasuraman, Sheridan und Wickens (2000) unterscheiden Automatisierung entlang von Informationsaufnahme, Informationsanalyse, Entscheidungsauswahl und Aktionsimplementierung. Diese Perspektive ist für AI-native Systeme wichtiger als ein globales Etikett wie „autonom“, weil ein Prozess beispielsweise Informationen automatisch sammeln und analysieren kann, während die finale Aktion weiterhin menschlich ausgelöst wird. Umgekehrt kann ein deterministischer Scheduler eine Aktion vollständig autonom ausführen, obwohl das Sprachmodell selbst keinerlei Entscheidungsbefugnis besitzt. Für die Fallstudie wird deshalb ein prozessbezogenes L0-L5-Schema verwendet.

2.3 Human-in-the-Loop und Human-AI Interaction

Human-in-the-Loop (HITL) bezeichnet in dieser Arbeit nicht bloß die abstrakte Möglichkeit menschlicher Kontrolle, sondern die konkrete Position des Menschen in einer Prozesskette. Entscheidend ist, ob eine menschliche Handlung je Vorgang technisch erforderlich ist. Diese Definition verhindert, dass einmalige Konfigurationen oder organisatorische Vorsätze fälschlich als laufende Aufsicht klassifiziert werden.

Amershi et al. (2019) betonen unter anderem die Korrigierbarkeit von KI-Ausgaben, die Erklärung von Systemverhalten, vorsichtige Anpassung sowie globale Kontrollmöglichkeiten. Diese Kriterien werden später mit den realen Mechanismen des Löwen Codex OS verglichen.

2.4 AI Governance und Responsible AI

Governance umfasst Richtlinien, technische Kontrollen, Verantwortlichkeiten, Dokumentation, Überwachung und Reaktionsmechanismen. Das NIST AI RMF beschreibt GOVERN als querschnittliche Funktion, die Risikomanagement über den gesamten Lebenszyklus unterstützt. Für die vorliegende Arbeit ist besonders relevant, ob Governance nur als Prompt oder Dokumentation existiert oder als Code-Gate technisch durchgesetzt wird. Der EU AI Act macht im Kontext von Hochrisiko-KI deutlich, dass menschliche Aufsicht wirksam und dem Risiko sowie Autonomiegrad angemessen ausgestaltet werden soll. Die hier untersuchten Systeme werden nicht pauschal als Hochrisiko-KI klassifiziert; die Norm dient vielmehr als regulatorischer Referenzpunkt für das Prinzip wirksamer Aufsicht.

2.5 Lokale KI und hybride Systemarchitekturen

Lokale KI verschiebt Systemgrenzen. Inferenz kann ohne Übertragung von Nutzdaten an einen externen Modellanbieter erfolgen, gleichzeitig entstehen neue operative Abhängigkeiten von GPU-Speicher, Modellverfügbarkeit, lokalen Laufzeiten und eigener Infrastruktur. Im untersuchten System laufen zentrale Sprach-, Embedding-, Vision- und Speech-Komponenten lokal. Cloud-Dienste bleiben für einzelne Kommunikations- und Infrastrukturaufgaben relevant. Die resultierende Architektur ist daher nicht rein lokal, sondern hybrid.

2.6 Agentic AI versus deterministische Orchestrierung

Der Begriff „Agent“ wird in Praxis und Forschung uneinheitlich verwendet. Für die Analyse wird deshalb zwischen modellgetriebener Aktionsautonomie und prompt-parametrierten Funktionen unterschieden. Der Codeexport zeigt für den zentralen Backend- und Content-Bestand keine allgemeine agentische Tool-Calling-Architektur: Modellaufrufe erzeugen Text oder strukturierte Daten; Aktionen, Reihenfolge,

Wiederholungen und Schreibvorgänge werden überwiegend deterministisch in Python oder JavaScript gesteuert. Diese Unterscheidung ist zentral, weil ein System als Ganzes hochautomatisiert sein kann, obwohl das LLM selbst keine Werkzeuge auswählt.

2.7 Observability, Reliability und Fail-Safe-Design

Automatisierung ist nur dann kontrollierbar, wenn der Systemzustand beobachtbar ist. Logs, Healthchecks, Metriken, Audit Trails, Backups und klare Fehlerzustände sind daher nicht bloß Betriebsdetails, sondern Governance-Infrastruktur. Ein Healthcheck muss die Funktionsfähigkeit des konkreten Dienstes prüfen und darf Portbelegung nicht mit korrektem Servicebetrieb gleichsetzen. Ebenso müssen Reports ihre Messdefinitionen offenlegen, weil ein formal grüner Bericht operative Fehler verdecken kann. Diese Prinzipien werden im Fall des Löwen Codex OS anhand konkreter Negativergebnisse untersucht.

2.8 Analytisches Rahmenmodell

Aus den theoretischen Perspektiven wird ein dreidimensionales Rahmenmodell abgeleitet. Dimension A ist der Automatisierungsgrad eines Prozesses. Dimension B ist seine betriebliche Wirksamkeit anhand beobachtbarer Zielmetriken. Dimension C ist die Governance-Stärke, gemessen an technisch erzwungenen Gates, Nachvollziehbarkeit, Korrekturmöglichkeiten und Notabschaltung. Die drei Dimensionen werden bewusst unabhängig bewertet. Ein Prozess kann hochautomatisiert, aber unwirksam sein; er kann wirksam, aber schwach kontrolliert sein; oder stark gegated, aber praktisch ungenutzt bleiben.

2.9 AI-native Informationssysteme als sozio-technische Systeme

Der Begriff AI-native wird in dieser Arbeit funktional verwendet. Gemeint ist ein Informationssystem, dessen Kernprozesse so entworfen sind, dass probabilistische KI-Komponenten, deterministische Software, persistente Daten, externe Dienste und menschliche Entscheidungen dauerhaft miteinander interagieren. Diese Definition grenzt das Untersuchungsobjekt von klassischen Anwendungen ab, in die lediglich ein Chatbot als zusätzliche Oberfläche eingebaut wird. Beim Löwen Codex OS greifen lokale Modelle in NLU, Retrieval, Antwort-Triage, Content-Generierung, Vision und einzelne

Diagnosefunktionen ein, während Aktionsausführung und Prozesszustand überwiegend durch konventionellen Code bestimmt werden.

Die sozio-technische Perspektive ist notwendig, weil beobachtete Leistung nicht allein aus Modellgüte erklärt werden kann. Ein System kann über leistungsfähige Modelle verfügen und dennoch an Zustandskonsistenz, Ressourcenknappheit, falschen Healthchecks oder unzureichender Governance scheitern. Umgekehrt können relativ kleine lokale Modelle innerhalb eng begrenzter, deterministisch abgesicherter Funktionen zuverlässig eingesetzt werden. Die Einheit der Analyse ist deshalb nicht das Modell, sondern die gesamte Prozesskette einschließlich Datenquelle, Orchestrierung, Schnittstellen, Operator und Zielmetrik.

Für einen Solo-Operator verschärft sich diese Betrachtung. Rollen, die in größeren Organisationen institutionell getrennt wären – Entwicklung, Betrieb, Vertrieb, Monitoring und Freigabe – liegen in einer Person. Technische Kontrollen übernehmen damit teilweise Funktionen organisatorischer Arbeitsteilung. Sandbox, read-only Zugriff, Freigabeparameter, Audit-Logs und Backups sind nicht nur technische Bequemlichkeiten, sondern Ersatzmechanismen für fehlende organisatorische Trennung.

2.10 Agentic AI, Planung, Tool Use und Memory

Aktuelle Übersichtsarbeiten zu LLM-basierten Agenten ordnen agentische Systeme typischerweise entlang von Planung, Tool Use, Feedback und Memory. Huang et al. (2024) unterscheiden bei der Agentenplanung unter anderem Task Decomposition, Plan Selection, externe Module, Reflection und Memory. Zhang et al. (2024) zeigen, dass Memory als eigenständige Architekturkomponente verstanden werden muss, weil langfristige Interaktion ohne geeignete Speichermechanismen weder konsistent noch lernfähig ist. Diese Literatur bietet eine nützliche Folie, um den tatsächlichen Agentencharakter des Löwen Codex OS nicht zu überschätzen.

Im untersuchten Kernbestand existiert keine universelle Schleife, in der ein Sprachmodell frei Werkzeuge auswählt und seine eigene Aktionsfolge ausführt. Modellaufrufe liefern überwiegend Text oder dateibasierte Zustandsdaten; Ausführung, Retry, Statuswechsel und Schreiboperationen liegen in deterministischem Python- oder

JavaScript-Code. Damit kann das Gesamtsystem hochautomatisiert sein, obwohl das Modell selbst geringe Aktionsautonomie besitzt. Für die Governance ist diese Trennung zentral: Das Risiko entsteht häufig durch die Orchestrierung und nicht durch eine freie Werkzeugauswahl des Modells.

Memory ist ebenfalls mehrschichtig. Operative eingebettete relationale Datenhaltung-Zustände, dateibasierte Zustandsdaten-States, kurzfristiger Dialogkontext, Betreiberlog, RAG-Index und potenziell ratifizierte Wissenseinträge erfüllen unterschiedliche Funktionen. Die dokumentierte Freshness-Lücke des Wissensindex zeigt, dass Speichermechanismen nicht nur hinsichtlich Kapazität, sondern hinsichtlich Aktualität, Provenienz und Konfliktbehandlung evaluiert werden müssen.

2.11 Lokale und Edge-nahe Sprachmodelle

Die Literatur zu On-Device- und Edge-LLMs beschreibt Datenschutz, geringere Netzabhängigkeit und potenziell niedrigere Latenz als zentrale Vorteile lokaler Inferenz, stellt diesen aber begrenzte Rechen- und Speicherressourcen gegenüber. Xu et al. (2024) und Qu et al. (2024) diskutieren Kompression, Hardwarebeschleunigung und hybride Edge-Cloud-Architekturen als zentrale Gestaltungsfragen. Der Löwen Codex ist kein mobiles Edge-System im engeren Sinn, teilt aber dieselbe Grundspannung: lokale Verarbeitung erhöht Datenlokalität und Kostenkontrolle, bindet die Leistungsfähigkeit jedoch unmittelbar an verfügbare GPU-Ressourcen und lokale Betriebsstabilität.

Der dokumentierte VRAM-Thrashing-Fall ist hierfür besonders aufschlussreich. Sprach-, Bild- und Videomodelle konkurrierten auf derselben GPU. Die erfolgreiche Reparatur bestand nicht in einem größeren Modell, sondern in einer Phasentrennung der Pipeline. Damit wird lokale KI zu einer Scheduling- und Ressourcenfrage. Die theoretische Konsequenz lautet: Bei lokalen Multi-Modell-Systemen muss die Hardwarebelegung als Bestandteil der Prozessarchitektur modelliert werden.

2.12 AI Governance und risikobasierte Kontrolle

Das NIST AI RMF versteht GOVERN als querschnittliche Funktion über den gesamten AI-Lebenszyklus. Die Generative-AI-Erweiterung des NIST konkretisiert, dass Risiken nicht nur beim Modelltraining, sondern auch bei Anwendung, Integration und Nutzung entstehen. Für die vorliegende Arbeit wird daraus eine technische Operationalisierung

abgeleitet: Governance ist dann stark, wenn eine Regel als wirksame Systemgrenze existiert, beobachtbar ist und an einem realen Risikopunkt greift.

Die Arbeit unterscheidet drei Stärken von Governance. Deklarative Governance besteht aus Dokumentation, Promptregeln oder Betriebsdoktrin. Signalbasierte Governance erzeugt Prüfresultate oder Warnungen, deren Beachtung optional ist. Erzwingende Governance blockiert eine Aktion technisch, wenn Bedingungen nicht erfüllt sind. Diese Abstufung hilft, Mechanismen wie den nicht durchgesetzten LLM-Auditor, Browser-confirm, Verify-Gates und Test-before-Promote sauber voneinander zu unterscheiden.

Der EU AI Act dient in dieser Arbeit nicht zur pauschalen Einstufung des Gesamtsystems als Hochrisiko-KI. Relevanter ist sein Grundprinzip, menschliche Aufsicht wirksam und risikoadäquat zu gestalten. Für ein heterogenes Operator-System bedeutet dies, dass die Intensität der Aufsicht an Außenwirkung, Reversibilität, Datenrisiko und potenzieller rechtlicher Relevanz ausgerichtet werden sollte.

2.13 Design Science zwischen DSR und Action Design Research

Hevner et al. (2004) positionieren den Erkenntnisgewinn der Design Science im Bau und in der Anwendung eines Artefakts. Peffers et al. (2007) operationalisieren den Prozess in sechs Aktivitäten. Gregor und Hevner (2013) betonen zusätzlich, dass eine DSR-Arbeit ihren Wissensbeitrag klar positionieren und Artefaktbeschreibung, Evaluation und abstrahiertes Designwissen verbinden muss. Für die vorliegende Monographie ist diese Perspektive zentral, weil der Wert nicht in der bloßen Größe des Systems liegt, sondern in den aus seinem Betrieb abgeleiteten Designprinzipien.

Gleichzeitig weist das Projekt Eigenschaften von Action Design Research auf. Sein et al. (2011) argumentieren, dass Artefakte während Entwicklung und Nutzung durch ihren organisatorischen Kontext geformt werden und Evaluation nicht notwendigerweise erst nach Fertigstellung stattfindet. Der Löwen Codex entstand aus fortlaufender betrieblicher Nutzung; Fehler, Umbauten und neue Anforderungen beeinflussten das Artefakt kontinuierlich. Methodisch bleibt die Arbeit dennoch bei einer retrospektiv rekonstruierten DSR-Fallstudie, nutzt ADR aber als theoretische Erklärung dafür, warum Entwicklung und Evaluation im realen Betrieb ineinandergreifen.

Abbildung 2: DSR-Zyklus der vorliegenden Untersuchung

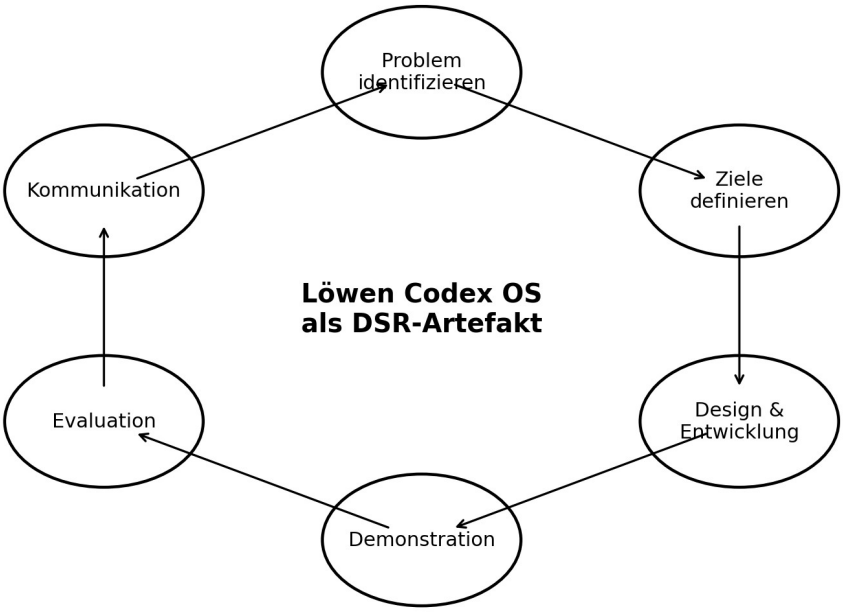
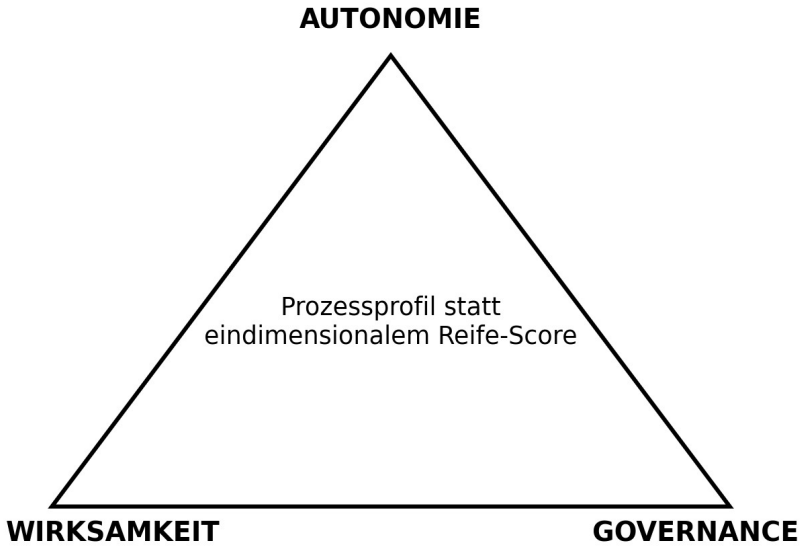


Abbildung 2: DSR-Zyklus der Untersuchung (eigene Darstellung).

Abbildung 3: Analytisches Dreieck der Evaluation



Jede Dimension wird unabhängig bewertet.

Abbildung 3: Autonomie, Wirksamkeit und Governance als getrennte Evaluationsdimensionen (eigene Darstellung).

Kapitel III – Forschungsmethodik

3.1 Forschungsdesign

Die Untersuchung kombiniert Design Science Research mit einer eingebetteten technischen Einzelfallstudie. Das Löwen Codex OS bildet das zentrale Artefakt. Die Analyse ist überwiegend retrospektiv: Entwicklungsschritte wurden vor der finalen wissenschaftlichen Strukturierung vorgenommen, anschließend jedoch durch eine systematische, ausschließlich lesende Bestandsaufnahme rekonstruiert. Die Evaluation verbindet statische Codeanalyse, Log- und Zustandsanalyse, Systemreports, Dateisystemmetadaten und Betreiberdokumentation.

3.2 DSRM-Anwendung

DSRM-Aktivität	Umsetzung in der Arbeit
Problemidentifikation	Begrenzte personelle Kapazität eines Solo-Operators bei gleichzeitig heterogenen Geschäftsprozessen.
Lösungsziele	Integration, Automatisierung, lokale KI, kontrollierbare Ausführung, zentrale Sichtbarkeit.
Design & Entwicklung	Iterative Entwicklung des Löwen Codex OS 2025-2026.
Demonstration	Produktive Prozessketten für Mail, Content, Voice, CRM, Knowledge und Backups.
Evaluation	Code, Logs, Reports, Betriebskennzahlen, Governance- und Negativergebnisanalyse.
Kommunikation	Dokumentation in der vorliegenden Arbeit und Evidenzanhängen.

3.3 Datenquellen und Evidenzklassen

Die Erhebung erfolgte lesend. Live-Datenbanken wurden nicht mit eingebettete relationale Datenhaltung geöffnet, um die dokumentierte Korruptionsschutz-Doktrin einzuhalten. Aussagen über Datenbankmengen stammen deshalb aus Schema-Lektüre oder systemeigenen Reports. Laufende Prozesse und offene Ports konnten aus der isolierten Analyseumgebung nicht direkt geprüft werden; Produktionsstatus wurde über frische Laufzeitartefakte rekonstruiert.

Klasse	Bedeutung	Verwendung
A	Rohdaten / reproduzierbarer Quellcode	stärkste technische Evidenz
B	Logs / Datenbankableitungen / Zustandsartefakte	Betriebs- und Ergebnisnachweis
C	Screenshot / automatisch erzeugter Report	sekundärer Betriebsnachweis
D	dokumentierte Betreiberbeobachtung	nur bedingt belastbar
E	Vermutung / Inferenz	nicht als gesicherter Befund verwendet

Im Evidenzregister des Exports wurden 122 Aussagen klassifiziert: 65 (53,3 %) als A, 29 (23,8 %) als B, 2 (1,6 %) als C, 18 (14,8 %) als D und 8 (6,6 %) als E. Damit basiert der überwiegende Teil der für Automatisierungsgrad, Governance und quantitative Betriebsergebnisse verwendeten Aussagen auf Code oder Logs.

3.4 Datenschutz und Forschungsethik

Personenbezogene Daten Dritter wurden aus dem Forschungsexport ausgeschlossen. Namen, E-Mail-Adressen, Telefonnummern und andere identifizierende Lead-Daten werden in dieser Arbeit nicht reproduziert. Secrets, Tokens, API-Schlüssel und Bankdaten werden ebenfalls nicht wiedergegeben. Sicherheits- und Datenschutzbefunde werden auf Struktur- und Mechanismenebene beschrieben.

3.5 Evaluationslogik

Die Evaluation nutzt vier Perspektiven: (1) Architektur- und Integrationsfähigkeit, (2) Automatisierungsgrad je Prozess, (3) Governance und menschliche Aufsicht sowie (4) betriebliche Wirksamkeit. Negative Ergebnisse werden ausdrücklich als Forschungsdaten behandelt. Ursachen werden nur dann als gesichert bezeichnet, wenn sie durch Code oder Logs belegt sind; ansonsten werden sie als Inferenz markiert.

3.6 Validität und Limitationen

Die interne Nachvollziehbarkeit ist durch Zeilenbelege, Logs und Evidenzklassen vergleichsweise hoch. Einschränkungen bestehen durch die Einzelfallnatur, die Beteiligung des Betreibers als Entwickler, fehlende randomisierte Kontrollgruppen, fehlende direkte Postmaster-Daten, teilweise fehlende Laufzeittelemetrie und fehlende

Versionskontrolle. Die Ergebnisse sind daher analytisch, nicht statistisch generalisierbar.

3.7 Operationalisierung der Evaluationsdimensionen

Die Dimension Autonomie wird anhand des realen Kontrollflusses operationalisiert. Für jeden Prozess wird geprüft, ob ein Mensch je Vorgang eingreifen muss, welche Bedingungen automatisch geprüft werden, welche Aktion extern wirksam wird und ob ein Modell lediglich Informationen erzeugt oder selbst Aktionsentscheidungen treffen kann. Die verwendete L0-L5-Skala ist eine fallbezogene Ordinalskala und wird nicht als extern validiertes Reifegradmodell dargestellt.

Wirksamkeit wird in Aktivitäts-, Zwischen- und Zielmetriken getrennt. Gesendete Mails, erzeugte Posts oder Wahlversuche sind Aktivitätsmetriken. Klicks, Replies und erfolgreich gerenderte Medien sind Zwischenmetriken. Termine, qualifizierte Übergaben oder erfolgreich veröffentlichte Inhalte bilden fachliche Zielzustände. Diese Trennung verhindert, dass ein Prozess aufgrund hoher Aktivität als erfolgreich klassifiziert wird, obwohl sein Zweck nicht erreicht wird.

Governance wird anhand technischer Erzwingung, Nachvollziehbarkeit, menschlicher Eingriffsmöglichkeit, Reversibilität und Notabschaltung bewertet. Die drei Dimensionen werden nicht zu einem Gesamtscore verdichtet. Gerade die Nicht-Aggregation ist methodisch gewollt, weil die Fallstudie Kombinationen sichtbar machen soll, die ein einzelner Reifegrad verdecken würde.

3.8 Retrospektive Rekonstruktion und Betreiberbias

Autor und Betreiber des Artefakts sind identisch. Daraus entsteht ein erhöhtes Risiko selektiver Wahrnehmung und nachträglicher Rationalisierung. Zur Begrenzung dieses Risikos werden Betreiberbeobachtungen grundsätzlich niedriger gewichtet als reproduzierbarer Code und maschinelle Laufzeitdaten. Quantitative Kernaussagen werden nach Möglichkeit auf Evidenz A oder B gestützt. Wo nur Betreiberbeobachtung oder Inferenz verfügbar ist, wird dies explizit gekennzeichnet.

Die Entwicklung wurde nicht als vorab registriertes Experiment gestartet. Viele Umbauten entstanden aus akuten Betriebsproblemen. Deshalb werden keine kausalen

Wirkungen behauptet, wenn Kontrollgruppe, Randomisierung oder stabile Vorher-Nachher-Bedingungen fehlen. Der VRAM-Umbau ist beispielsweise ein gut dokumentierter Engineering-Fall, aber kein randomisiertes Experiment. Diese Unterscheidung bewahrt die Arbeit davor, operative Erfahrung nachträglich als experimentelle Evidenz umzudeuten.

3.9 Reproduzierbarkeit und Auditierbarkeit

Die vier IST-Exporte A-D bilden den primären Forschungsanhang. Sie dokumentieren Pfade, Dateigrößen, Zeitstempel, Codezeilen, Logs und Evidenzklassen. Personenbezogene Daten und Secrets wurden bewusst ausgeschlossen. Die Reproduzierbarkeit ist dennoch begrenzt, weil kein eingefrorenes Git-Repository des untersuchten Stands existiert. Die Monographie kompensiert dies durch einen Erhebungsfreeze zum 22. August 2026 und durch die Trennung von aktuell belegtem Zustand und historischen Betreiberangaben.

Für eine zukünftige Replikation wäre ein versionierter Snapshot aus Quellcode, Konfigurationen ohne Secrets, anonymisierten Zustandsdateien, Hashwerten und read-only Analysedaten erforderlich. Diese Anforderung wird in den Designprinzipien als Versioned Change Control wieder aufgegriffen.

3.10 Gütekriterien der Fallstudie

Die interne Nachvollziehbarkeit wird durch Quellenprovenienz, Evidenzklassen und die explizite Behandlung widersprüchlicher Betreiberlogs gestützt. Konstruktvalidität wird durch die Trennung von Autonomie, Wirksamkeit und Governance verbessert. Externe Validität ist bewusst eingeschränkt: Der Fall ist ein einzelnes, historisch gewachsenes System auf einer spezifischen Hardware- und Organisationskonfiguration. Ziel ist analytische Generalisierung in Form von Designprinzipien, nicht statistische Repräsentativität.

Die methodische Strenge wird zusätzlich dadurch erhöht, dass fehlende Messungen als Ergebnis dokumentiert werden. Nicht erhobene Latenz, fehlende Postmaster-Daten, nicht vorhandene Referenztranskripte oder fehlende Modellbenchmarks werden nicht durch Schätzungen ersetzt. Gerade diese Messlücken sind Teil der Evaluation der Observability-Reife.

Kapitel IV – Entwicklung und Architektur des Löwen Codex OS

4.1 Vom explorativen Konzept zum implementierten Artefakt

Der Entwurf von 2025 beschrieb ein sehr breites Forschungs- und Anwendungssystem, enthielt jedoch offene Forschungsfragen und zahlreiche spekulative Elemente. Bis August 2026 verschob sich der Schwerpunkt auf ein konkret implementiertes Operator-System. Die aktuelle Architektur ist historisch gewachsen und folgt keiner klassischen Drei-Schichten-Struktur. Der Export beschreibt sechs nebeneinanderliegende Prozessketten auf einer gemeinsamen Maschine, gekoppelt über eingebettete relationale Datenhaltung, dateibasierte Zustandsdaten-Dateien im Webroot und lokale HTTP-Verbindungen. Der Windows-Taskplaner fungiert als zentraler Zeitgeber.

4.2 Gesamtarchitektur

Schicht	Funktion	Beispiele
S6 Betreiberoberfläche	War Room und Cockpits	große HTML-Oberfläche, weitere Cockpits, Telegram-Kanal
S5 Anwendungs-API	Backend und Nebenserver	FastAPI :8000, Outreach-, Content-, Operator- und Redirect-Server
S4 Assistenz-/KI-Layer	NLU, RAG, Operatorrollen	LEO, lokale Ollama-Modelle, Dev-/Skill-Funktionen
S3 Engines / Batch	operative Prozessketten	Outreach, Open-Data-Leadquelle, Content, Voice, Krypto, Blog
S2 Datenhaltung	persistente Zustände	leads.db, outreach.db, dateibasierte Zustandsdaten, Demo-HTML
S1 Infrastruktur	Routing und Zeitsteuerung	Reverse Proxy, externer Tunnel-/Zugriffsdienst Tunnel, Windows Task Scheduler, Ollama, ComfyUI

Die Architektur verwendet keinen Message Bus, keine Service Registry und keinen Container-Orchestrator. Diese Einfachheit reduziert Infrastrukturkomplexität, erhöht

jedoch die Kopplung an Dateipfade, Ports und lokale Zustände. Die Backend-Anwendung umfasst nach dem Export eine umfangreiche API-Oberfläche mit zahlreichen Routen in einer einzelnen main.py von rund 493 KB. Git-Repositories wurden im untersuchten Bestand nicht gefunden; Versionierung erfolgt über zeitgestempelte Dateikopien und ein Prosa-Deploy-Logbuch.

4.3 Lokale KI-Architektur

Zentrale Modellrollen werden über lokale LLM-Laufzeit lokal bereitgestellt. Für NLU, RAG und mehrere Backend-Rollen ist einem lokalen Sprachmodell konfiguriert; einem lokalen Sprachmodell wird unter anderem für Reply-Triage und Fallbacks verwendet, qwen2.5vl:7b für Vision-Aufgaben und einem lokalen Embedding-Modell für Embeddings. Ein einem lokalen Code-Modell ist für Entwicklungsfunktionen referenziert, deren produktive Nutzung jedoch nicht belegt ist. Content-Logs vom Erhebungstag belegen die produktive Nutzung mehrerer lokaler Modelle.

Der wichtigste Architekturbefund ist, dass das zentrale System nicht als allgemeine agentische Tool-Calling-Plattform arbeitet. Modellaufrufe liefern Text oder dateibasierte Zustandsdaten; Aktionsausführung, Reihenfolge, Retry-Mechanismen und Schreiboperationen liegen überwiegend in deterministischem Code. Damit ist die Systemautonomie stärker eine Eigenschaft der umgebenden Orchestrierung als der Sprachmodelle selbst.

4.4 Daten- und Wissensarchitektur

Die Persistenz verteilt sich auf zwei große eingebettete relationale Datenhaltung-Datenbanken, zahlreiche dateibasierte Zustandsdaten-Zustände, HTML-Artefakte und Logdateien. Das Wissenssystem indexiert Dokumente und Betreiberlogs über Embeddings. Zum Erhebungszeitpunkt war der Index jedoch veraltet: einem umfangreichen indexierten Wissensbestand reichten nur bis LOG-0276, während das Betreiberlog bereits LOG-0322 enthielt. Eine Neuindizierung hätte 1.189 Chunks ergeben; 260 Chunks bzw. rund 22 % des aktuellen Wissens fehlten im Index. Der Hygiene-Mechanismus war damit konzeptionell vorhanden, konnte aber aktuelles Wissen aktiv unterdrücken.

4.5 Infrastruktur, Deployment und Recovery

Reverse Proxy dient als Reverse Proxy und statischer Host für zahlreiche Subdomains. externer Tunnel-/Zugriffsdienst Tunnel stellt die externe Erreichbarkeit her; die konkrete Laufzeit war aus der Sandbox nicht prüfbar. Der Windows-Taskplaner steuert Nachtläufe, Backups, Content-Posting und Outreach. Backups sind ein vergleichsweise starker Bereich: tägliche Datenbank- und Gesamtbackups sind dokumentiert. Demgegenüber fehlen Git, Branches, CI und eine belastbare Commit-Historie. Rollbacks erfolgen über Dateikopien.

4.6 Architekturbezogene Risiken

Die Architektur zeigt mehrere Single-Host- und Kopplungsrisiken. Dienste teilen GPU-Ressourcen und lokale Ports; Zustände werden teilweise über Dateien statt transaktionale Schnittstellen synchronisiert. Das Backend besitzt keine eigene umfassende Authentifizierung und delegiert Zugriffsschutz an die Netzwerkschicht. Mehrere Schreibendpunkte liegen außerhalb der beschriebenen Berechtigungslogik. Diese Befunde werden in Kapitel VI als Governance-Risiken bewertet.

4.7 Integrationsarchitektur der öffentlichen Fassung

Das System koppelt Datenhaltung, Zustandsdateien und lokale Services über eine pragmatische Single-Host-Architektur. Für die öffentliche Fassung werden konkrete Dateinamen, interne Prioritätslogiken und reproduktionsnahe Integrationsdetails nicht offengelegt. Wissenschaftlich relevant bleibt der Befund, dass implizite Datenquellen, heterogene Zustandsablagen und fehlende zentrale Provenienz zu systemweiten Fehlerbildern führen können.

Die zentrale Designableitung lautet daher: Datenquellen müssen eindeutig identifizierbar, auf Integrität prüfbar und für Analysezwecke von produktiven Schreibpfaden getrennt sein.

4.8 Hardware- und Ressourcenarchitektur

Die lokale KI-Infrastruktur nutzt dieselbe physische GPU für Sprachmodelle, Bildgenerierung und Videopipelines. Damit wird Ressourcenmanagement Teil der Softwarearchitektur. Die dokumentierte VRAM-Konkurrenz führte dazu, dass Modelle

sich gegenseitig aus dem Speicher verdrängten. Das zunächst verschränkte Abarbeiten von Text, Bild und Video pro Slot erwies sich als instabil.

Der erfolgreiche Umbau auf ein Zwei-Phasen-Modell – zuerst Text für alle Marken, danach explizites Unload und anschließend Rendering – zeigt, dass lokale Multi-Modell-Systeme nach Ressourcenklassen orchestriert werden sollten. Diese Regel ist über den konkreten Stack hinaus plausibel: Auf nicht elastischer Hardware müssen Modellwechsel, Speicherbelegung und Warmup-Kosten explizit geplant werden.

4.9 War Room und Operator-Interface

Der War Room ist nicht nur Visualisierung, sondern Teil der Human-in-the-Loop-Architektur. Er entscheidet darüber, welche Zustände der Operator wahrnimmt und welche Interventionen praktisch möglich sind. Die zentrale HTML-Datei ist mit rund 2,47 MB und zahlreichen eingebetteten Skriptblöcken zugleich die größte Einzelkomponente des Systems. Ihr Vorteil ist schnelle Iteration ohne Buildschritt; ihr Nachteil ist hohe Kopplung und schwierige testbare Modularisierung.

Der 0-Byte-Datenbankfall zeigt, warum Oberflächen Datenprovenienz sichtbar machen sollten. Ein angezeigter Wert 0 kann echten Nullzustand, fehlende Daten, falsche Quelle oder Fehler bedeuten. Für kritische KPIs sollte das Interface daher neben Wert und Trend auch Quelle, Aktualitätszeitpunkt und Datenqualitätsstatus darstellen.

4.10 Lokale KI-Rollen und Modellzuordnung

Die produktiven Modellrollen sind spezialisiert: einem lokalen Sprachmodell für NLU, RAG und Content-Texte, einem lokalen Sprachmodell für Reply-Triage und Fallbacks, qwen2.5vl:7b für Vision und einem lokalen Embedding-Modell für Embeddings. einem lokalen Code-Modell ist für Entwicklungsfunktionen vorgesehen, deren produktive Nutzung jedoch nicht belegt ist. Diese Spezialisierung entspricht keinem dynamischen Model Router, sondern einer statischen Zuordnung je Funktion.

Der Verzicht auf universelles Tool Calling reduziert bestimmte Risikoklassen. Das Modell kann im zentralen Backend keine beliebige Funktion aufrufen; es liefert Daten, die anschließend von Code verarbeitet werden. Diese implizite Sperre ist governance-relevant, weil sie den möglichen Aktionsraum des Modells fundamental begrenzt.

4.11 Entwicklungshistorie und technologische Pfadabhängigkeit

Der Dateibestand und die Logs erlauben eine grobe Rekonstruktion der Entwicklung. 2025 dominierte die konzeptionelle Phase. Im Frühjahr 2026 verdichteten sich Backend und War Room. Zwischen 20. und 25. Juni entstand der Cold-Mail-Kern innerhalb weniger Tage. Die Content-Engine folgte bis zum ersten produktiven Post am 2. Juli. In der zweiten Julihälfte erreichte das System seine stärkste kombinierte Produktionsphase: Content lag zeitweise bei rund 15,3 Posts pro Tag, während der Erstmail-Versand bis auf etwa mehrere hundert pro Tag stieg. Im August verschob sich der Schwerpunkt auf Härtung, Fehlersuche und neue Governance-Mechanismen.

Diese Geschwindigkeit erzeugte Pfadabhängigkeit. Alte Serverstände, parallele Dateien und rückwärtskompatible Schutzprüfungen blieben bestehen. Der technische Zustand ist deshalb nicht nur Ergebnis bewusster Architekturentscheidungen, sondern auch Ergebnis iterativer Reparaturen. Für die DSR-Auswertung ist genau diese Historie wertvoll, weil Designprinzipien aus konkreten Fehlentwicklungen abgeleitet werden können.

4.12 Architektur-Gesamtbewertung

Das Artefakt besitzt hohe Funktionsbreite bei vergleichsweise niedriger Infrastrukturkomplexität. Ein einzelner Host trägt Reverse Proxy, lokale KI, Datenbanken, Content, Outreach, Voice und Control Center. Dieser Minimalismus ist eine Stärke für Geschwindigkeit und Kosten, zugleich ein Single Point of Failure. Fehlende Versionskontrolle, fehlendes Staging und heterogene Persistenz erhöhen das Betriebsrisiko.

Die zentrale architektonische Erkenntnis lautet daher nicht, dass Microservices oder Container zwingend erforderlich wären. Relevanter ist die Notwendigkeit eindeutiger Eigentumsgrenzen für Daten, reproduzierbarer Versionierung, fachlicher Healthchecks und zentraler Policy-Gates. Diese Prinzipien können sowohl in monolithischen als auch verteilten Architekturen umgesetzt werden.

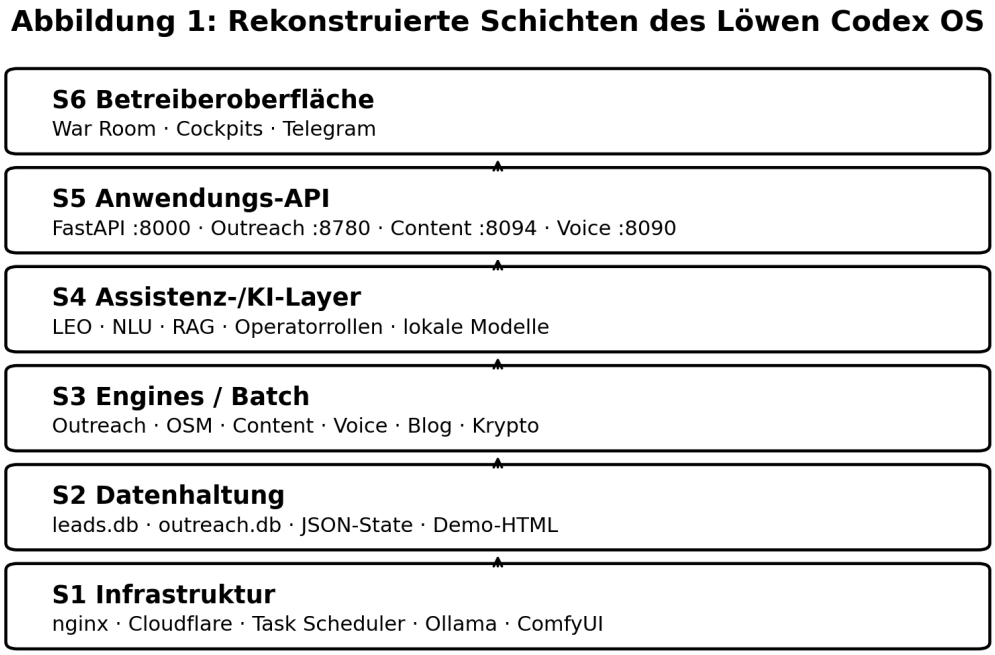


Abbildung 1: Rekonstruierte Schichten des Löwen Codex OS (eigene Darstellung nach IST-Export).

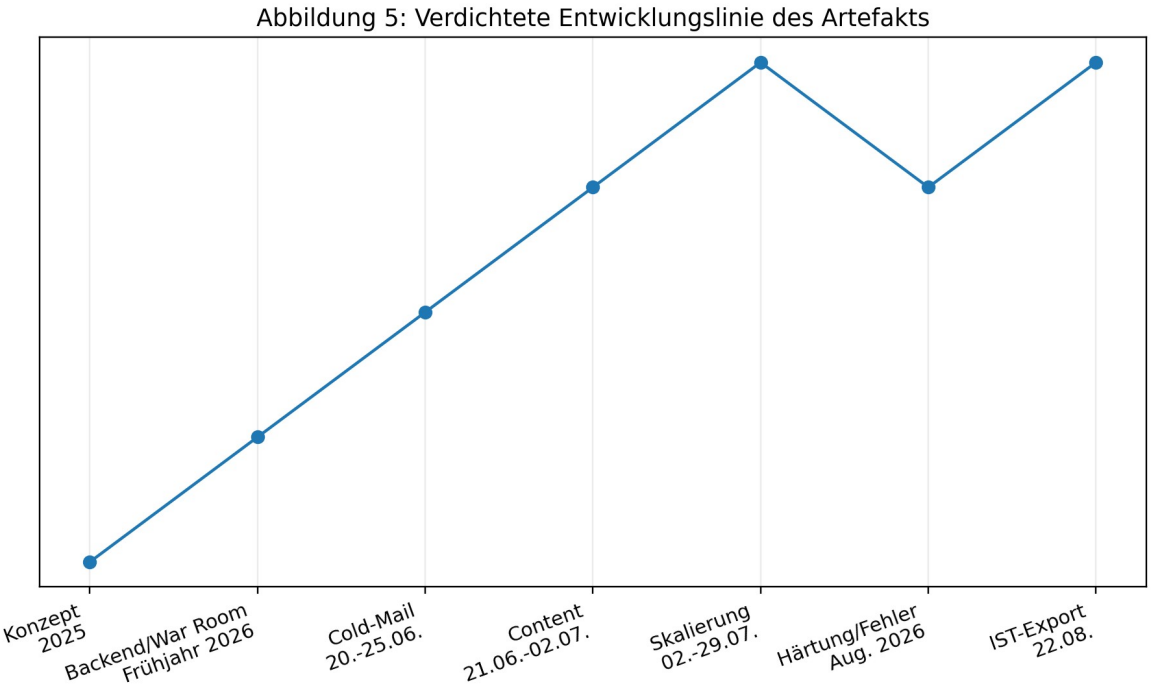


Abbildung 5: Verdichtete Entwicklungslinie des Artefakts (eigene Darstellung).

Architekturaspekt	Belegter Zustand	Implikation
Host-Modell	ein Single-Host-System	geringe Infrastrukturkomplexität,

		keine Redundanz
Persistenz	2 eingebettete relationale Datenhaltung-DBs + ~60 dateibasierte Zustandsdaten-Zustände	hohe Transparenz, heterogene Konsistenzmodelle
HTTP-Routing	Reverse Proxy + mehrere lokale Ports	einfache Entkopplung, Port-/Service-Health relevant
API	FastAPI, 225 Routen in main.py	breite Funktionalität, starke Monolithkopplung
Versionierung	kein Git im geprüften Bestand	erschwerte Reproduzierbarkeit und Rollbacks
KI	lokales Ollama + ComfyUI	Datenlokalität, GPU als Ressourcenengpass

Kapitel V – Implementierte Geschäftsprozesse

5.1 Mail- und Outbound-Engine

Die Mail-Engine ist die am stärksten skalierte operative Prozesskette. Im Export sind mehrere getrennte Versanddomänen und 195 Postfächer dokumentiert. Versand erfolgt über IONOS SMTP mit Warmup-, Domain-Ramp-, Verify-, Bounce- und Zeitfensterlogik. Kampagnen werden über einen Autopilot verarbeitet; Follow-ups, Klick-Follow-ups, Re-Warm-Mechanismen, Reminder und Eskalationen sind weitgehend deterministisch automatisiert. mehr als 50.000 personalisierte Demo-HTML-Dateien belegen die starke Artefaktproduktion des Systems.

Die KI spielt im eigentlichen ausgehenden Versandkanal eine kleinere Rolle als die Systembezeichnung vermuten lässt. Der Versand selbst ist regelbasiert; KI wird vor allem für Antwortklassifikation und Antwortentwürfe eingesetzt. Individuelle Antworten werden nicht automatisch versendet. Diese Trennung ist für die spätere Autonomiebewertung wesentlich.

5.2 Voice AI

Die Voice-Kette verbindet Demo-Klicks mit einem Autodialer, externer Telefonieprovider, lokaler Spracherkennung, lokalem LLM und lokaler Sprachsynthese. Der dokumentierte Pfad nutzt Whisper large-v3 auf CUDA, einem lokalen Sprachmodell über lokale LLM-Laufzeit und Piper TTS. Kritische Gesprächsentscheidungen wie Terminbuchung und Entscheiderprüfung werden teilweise serverseitig durch Zustandsmaschinen und Regex-Gates abgesichert. Die technische Tiefe der Voice-Implementierung steht jedoch in starkem Kontrast zu den operativen Ergebnissen, die in Kapitel VI analysiert werden.

5.3 Content Engine

Die Content-Engine erzeugt Captions, Hooks, Bildprompts, Bilder und Reels und veröffentlicht Inhalte für mehrere Marken. Sechs Marken waren zum Erhebungszeitpunkt aktiv; das konfigurierte Tagesziel lag bei 18 Posts. Die Engine nutzt

lokale Sprach- und Visionmodelle sowie lokale Medienpipelines. Der Review-Modus ist technisch vorhanden, aber die aktiven Marken stehen auf Auto-Modus. Dadurch werden Inhalte ohne menschliche Einzelprüfung veröffentlicht. Für regulierungsnahe Marken ist dies ein zentraler Governance-Befund.

5.4 Sales und CRM

Das Backend bündelt Lead-Intake, Priorisierung, CRM-Status, Diagnose, Angebots- und Rechnungslogik sowie verschiedene Operatoransichten. Mehrere Funktionen erzeugen lediglich Vorschläge oder Entwürfe. Die produktive Nutzung ist heterogen: CRM- und Lead-Funktionen sind sichtbar, während Angebots- und Rechnungsverzeichnisse zum Erhebungszeitpunkt leer waren. Dadurch ist zwischen implementierter Fähigkeit und tatsächlicher Nutzung zu unterscheiden.

5.5 Knowledge, LEO und Operatorfunktionen

LEO verbindet NLU, RAG, Operatorrollen und verschiedene Assistenzfunktionen. Besonders interessant ist die Konflikt- und Ratifizierungslogik des Wissenssystems: widersprüchliche Fakten werden markiert, ratifizierte Fakten sollen im Ranking bevorzugt werden, und die ursprüngliche Wissensquelle wird nicht automatisch überschrieben. Dieser Entwurf ist stark Human-in-the-Loop-orientiert. Zum Stichtag existierte jedoch keine Datei ratifizierter Fakten, und der Index war veraltet.

5.6 Backups und Monitoring

Backups laufen automatisiert und sind über aktuelle Erfolgsartefakte belegt. Das Monitoring ist dagegen überwiegend beobachtend. Der Watchdog meldet Alarme, greift aber gemäß Doktrin nicht automatisch ein. Ein globaler Kill Switch existiert nicht. Diese Kombination zeigt eine bewusste Zurückhaltung bei automatischen Reparaturen, zugleich aber eine operative Schwäche bei schnellen Notabschaltungen.

5.7 Outbound als deterministisches Automatisierungssystem

Die Mail-Engine illustriert, dass hohe Autonomie nicht zwingend agentische LLM-Entscheidung voraussetzt. Kampagnenstatus, Warmup, Tagesbudgets, Bounce-Guard, Follow-up-Priorisierung, Feiertags- und Wochenendlogik, Drip-Jitter, Domain-Rotation

und Suppression sind deterministisch implementiert. Das LLM kommt primär bei semantischer Antwortklassifikation und Formulierung von Antwortentwürfen zum Einsatz.

Eine besonders robuste Konstruktion ist der Demo-Link-Platzhalter in der Reply-Triage. Das Modell darf keine URL erfinden, sondern nur einen Token ausgeben, der anschließend deterministisch aus der Datenbank ersetzt wird. Damit bleibt semantische Flexibilität beim Modell, während die referenzielle Wahrheit im Code liegt. Dieses Muster lässt sich als constrained generation with deterministic resolution beschreiben.

Die Versandautomation ist zusätzlich durch Caps und Zustellbarkeitsregeln begrenzt. Gleichzeitig ist sie nach einmaliger Aktivierung weitgehend Human-out-of-the-Loop. Diese Kombination macht sie zu einem geeigneten Beispiel für einen gut begrenzten L4-Prozess, dessen zentrale Schwäche nicht Ausführungsstabilität, sondern gemessene Geschäftswirkung ist.

5.8 Lead-Beschaffung und Open-Data-Leadquelle-Autoloader

Der Open-Data-Leadquelle-Autoloader ergänzt die Outbound-Kette durch kontinuierliche Lead-Beschaffung. Logs belegen produktive Läufe bis zum Erhebungsstichtag. Die Ausbeute variiert stark nach Segment; einzelne US-Segmente erreichten hohe E-Mail-Anteile, während andere Segmente null neue Leads lieferten. Diese Streuung zeigt, dass ein autonomer Scraper nicht als homogene Datenquelle behandelt werden sollte. Segmentqualität ist eine eigene Messgröße.

Für die Prozessarchitektur ist relevant, dass Lead-Beschaffung, Import, Deduplikation, Offer-Ableitung und spätere Kommunikation getrennte Zustände besitzen. Damit kann der Datenstrom an mehreren Stellen qualitativ degradieren. Die Forschungsexporte messen Mengen, aber keine systematische Lead-Qualität oder spätere Conversion nach Quelle. Diese Attribution bleibt eine offene Messlücke.

5.9 Content-Autopilot zwischen Skalierung und Freigabe

Die Content-Engine ist eine vollständige lokale Produktionskette aus Caption, Hook, Bildprompt, Bild- oder Reel-Rendering, Audio-Mux und Meta-Publishing. Im Zeitraum 2. Juli bis 22. August dokumentiert daypost.log mehrere hundert Posting-Versuche, davon

456 erfolgreich. Die resultierende Publishing-Erfolgsquote von über 90 % zeigt, dass die Kette grundsätzlich produktiv ist, obwohl einzelne Fehlerklassen – Zeichensatz, fehlende Dateien, abgelaufene Tokens und Timeouts – sichtbar bleiben.

Zwischen 4. und 11. August wurden im Queue-Zeitraum im Mittel 12,0 Posts pro Tag veröffentlicht; zwischen 12. und 22. August sank der Wert auf 3,3. Nur an zwei Tagen wurde das Soll von 18 erreicht. Dieser Abfall ist ein wichtiger Reliability-Befund, weil die Pipeline formal produktiv bleibt, ihre Durchsatzleistung aber deutlich degradiert.

Governance-seitig existiert ein Review-Modus, ist für die sechs aktiven Marken aber deaktiviert. Damit operiert das Publishing als L5-Prozess im verwendeten Schema. Die formalen Validatoren können Syntax- und Hook-Probleme begrenzen, nicht jedoch regulatorisch problematische Aussagen. Ein risikobasierter Betrieb sollte deshalb zwischen Markenklassen unterscheiden.

5.10 Voice als End-to-End-Prozess

Die Voice-Kette ist das technisch heterogenste Subsystem. Ein Demo-Klick führt über Autodialer und Zeitfensterlogik zum Voice-Agent, der externer Telefonieprovider mit lokalen STT-, LLM- und TTS-Komponenten verbindet. Server-seitige Gesprächsgates sichern Terminbuchung, Entscheiderprüfung und IVR-Erkennung. Die Architektur ist damit anspruchsvoller als die reine Implementierungszahl vermuten lässt.

Die operative Wirkung bleibt jedoch minimal. über 1.500 protokollierte Outcomes umfassen nahezu allen nicht erreichte Versuche und genau einen falschen Ansprechpartner; Termine und Rückrufwünsche blieben null. dem weit überwiegenden Teil Fälle endeten bereits als externer Telefonieprovider failed. Seit dem 12. August wurden keine erfolgreichen Call-Outcomes mehr fortgeschrieben, obwohl der Autodialer weiterhin Zustände veränderte. Das Subsystem bildet deshalb den stärksten Fall für die Trennung von Aktivität und Wirkung.

Die Kette enthält außerdem Governance-Lücken: Recording ist aktiviert, ohne dass im analysierten eigenen Begrüßungspfad ein Aufnahmehinweis gefunden wurde; eine explizite DNC-/Suppression-Logik fehlt. Diese Befunde werden technisch beschrieben, nicht abschließend juristisch bewertet.

5.11 LEO, RAG und Operatorrollen

LEO kombiniert Browser-Router, NLU, RAG, Operatorrollen und mehrere Batchfunktionen. Produktiv belegt sind insbesondere Feeds, Akquise-Ideen, Aktionsaudit und Memory-Batches. Andere Komponenten wie Dev-Agent, /learn oder Modell-Router sind implementiert, hinterließen aber keine produktiven Zustandsartefakte. Die Fallstudie unterscheidet daher konsequent zwischen implementierter Fähigkeit und realer Nutzung.

Die Operatorrollen sind datenseitig begrenzt, indem unterschiedliche Quellen gesammelt werden. Es existiert jedoch keine Agent-to-Agent-Kommunikation. Der sogenannte Conductor koordiniert nicht technisch zwischen Modellen, sondern liefert dem Menschen eine Empfehlung, welche Rolle als Nächstes relevant ist. Das System ist damit eher ein Multi-Role-Assistenzsystem als ein autonomes Multi-Agent-System.

5.12 Backups, Monitoring und Recovery

Der Backup-Pfad gehört zu den reifsten Automationen. Tägliche Datenbank- und Gesamtbackups sind mit aktuellen Erfolgsartefakten belegt; ein Disaster-Drill wurde einmal dokumentiert. Monitoring ist demgegenüber bewusst passiv. Der Watchdog meldet, greift aber nicht automatisch ein. Diese Doktrin reduziert das Risiko automatischer Fehlreparaturen, setzt jedoch voraus, dass Alarme sichtbar sind und ein Operator reagieren kann.

Ein globaler Kill Switch fehlt. Dienste können teilweise über APIs gestartet, aber nicht zentral gestoppt werden. Für ein System mit autonomen Außenkanälen ist dies eine strukturelle Lücke. Die spätere Designempfehlung eines zentralen Stop-Mechanismus folgt unmittelbar aus diesem Befund.

Abbildung 4: Vereinfachte End-to-End-Kette der Voice-Automation

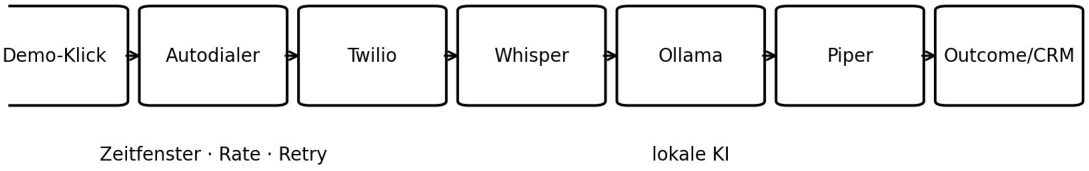


Abbildung 4: Vereinfachte End-to-End-Kette der Voice-Automation (eigene Darstellung).

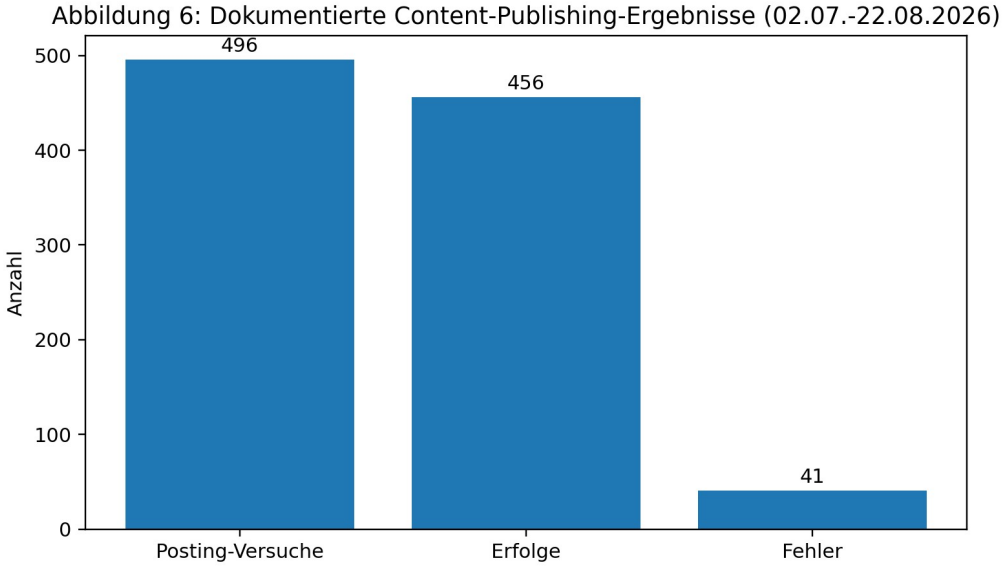


Abbildung 6: Dokumentierte Content-Publishing-Ergebnisse 02.07.-22.08.2026 (Quelle: daypost.log, aggregiert im IST-Export).

Kapitel VI – Evaluation und Ergebnisse

6.1 Automatisierungsgrad

Für die Fallstudie wird ein L0-L5-Schema verwendet. L0 bezeichnet manuelle Prozesse, L1 KI-gestützte Information, L2 KI-erzeugte Artefakte mit menschlicher Ausführung, L3 Ausführung mit menschlicher Bestätigung kritischer Schritte, L4 weitgehend autonome Ausführung innerhalb kodierter Grenzen und L5 vollständig autonome Ausführung ohne wirksame inhaltliche Grenze. Entscheidend ist die menschliche Beteiligung je Vorgang.

Prozess	Level	Kernbegründung
Cold-Mail Erstversand	L4	täglicher Versand ohne Einzelbestätigung; Caps und Verify-Gates
Follow-up-Kaskade	L4	zeit- und ereignisgesteuert
Domain-Rotation	L4	Auto-Pfad ohne Prompt
Antwort-Triage	L2	KI erzeugt Entwürfe, sendet nicht
Individueller Antwortversand	L2/L3	menschliche Browserbestätigung; serverseitig schwach
Content-Generierung	L4	automatischer Nachtlauf
Content-Publishing	L5	aktive Marken im Auto-Modus, kein Human Gate
Autodialer	L4	automatische Auswahl und Wahl innerhalb harter Grenzen
Dev-Agent Promote	L3 nach Bauart	Sandbox/Test/Promote, aber nicht produktiv genutzt
Backup	L5	vollautomatisch
Watchdog	L1	meldet nur, greift nicht ein

6.2 Mail-Funnel und Geschäftswirkung

Der wichtigste quantitative Negativbefund betrifft den Outbound-Funnel. Bis zum 21. August 2026 dokumentiert der Revenue-Report über 19.000 Sendungen, über 1.000 Klicks und mehreren Dutzend Antworten, aber keinen kampagnenzuordenbaren Termin. Die aggregierte Klickrate beträgt 5,4 %, die Antwortrate rund 0,2 % und die kampagnenbezogene Terminquote 0 %. Eine weitere Buchung existierte, war aber

keiner Kampagne zuordenbar. Damit ist Aktivität klar belegt, Geschäftswirkung jedoch nicht.

Stufe	Wert	Interpretation
Sendungen	über 19.000	hohe operative Aktivität
Klicks	über 1.000 (5,4 %)	Interesse/Interaktion vorhanden
Antworten	48 (~0,25 %)	starker Funnel-Abfall
Voice-Wahlversuche	über 1.500	automatisierte Anschlussaktivität
erreichte Gesprächspartner	1	Voice-Brücke faktisch unwirksam
kampagnenzuordenbare Termine	0	keine nachgewiesene Conversion

Der Export warnt ausdrücklich vor einer vorschnellen Ursachenzuschreibung. Der systemeigene Reputationsreport markiert mehrere Mandantentöpfe mit DRIFT und weist darauf hin, dass stille Spam-Zustellung keinen Bounce erzeugt. Gleichzeitig fehlen Google- und Microsoft-Postmaster-Daten; tatsächliches Inbox-Placement bleibt unbeobachtet. Deshalb kann die niedrige Antwortrate nicht eindeutig auf Copy, Zustellung oder Zielgruppenauswahl zurückgeführt werden.

6.3 Voice-Evaluation

Zwischen 14. Juli und 12. August 2026 wurden über 1.500 Voice-Outcomes protokolliert. nahezu allen davon waren „nicht_erreicht“, ein Fall wurde als falscher Ansprechpartner klassifiziert. Termine und Rückrufwünsche: jeweils null. dem weit überwiegenden Teil Versuche endeten mit „externer Telefonieprovider: failed“. Seit dem 12. August wurden keine erfolgreichen externer Telefonieprovider-Anrufe mehr nachgewiesen, während der Autodialer weiterhin Zustandsdateien aktualisierte und Versuchsstände erhöhte. Dies ist ein besonders klares Beispiel dafür, dass Prozessaktivität nicht mit erfolgreicher Aktionsausführung gleichgesetzt werden darf.

6.4 Governance-Evaluation

Die Governance-Analyse zeigt eine asymmetrische Architektur. Wo Gates existieren, sind sie häufig sauber implementiert: confirm-Parameter verhindern Schreibvorgänge, Sandbox- und Testmechanismen schützen Entwicklungsartefakte, Kontext-Whitelists reduzieren Datenabfluss und Secret-Scrubbing filtert sensible Fragmente. Gleichzeitig

sind diese Mechanismen opt-in und nicht systemweit als Default-Deny verankert. Ein großer Anteil schreibender Endpunkte besitzt kein entsprechendes Gate.

Besonders auffällig ist die Diskrepanz zwischen dokumentierter Doktrin und realem Betrieb. Die Doktrin betont menschliche Freigabe, während release_guard und Domain-Flip bestimmte Aktivierungen automatisch ausführen. Content-Publishing läuft für aktive Marken ohne Einzelprüfung. Der einzige produktive Punkt mit expliziter menschlicher Bestätigung einer individuellen ausgehenden Antwort liegt im Browser; der Serverendpunkt selbst erzwingt diese Bestätigung nicht. Umgekehrt sind aufwendige Governance-Ketten für Dev-Agent, Sitebuild, Learn, Angebote und Rechnungen gebaut, aber zum Stichtag praktisch ungenutzt.

6.5 Security-, Datenschutz- und Compliance-Befunde

Die Anwendung besitzt keine umfassende eigene Authentifizierung und verlässt sich auf die vorgelagerte Netzwerkschicht. Eine Berechtigungs-Middleware ist an mehreren Stellen fail-open. Mehrere hochprivilegierte Endpunkte liegen außerhalb ihrer Pfadzuordnung. Für Voice-Aufnahmen ist Record=true dokumentiert, während im analysierten Prompt- und Begrüßungspfad kein eigener Aufnahmehinweis gefunden wurde. Eine explizite DNC-/Suppression-Prüfung im Voice-Pfad fehlt. Diese Befunde werden nicht als abschließende Rechtsbewertung interpretiert, zeigen jedoch klare Governance- und Compliance-Lücken.

6.6 Knowledge- und Memory-Evaluation

Das Wissenssystem zeigt gleichzeitig ein gutes Governance-Konzept und einen Betriebsfehler. Konflikte sollen sichtbar gemacht und ratifizierte Fakten durch bewusste menschliche Freigabe priorisiert werden. Der Index war jedoch acht Tage alt und enthielt 46 neuere Logeinträge nicht. Weil indexierte Treffer frische Keyword-Ergebnisse verdrängen können, kann ein Hygiene-Mechanismus dadurch veraltetes Wissen stabilisieren. Die Lehre lautet: Governance-Mechanismen benötigen nicht nur korrekte Logik, sondern auch überwachte Aktualität.

6.7 Negative Resultate als Designwissen

Negativergebnis	Befund	abgeleitete Lehre

Fehlende Versionskontrolle	Kein Git-Repository im untersuchten Bestand; Rückweg über Backups und Dateikopien.	Versionskontrolle ist Governance-Infrastruktur, nicht Komfortfunktion.
Port-Healthcheck	Portbelegung kann als Dienstverfügbarkeit interpretiert werden.	Healthchecks müssen Serviceidentität und Funktionsfähigkeit prüfen.
Voice-Ausfall	Autodialer verarbeitet Leads weiter, obwohl reale Calls nicht mehr nachweisbar sind.	Erfolg muss end-to-end bestätigt werden, bevor Retry-Zähler verbraucht werden.
Wissensindex veraltet	22 % des aktuellen Wissens fehlten im Index.	RAG braucht Freshness-Metriken und automatische Reindex-Gates.
Governance ungleich verteilt	starke Gates an ungenutzten, schwache an produktiven Pfaden.	Governance nach Risiko und Außenwirkung priorisieren.
Outbound ohne Conversion	hohe Aktivität, null kampagnenzuordenbare Termine.	Business-KPI muss über Aktivitäts-KPI stehen.

6.8 Beantwortung der Forschungsfragen auf Ergebnisebene

RQ1: Integration gelingt im Fallartefakt durch pragmatische Kopplung gemeinsamer Datenbanken, dateibasierte Zustandsdaten-Zustände, lokaler HTTP-Dienste, Reverse Proxy und Task Scheduler. Die Architektur ist funktionsfähig, aber stark host- und dateipfadabhängig. Entscheidend ist die Trennung von generativer KI und deterministischer Ausführung.

RQ2: Hohe Automatisierung ist bei repetitiven, klar begrenzten Prozessen erreichbar. Grenzen entstehen bei externer Dienstabhängigkeit, Datenqualität, fehlender End-to-End-Telemetrie, regulatorischer Außenwirkung und Prozessen, deren Zielzustand semantisch statt technisch definiert ist.

RQ3: Technisch erzwungene Gates sind wirksamer als Prompt- oder Dokumentationsregeln. Die Fallstudie zeigt jedoch, dass Governance nur dann wirkt, wenn sie systematisch und an produktiven Risikopunkten angewandt wird. Opt-in-Gates erzeugen Lücken.

RQ4: Negative Resultate zeigen, dass Aktivität, Autonomie und Wirksamkeit getrennt gemessen werden müssen. Die wichtigsten Lernquellen sind nicht nur erfolgreiche Features, sondern Fehlzustände, veraltete Zustände, nicht genutzte Schutzmechanismen und Diskrepanzen zwischen Bericht und Realität.

6.9 Cross-Case-Vergleich der Prozessketten

Mail, Content und Voice zeigen drei unterschiedliche Profile. Mail besitzt hohe Autonomie und zahlreiche deterministische Schutzmechanismen, erreicht im gemessenen Funnel jedoch keine kampagnenzuordenbaren Termine. Content besitzt hohe Autonomie, reale Veröffentlichungserfolge und zugleich eine abgeschaltete Human-Review-Schicht. Voice besitzt mehrere Gesprächsgates und eine aufwendige technische Kette, erreicht aber praktisch keine fachliche Wirkung. Kein einzelner Begriff wie autonom, agentisch oder produktiv beschreibt diese Unterschiede ausreichend.

Die Vergleichsanalyse zeigt außerdem, dass Governance nicht linear mit Automatisierungsgrad zunimmt. Ein L4-Prozess kann durch harte Caps, Suppression und Idempotenz gut begrenzt sein. Ein L2-Prozess kann schwächer abgesichert sein, wenn die menschliche Bestätigung nur im Browser liegt und serverseitig umgangen werden kann. Bewertet werden muss der vollständige Aktionspfad inklusive alternativer Aufrufwege.

6.10 Forschungsqualität der vorhandenen A/B-Mechanismen

Der ursprüngliche Entwurf sah A/B-Tests, p-Werte und Effektstärken vor. Die tatsächliche Varianteninfrastruktur der Mail-Engine verknüpft Variante und Ergebnis, weist Empfänger jedoch deterministisch alternierend zu. Eine vorab dokumentierte Hypothese, Randomisierung und echte Kontrollgruppe fehlen. Follow-ups sind zudem nicht variantenzuordenbar. Für eine kausale Interpretation sind diese Daten deshalb ungeeignet.

Die Content-Engine erzeugt pro Aufgabe genau eine Caption und einen Hook; es existiert kein Kandidatenvergleich. Die NLU-Lernschleife enthält nur drei dokumentierte Totalausfälle und erfasst keine selbstbewusst falschen Klassifikationen. Systemweit wurde keine Eval-Harness mit Genauigkeit, Precision, Recall oder Golden Testset gefunden. Diese Nicht-Existenz ist selbst ein wichtiges Ergebnis: Das Artefakt ist stark in produktiver Automation, aber schwach in formalisierter Modellqualitätsevaluation.

6.11 Security- und Datenschutzbefunde der öffentlichen Fassung

Die Untersuchung identifizierte mehrere für historisch gewachsene Single-Operator-Systeme typische Governance- und Datenschutzrisiken. Dazu zählen unzureichende Trennung von Entwicklungs-, Backup- und Auslieferungsartefakten sowie eine starke Abhängigkeit von vorgelagerten Schutzschichten. Konkrete Pfade, Dateimuster, Mengen und potenziell ausnutzbare Implementierungsdetails werden in der öffentlichen Fassung nicht dokumentiert.

Der wissenschaftliche Kernbefund bleibt erhalten: Sicherheits- und Datenschutzkontrollen müssen entlang des tatsächlich extern erreichbaren Aktionspfads bewertet werden; lokale Einzelmaßnahmen ersetzen keine systemweite Policy.

6.12 Reliability-Fallstudie I: 0-Byte-Datenbankattrappe

Die leere operative Outreach-Datenbank im einem internen Unterverzeichnis ist ein anschaulicher Fall impliziter Datenquellen. Mehrere Module wählten die erste existierende Datei aus einer Kandidatenliste. Weil die Attrappe formal existierte, aber keine Tabellen enthielt, entstanden Nullanzeigen und Fehler. Die Reparatur führte zu Mindestgrößenprüfung, geänderter Priorität und read-only Schutz. Die Ursachedatei selbst blieb bestehen.

Die Designlehre lautet: Datenquellen benötigen Identität und Integritätsprüfung, nicht bloß Existenz. Ein Consumer sollte Schema-Version, erwartete Tabellen oder eine eindeutige Source-ID prüfen. Noch besser ist eine zentrale Datenquellenkonfiguration, sodass Schutzlogik nicht mehrfach an Konsumenten dupliziert werden muss.

6.13 Reliability-Fallstudie II: VRAM-Thrashing

Die Content-Degradation wurde zunächst nicht korrekt diagnostiziert. Erst die spätere Analyse identifizierte den wiederholten Modellwechsel zwischen lokale LLM-Laufzeit, lokales Bildmodell und lokales Videomodell als zentralen Engpass. Der Zwei-Phasen-Umbau reduzierte Modellwechsel und stabilisierte die Ressourcennutzung. Dieser Fall zeigt, wie Betreiberhypothesen iterativ falsifiziert werden können und warum finale,

am Code bestätigte Erklärungen höher zu gewichten sind als frühe Logbuchinterpretationen.

6.14 Reliability-Fallstudie III: Encoding und Plattformgrenzen

Unicode- und Emoji-Ausgaben führten in Teilen der Windows-Pipeline zu Abstürzen. Der Fehler musste in mehreren Pfaden gehärtet werden. Für AI-native Systeme ist dies kein Randthema: Generative Modelle produzieren vielfältige Unicode-Ausgaben. Die Textpipeline muss daher von Generierung über Logging bis Posting UTF-8-sicher sein. Klassische Plattformdetails können sonst den Nutzen einer funktionierenden KI vollständig blockieren.

6.15 Reifegradprofil des Artefakts

Auf Funktionsbreite und Automatisierung ist das System fortgeschritten. Lokale KI, mehrere produktive Engines, zahlreiche Domains, tausende Demos und automatisierte Scheduler sind belegt. Auf Software-Engineering-Reife ist das Profil deutlich schwächer: kein Git, keine zentrale Testsuite, heterogene Zustandsablage und Single-Host-Betrieb. Governance ist gemischt: einzelne Gates sind sorgfältig, die Abdeckung ist jedoch ungleich. Observability ist datenreich, aber zentrale End-to-End-Metriken fehlen.

Das System ist damit weder bloßer Prototyp noch gehärtete Produktionsplattform im Enterprise-Sinn. Es ist ein produktiv genutztes Forschungs- und Betriebsartefakt mit hoher funktionaler Ambition und ungleichmäßiger Härtung. Diese Zwischenposition ist wissenschaftlich besonders ergiebig, weil erfolgreiche und gescheiterte Designentscheidungen gleichzeitig sichtbar bleiben.

6.16 Synthese der Ergebnisse

Die zentrale Synthese lautet: Autonomie, Wirksamkeit und Governance sind orthogonale Eigenschaften. Ein Prozess kann hochautonom und unwirksam sein, wie die Voice-Aktivität zeigt. Ein Prozess kann hochautonom und funktional wirksam sein, aber Governance-Lücken besitzen, wie das Content-Publishing. Ein Prozess kann stark gegated, aber ungenutzt sein, wie Dev-Agent und Sitebuild. Erst das gemeinsame Profil ermöglicht eine angemessene Bewertung.

Damit verschiebt sich die Entwicklungsfrage von „Wie automatisieren wir mehr?“ zu „Welche Entscheidung darf unter welchen messbaren Bedingungen automatisch erfolgen, und woran erkennt das System selbst, ob sie erfolgreich war?“. Diese Frage verbindet Automatisierung, Reliability und Governance und bildet den Kernbeitrag der Fallstudie.

Prozess	Autonomie	Wirksamkeit	Governance	Kernbefund
Cold-Mail	hoch (L4)	niedrige Zielwirkung	mittel bis hoch	stabile Automation, aber Funnel bricht vor Termin ab
Reply-Triage	mittel (L2)	Entwurfsfunktion belegt	hoch	Modell formuliert, Mensch sendet
Content Publishing	sehr hoch (L5)	Publishing 91,7 % je Versuch	mittel/niedrig	Review vorhanden, aber deaktiviert
Voice	hoch (L4)	sehr niedrig	gemischt	starke Gesprächsgates, schwache End-to-End-Wirkung
Dev-Agent	L3 nach Bauart	nicht produktiv belegt	hoch nach Bauart	Governance dichter als Nutzung
Backup	sehr hoch (L5)	Erfolge belegt	hoch	Autonomie in engem, reversiblen Prozess sinnvoll

Kapitel VII – Diskussion, Designprinzipien und Fazit

7.1 Diskussion der zentralen Ergebnisse

Die Fallstudie widerspricht einer verbreiteten intuitiven Gleichsetzung von „mehr KI“ mit „mehr Autonomie“ und von „mehr Autonomie“ mit „mehr Leistung“. Im Löwen Codex OS werden viele zentrale Aktionen nicht vom Sprachmodell entschieden, sondern von deterministischem Code. Dennoch ist das Gesamtsystem in mehreren Prozessketten hochautomatisiert. Autonomie ist damit eine Systemeigenschaft der Orchestrierung und nicht notwendigerweise eine Eigenschaft des Modells.

Ebenso zeigt die Untersuchung, dass technische Reife einzelner Komponenten nicht mit Geschäftswirkung gleichgesetzt werden darf. Der Voice-Stack enthält lokale STT-, LLM- und TTS-Komponenten sowie serverseitige Gates, erreichte im beobachteten Zeitraum aber praktisch keine Gesprächspartner. Die Mail-Engine skaliert Domains, Postfächer, Sequenzen und Demo-Artefakte, erzeugte jedoch keine kampagnenzuordenbaren Termine. Diese Befunde sind kein Beweis gegen AI-native Automatisierung; sie zeigen vielmehr, dass End-to-End-Zielmetriken die zentrale Evaluationsgröße sein müssen.

7.2 Abgeleitete Designprinzipien

Designprinzip	Inhalt
DP1 Deterministic Boundaries	Kritische Aktionen sollen durch deterministischen Code begrenzt werden; Prompts allein sind keine Governance.
DP2 Process-Level Autonomy	Autonomie muss pro Prozess und Entscheidungspunkt klassifiziert werden, nicht als globales Systemlabel.
DP3 Default-Deny Governance	Schreibende oder extern wirksame Aktionen sollten standardmäßig blockiert und explizit freigegeben werden.
DP4 End-to-End Observability	Ein Prozess gilt erst als erfolgreich, wenn der fachliche Zielzustand bestätigt ist; Port, HTTP 200 oder Queue-Fortschritt reichen nicht.
DP5 Governance follows Exposure	Die stärksten Gates gehören an Prozesse mit Außenwirkung, Datenrisiko oder finanzieller/rechtlicher Relevanz.
DP6 Fresh Knowledge	RAG- und Memory-Systeme benötigen Freshness-Metriken, Reindex-Überwachung und sichtbare

	Wissensstände.
DP7 Global Stop Capability	Autonome Systeme benötigen einen technisch wirksamen, zentralen Kill Switch sowie stop-fähige Servicekontrollen.
DP8 Separate Activity from Outcome	Sends, Posts, Calls und generierte Artefakte sind Aktivitätsmetriken; Conversion, Erreichbarkeit und Zielerfüllung sind Ergebnisgrößen.
DP9 Versioned Change Control	Produktive AI-native Systeme benötigen diff-basierte Versionskontrolle, reproduzierbare Deployments und nachvollziehbare Rollbacks.
DP10 Human Oversight by Risk	Human-in-the-Loop sollte risikobasiert eingesetzt werden: nicht jede Routineaktion braucht Freigabe, aber kritische Außenwirkung muss wirksam kontrollierbar sein.

7.3 Beitrag zur Theorie

Die Arbeit konkretisiert bestehende Automatisierungs- und Governance-Perspektiven für einen AI-nativen Einzelfall. Sie schlägt vor, Autonomie, Wirksamkeit und Governance als orthogonale Evaluationsdimensionen zu behandeln. Diese Trennung erweitert klassische Automatisierungsbetrachtungen um die Frage, ob ein hochautomatisierter Prozess sein fachliches Ziel tatsächlich erreicht und ob seine Grenzen technisch durchsetzbar sind.

7.4 Beitrag zur Praxis

Für Entwickler und Betreiber kleiner AI-Systeme folgt daraus eine klare Priorisierung: Erstens müssen Zielmetriken end-to-end messbar sein. Zweitens sollten externe Aktionen und persistente Schreibvorgänge über zentrale Policy-Gates laufen. Drittens sind Versionskontrolle, Service-Healthchecks, Kill Switches und Wissens-Freshness keine nachgelagerten Betriebsdetails, sondern Kernbestandteile der AI-Governance. Viertens sollte lokale KI dort eingesetzt werden, wo Datenschutz, Kosten oder Latenz davon profitieren, ohne die operativen Grenzen eigener Infrastruktur zu unterschätzen.

7.5 Limitationen

Die Untersuchung analysiert ein einzelnes, vom Autor selbst entwickeltes System. Sie erlaubt daher keine statistische Generalisierung. Mehrere Datenquellen sind systemeigene Reports, deren Definitionen einzeln geprüft werden müssen. Live-

Datenbanken wurden nicht direkt geöffnet; laufende Prozesse konnten aus der Analyseumgebung nicht abgefragt werden. Es fehlen kontrollierte A/B-Experimente mit sauber dokumentierten Hypothesen, umfassende Latenztelemetrie, externe Zustellbarkeitsdaten und ein unabhängiger Security-Audit. Aussagen über Ursachen niedriger Conversion bleiben deshalb teilweise offen.

7.6 Zukünftige Forschung

Künftige Forschung sollte die abgeleiteten Designprinzipien in weiteren Solo-Operator- und KMU-Systemen replizieren. Besonders relevant sind kontrollierte Vergleiche zwischen lokaler und Cloud-Inferenz, zwischen promptbasierter und codebasierter Governance sowie zwischen unterschiedlichen Human-in-the-Loop-Positionen. Für das Fallartefakt selbst bieten sich prospektiv instrumentierte Experimente an, bei denen Hypothesen, Kontrollbedingungen, Zielmetriken und Abbruchkriterien vorab definiert werden.

7.7 Fazit

Das Löwen Codex OS demonstriert, dass ein einzelner Operator ein technisch umfangreiches AI-natives System aufbauen kann, das lokale KI, Outreach, Content, Voice, CRM, Wissensfunktionen und Automatisierung verbindet. Die wissenschaftlich wichtigste Erkenntnis liegt jedoch nicht in der Größe des Systems, sondern in seinen Widersprüchen. Hochautomatisierte Prozesse können unwirksam sein. Sauber gebaute Governance kann ungenutzt bleiben. Dokumentierte Human-in-the-Loop-Doktrinen können von der realen Codeausführung abweichen. Und ein System kann aktiv erscheinen, obwohl der fachliche Zielzustand nicht erreicht wird.

Daraus folgt die zentrale Schlussfolgerung dieser Arbeit: AI-native Operating Systems sollten nicht anhand ihrer Modellanzahl oder Automatisierungsdichte bewertet werden, sondern anhand der nachweisbaren Verbindung von Autonomie, Wirksamkeit und Governance. Erst wenn ein System fachliche Ergebnisse end-to-end beobachten, riskante Aktionen technisch begrenzen, Fehlerzustände sichtbar machen und menschliche Kontrolle an den richtigen Stellen erzwingen kann, wird aus technischer Automatisierung ein belastbares Betriebssystem.

7.8 Rückbindung an Design Science Research

Der DSR-Beitrag dieser Arbeit liegt in der Verbindung von Artefakt, Problemkontext, Evaluation und abstrahiertem Designwissen. Das Artefakt adressiert das reale Problem begrenzter Operator-Kapazität. Es demonstriert Mechanismen für lokale KI, Automatisierung und zentrale Steuerung. Die Evaluation legt sowohl produktive Stärken als auch Negativergebnisse offen. Aus diesen Befunden werden Designprinzipien abgeleitet, deren Geltungsanspruch über den Einzelfall hinaus formuliert, aber nicht als abschließend validiert dargestellt wird.

Gregor und Hevner (2013) unterscheiden unterschiedliche Arten von DSR-Wissensbeiträgen und betonen die explizite Positionierung des Beitrags. Die vorliegende Arbeit ist primär als Improvement- und Invention-nahe Fallstudie zu verstehen: Das Problemfeld AI-nativer Solo-Operator-Systeme ist praktisch relevant, während die konkrete Kombination aus lokaler KI, deterministischen Engines und Human-Governance im untersuchten Kontext eigenständig konstruiert wurde. Der stärkste Wissensbeitrag besteht in den abstrahierten Designprinzipien, nicht in der Behauptung, dass die konkrete Implementierung optimal sei.

7.9 Designprinzipien als prüfbare Hypothesen für weitere Zyklen

Die zehn Designprinzipien können in Folgestudien als prüfbare Designhypothesen operationalisiert werden. Deterministic Boundaries ließe sich etwa durch den Vergleich promptbasierter und codebasierter Schutzmechanismen evaluieren. End-to-End Observability könnte durch einen Voice-Circuit-Breaker getestet werden. Fresh Knowledge ließe sich über definierte Reindex-SLAs und Retrieval-Freshness-Metriken untersuchen. Damit wird aus der retrospektiven Fallstudie eine prospektive Forschungsagenda.

Besonders wichtig ist die Reihenfolge zukünftiger Experimente. Zunächst müssen Messbarkeit und Safety hergestellt werden, bevor Conversion optimiert wird. Ein Voice-Experiment zur Terminquote wäre methodisch schwach, solange Call-Starts und Providerfehler nicht zuverlässig beobachtet werden. Ebenso ist Content-Optimierung ohne Attribution zwischen Generierungsentscheidung und Performance nur

heuristisch. Die Arbeit leitet deshalb Instrumentierung als Vorbedingung experimenteller Optimierung ab.

7.10 Praktische Architekturagenda für Version 2

Aus der Evaluation ergibt sich ein konkretes Zielbild: Git-basierte Versionskontrolle mit reproduzierbaren Releases; Trennung von Source, Deployment und Backup; zentrale Policy-Schicht für externe Schreibaktionen; echte Authentifizierung und Rollenprüfung im Backend; globaler Kill Switch; fachliche Healthchecks; read-only Analytics-Snapshots; Reindex-Freshness-Metriken; providerbestätigte Outcomes und Circuit Breaker; risikobasierter Content-Review.

Diese Agenda ist ausdrücklich keine Beschreibung des Ist-Stands. Sie ist das aus den Befunden abgeleitete nächste Designstadium. Damit erfüllt sie die DSR-Funktion eines übertragbaren Verbesserungsvorschlags, dessen Nutzen in einem weiteren Zyklus empirisch geprüft werden kann.

7.11 Bedeutung für Solo-Operatoren und kleine Unternehmen

Die Fallstudie zeigt, dass ein Einzeloperator eine technische Breite erreichen kann, die früher mehrere Spezialrollen erfordert hätte. Der Engpass verschiebt sich jedoch von Implementierungskapazität zu Governance- und Beobachtungskapazität. Je schneller neue Funktionen gebaut werden können, desto wichtiger werden Mechanismen, die verhindern, dass veraltete, ungenutzte oder falsch konfigurierte Komponenten unbemerkt bestehen bleiben.

Für kleine Unternehmen folgt daraus keine Empfehlung, möglichst viele autonome Systeme einzusetzen. Sinnvoller ist eine selektive Automatisierung enger, gut messbarer Prozesse. Backups sind ein positives Beispiel: hoher Autonomiegrad, klarer Zielzustand, hohe Reversibilität. Regulierungsnahe Kommunikation ist das Gegenbeispiel: Außenwirkung und normative Risiken sprechen für stärkere menschliche oder technische Kontrolle.

7.12 Schlussbemerkung

Das Löwen Codex OS ist als Forschungsartefakt gerade deshalb wertvoll, weil es nicht als glatt polierte Erfolgsgeschichte erscheint. Seine produktiven Funktionen,

Fehlschläge, Messlücken und Governance-Widersprüche erlauben eine realistische Betrachtung AI-nativer Systementwicklung. Der zentrale Fortschritt der Arbeit liegt darin, diese Betriebsrealität in ein prüfbares wissenschaftliches Modell zu überführen.

Literaturverzeichnis

Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for Human-AI Interaction. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. DOI: 10.1145/3290605.3300233.

European Parliament & Council of the European Union. (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act), insbesondere Art. 14 Human oversight.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75-105. DOI: 10.2307/25148625.

National Institute of Standards and Technology (NIST). (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1.

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A Model for Types and Levels of Human Interaction with Automation. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 30(3), 286-297.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems, 24(3), 45-77. DOI: 10.2753/MIS0742-1222240302.

Arnott, D., & Pervan, G. (2012). Design Science in Decision Support Systems Research: An Assessment using the Hevner, March, Park, and Ram Guidelines. Journal of the Association for Information Systems, 13(11). DOI: 10.17705/1jais.00315.

Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. MIS Quarterly, 37(2), 337-355. DOI: 10.25300/MISQ/2013/37.2.01.

Huang, X., Liu, W., Chen, X., Wang, X., Wang, H., Lian, D., Wang, Y., Tang, R., & Chen, E. (2024). Understanding the Planning of LLM Agents: A Survey. arXiv:2402.02716.

March, S. T., & Storey, V. C. (2008). Design Science in the Information Systems Discipline: An Introduction to the Special Issue on Design Science Research. MIS Quarterly, 32(4), 725-730.

Masterman, T., Besen, S., Sawtell, M., & Chao, A. (2024). The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey. arXiv:2404.11584.

National Institute of Standards and Technology (NIST). (2024). Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. NIST AI 600-1.

Qu, G., Chen, Q., Wei, W., Lin, Z., Chen, X., & Huang, K. (2024). Mobile Edge Intelligence for Large Language Models: A Contemporary Survey. arXiv:2407.18921.

Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action Design Research. MIS Quarterly, 35(1), 37-56. DOI: 10.2307/23043488.

Xu, J., Li, Z., Chen, W., Wang, Q., Gao, X., Cai, Q., & Ling, Z. (2024). On-Device Language Models: A Comprehensive Review. arXiv:2409.00088.

Zhang, Z., Bo, X., Ma, C., Li, R., Chen, X., Dai, Q., Zhu, J., Dong, Z., & Wen, J.-R. (2024). A Survey on the Memory Mechanism of Large Language Model based Agents. arXiv:2404.13501.

Empirische Primär- und Projektdokumente

Marschall, J. / Löwen Codex OS (2026a). EXPORT_TEIL_A.md - IST-Export: Architektur, Software-/Serviceinventar, Datenhaltung und Observability. Erhebungsstand 22.08.2026.

Marschall, J. / Löwen Codex OS (2026b). EXPORT_TEIL_B.md - KI-Architektur, Orchestrierung, lokale KI und Memory. Erhebungsstand 22.08.2026.

Marschall, J. / Löwen Codex OS (2026c). EXPORT_TEIL_C.md - Voice AI, Mail-/Outbound-Engine, Content-Engine und Sales-/CRM-System. Erhebungsstand 22.08.2026.

Marschall, J. / Löwen Codex OS (2026d). EXPORT_TEIL_D.md - Negative Resultate, Automatisierungsgrad, Governance, Historie und Evidenzregister. Erhebungsstand 22.08.2026.

Marschall, J. (2025). Chapter I - Einleitung & Forschungsdesign. Unveröffentlichter explorativer Arbeitsentwurf.

Externer Reviewbogen

1. Ist die Forschungsfrage klar und durch das gewählte Design beantwortbar?
2. Ist die Design-Science-Research-Methodik nachvollziehbar angewandt?
3. Sind empirischer Befund, Betreiberbeobachtung und Interpretation sauber getrennt?
4. Sind Architektur-, Automatisierungs- und Governance-Aussagen durch Evidenz gedeckt?
5. Werden Negativergebnisse angemessen und ohne Beschönigung behandelt?
6. Sind Limitationen und Messlücken vollständig genug dargestellt?
7. Welche Aussagen benötigen zusätzliche Literatur oder schwächere Formulierungen?
8. Sind die Designprinzipien plausibel aus dem Fall abgeleitet?
9. Welche Gegenposition oder alternative Architektur fehlt?
10. Würden Sie die Arbeit in dieser Form öffentlich zitierbar veröffentlichen?

Name / Funktion: _____

Institution (optional): _____

Datum: _____

Gesamturteil / wichtigste Änderungen:

_____

_____

_____

Anhang A – Forschungsfragen-Evidenz-Matrix

RQ	Empirische Kernbelege	Evidenz	Grenze
RQ1	25 vHosts; 225 FastAPI-Routen; eingebettete relationale Datenhaltung/dateibasierte Zustandsdaten/HTTP-Kopplung; lokale KI	A/B	keine Aussage, dass die konkrete Architektur optimal ist
RQ2	L0-L5-Matrix; Mail L4; Content Publishing L5; Voice L4; Reply-Triage L2	A/B	Skala ist fallbezogene Operationalisierung
RQ3	confirm-Gates; Sandbox/Test; Suppression; fehlendes Default-Deny; kein globaler Kill Switch	A/B	kein vollständiger Penetrationstest
RQ4	über 19.000 Sends; über 1.000 Klicks; mehreren Dutzend Antworten; 0 zuordenbare Termine; über 1.500 Voice-Outcomes; Incidents	B	Kausalursachen ohne Kontrollgruppen nicht eindeutig

Anhang B – Komponenten- und Statusinventar

Komponente	Funktion	Status	Evidenz
Reverse Proxy	Reverse Proxy / statischer Host	BESTÄTIGT/ PRODUKTIV	A/B
FastAPI Backend	zentrale API, 225 Routen	BESTÄTIGT/ PRODUKTIV	A/B
Outreach Engine	Versand, Follow-ups, Verify, Reports	BESTÄTIGT/ PRODUKTIV	A/B
Open-Data-Leadquelle Autoloader	Lead-Beschaffung	BESTÄTIGT/ PRODUKTIV	A/B
Reply-Triage	lokale KI für Klassifikation/Entwürfe	BESTÄTIGT/ PRODUKTIV	A/B
Content Engine	Text/Bild/Reel/ Publishing	BESTÄTIGT/ PRODUKTIV	A/B
Content Review	manueller Review-Pfad	IMPLEMENTIERT/ NICHT AKTIV	A
Voice Agent	STT-LLM-TTS-Kette	IMPLEMENTIERT/ NICHT PRODUKTIV	A/B
Autodialer	automatische Leadwahl	PROZESS PRODUKTIV, WIRKUNG NICHT SICHTBAR	B
LEO / RAG	NLU/Retrieval/ Operatorassistenz	PRODUKTIV mit Freshness-Lücke	A/B
Dev-Agent	Sandbox/Test/Promote	IMPLEMENTIERT/ NICHT PRODUKTIV	A
Backup	DB- und Gesamtbackup	BESTÄTIGT/ PRODUKTIV	B
Watchdog	Alarmierung	BESTÄTIGT/ PRODUKTIV	A/B

Anhang C – Automatisierungs- und Human-Gate-Matrix

Prozess	Level	Human Gate	Technische Grenze
Cold-Mail Erstversand	L4	kein Gate je Sendung	Verify, Cap, Warmup, Zeitfenster
Follow-ups	L4	kein Gate	Sequenzstatus, Budget, Idempotenz
Kampagnen-Release	L4	kein zwingendes Human Gate	5 maschinelle Tore
Domain Flip	L4	Auto-Pfad ohne Prompt	DNS fail-closed, max. 4/Lauf
Open-Data-Leadquelle Scraping	L4	kein Gate	Lock/Heartbeat, Importfenster
Reply-Triage	L2	Mensch vor Versand	Entwurf-only
Reply-Send	L2/L3	Browser-confirm	serverseitig nicht erzwungen
Content-Generierung	L4	kein Gate	Validatoren, Ressourcenphasen
Content-Publishing	L5	Review deaktiviert	Plattformstatus-Polling
Voice Lead Selection	L4	kein Gate	Zeitfenster, Rate, Retry
Backup	L5	kein Gate	Scheduler
Watchdog	L1	Mensch reagiert	nur Alarm
Dev-Agent	L3 nach Bauart	explizites Promote	Sandbox/Test

Anhang D – Negative Resultate und Reparaturwissen

Fall	Symptom	Ursache	Reaktion	Stand
0-Byte outreach.db	falsche Quelle / Nullanzeigen	Existenzprüfung statt Integritätsprüfung	Mindestgröße, Priorität, read-only Filter	weitgehend behoben
VRAM-Thrashing	Content unter Ziel	verschachtelte Modellwechsel	Zwei-Phasen-Trennung	Architekturfix belegt
Encoding/Emoji	Abstürze beim Generieren/Posten	Windows-Codepage	UTF-8-Härtung	in mehreren Pfaden behoben
Voice-Degradation	Retry-Zähler laufen, Outcomes stagnieren	End-to-End-Bestätigung fehlt	Ursache nicht abschließend live verifiziert	offen
RAG Freshness	aktuelles Wissen fehlt	Index nicht nachgeführt	Freshness-Lücke identifiziert	offen
Port Health	falsches Grün möglich	Portbelegung statt Serviceidentität	bessere Prüfung existiert in anderem Modul	uneinheitlich
Webroot Backups	alte sensible Zustände in Backups	Source/Backup/Deploy nicht getrennt	aktive Datei bereinigt	strukturelles Risiko bleibt
Keine Git-Historie	Änderungen schwer rekonstruierbar	Dateikopien statt VCS	keine belegte Umstellung	offen

Anhang E – Messlücken und Forschungsgrenzen

- Keine direkte eingebettete relationale Datenhaltung-Abfrage der Live-Datenbanken im Forschungsexport.
- Keine Live-Prozess- oder Portprüfung aus der isolierten Analyseumgebung.
- Keine externen Postmaster-Daten zum tatsächlichen Inbox Placement.
- Keine Ende-zu-Ende-Latenzmessung der Voice-Kette.
- Keine Referenztranskripte und damit keine STT-Fehlerrate.
- Keine kontrollierte Eval-Harness für LLM-Qualität.
- Keine automatische Attribution von Content zu Leads oder Umsatz.
- Keine echte Randomisierung der vorhandenen Mailvarianten.
- Keine belastbare historische Gesamtzahl aller vereinbarten Termine aus direkter DB-Abfrage.
- Keine Git-basierte Versionshistorie des untersuchten Artefaktstands.

Anhang F – Prospektives Experimentprotokoll

11. Hypothese und primäre Zielmetrik vor Start schriftlich fixieren.
12. Population sowie Ein- und Ausschlusskriterien definieren.
13. Randomisierte Zuteilung mit reproduzierbarem Seed implementieren.
14. Kontroll- oder Baseline-Bedingung festlegen.
15. Mindestdauer und Abbruchregeln vorab definieren.
16. Konfiguration und Rohdaten am Ende unveränderbar einfrieren.
17. Technische Ausfälle als eigene Kategorie kennzeichnen.
18. Effektgröße und Unsicherheit berichten; Signifikanztests nur bei geeignetem Design.
19. Replikation in zweiter Periode oder zweitem Segment durchführen.
20. Erst Reliability und Safety instrumentieren, dann Conversion optimieren.

Anhang G – Begriffsglossar

Begriff	Arbeitsdefinition
AI-native Operating System	Informationssystem, in dem KI, deterministische Automation, Datenhaltung und menschliche Steuerung in Kernprozesse integriert sind.
Autonomie	Ausmaß, in dem ein Prozess ohne menschliche Einzelintervention ausgeführt wird.
Human Gate	technisch erforderliche menschliche Bestätigung je Vorgang.
Governance	Regeln, technische Kontrollen, Verantwortlichkeiten, Audit, Monitoring und Eingriffsmöglichkeiten.
Observability	Fähigkeit, technischen und fachlichen Systemzustand aus Telemetrie nachvollziehbar zu bestimmen.
RAG	Retrieval-Augmented Generation: Generierung mit vorgelagertem Wissensabruf.
Idempotenz	Eigenschaft, dass Wiederholung denselben fachlichen Effekt nicht mehrfach erzeugt.
Circuit Breaker	Mechanismus, der bei wiederholten Fehlern weitere externe Aktionen temporär stoppt.
Default-Deny	Aktionen sind standardmäßig blockiert und werden nur bei erfüllten Bedingungen erlaubt.
Freshness	Aktualität eines Wissens-, Daten- oder Indexzustands.
Aktivitätsmetrik	misst ausgeführte Arbeit, z. B. Sends oder Calls.
Zielmetrik	misst den fachlichen Endzustand, z. B. Termin oder erfolgreiche Veröffentlichung.

Anhang H – Veröffentlichung und externe Begutachtung

21. Diese Fassung vollständig selbst lesen und inhaltliche Korrekturen markieren.
22. Mindestens zwei externe Reviewer gewinnen: Information Systems/Design Science sowie AI/Software Architecture.
23. Reviewer erhalten PDF, Reviewbogen und bei Bedarf anonymisierte technische Exporte A-D.
24. Review-Kommentare dokumentieren und in Version 1.1 oder 2.0 einarbeiten.
25. Zentrale Fremdaussagen vor DOI-Veröffentlichung nochmals gegen Originalquellen prüfen.
26. Finale PDF mit Versionsnummer, Veröffentlichungsdatum, Lizenz und DOI versehen.
27. Keine Bezeichnungen wie Masterabschluss, M.Sc. oder peer reviewed verwenden, solange sie nicht tatsächlich zutreffen.
28. Nach Veröffentlichung eine kurze öffentliche Projektseite mit DOI, Abstract, Methodik und Reproduzierbarkeitshinweisen anlegen.

Anhang I – Literatur- und Theorie-Matrix

Die Matrix dokumentiert, welche theoretische Funktion die zentrale Literatur in der Arbeit erfüllt. Sie soll verhindern, dass Quellen lediglich dekorativ zitiert werden. Jede Quelle ist einem konkreten Analysebaustein zugeordnet; die empirischen Aussagen über den Löwen Codex OS stammen weiterhin aus den Projektexporten A-D.

Quelle	Theoriebaustein	Funktion in der Argumentation	Verwendung
Hevner et al. 2004	Design Science Research	Artefaktbau und -evaluation als Erkenntnismodus	Kap. I-III, VII
Peffers et al. 2007	DSRM	sechs Aktivitäten des Design-Science-Prozesses	Kap. I, III
Gregor & Hevner 2013	DSR Knowledge Contribution	Positionierung von Artefakt- und Designwissen	Kap. II, VII
Sein et al. 2011	Action Design Research	gegenseitige Formung von Artefakt und Nutzungskontext	Kap. II, VII
Arnott & Pervan 2012	DSR-Qualität	Rigor, Evaluation und Relevanz als Qualitätsprobleme	Kap. II-III
Parasuraman et al. 2000	Levels of Automation	Automatisierung als mehrdimensionale Eigenschaft	Kap. II, VI
Amershi et al. 2019	Human-AI Interaction	Korrigierbarkeit, Feedback und Kontrollierbarkeit	Kap. II, VI
NIST AI RMF 1.0	AI Governance	Governance als querschnittliche Lebenszyklusfunktion	Kap. II, VI-VII
NIST AI 600-1	Generative AI Risk	GAI-spezifische Risiken und Risikomanagement	Kap. II, VI
EU AI Act 2024	Human Oversight	risikoadäquate menschliche Aufsicht als Referenzprinzip	Kap. II, VI
Huang et al. 2024	LLM-Agent Planning	Planung, Reflection, Memory und externe Module	Kap. II, IV
Zhang et al. 2024	Agent Memory	Memory-Mechanismen und	Kap. II, IV-VI

		Langzeitinteraktion	
Xu et al. 2024	On-device LLMs	Datenlokalität, Hardware- und Ressourcenrestriktionen	Kap. II, IV
Qu et al. 2024	Edge LLMs	hybride Edge-/Cloud-Perspektive und Ressourceneffizienz	Kap. II, IV
Masterman et al. 2024	Agent Architectures	Architekturen für Reasoning, Planning und Tool Calling	Kap. II, IV

I.1 Einordnung der Literaturbasis

Die Literaturbasis ist bewusst interdisziplinär, bleibt aber um die Informationssystem-Perspektive zentriert. DSR liefert den methodischen Rahmen; Human-Automation- und Human-AI-Literatur liefert Konzepte zur Verteilung von Entscheidung und Kontrolle; Governance-Rahmen liefern Kategorien für Risiko und Aufsicht; aktuelle Agenten- und Edge-LLM-Surveys helfen, die technische Architektur präzise einzuordnen. Diese Kombination ist notwendig, weil das Artefakt zugleich Informationssystem, Automatisierungsplattform und lokale KI-Laufzeit ist.

I.2 Grenzen der Literaturbasis

Die Literaturrecherche ist keine systematische Review im PRISMA-Sinn. Die Quellen wurden problemorientiert ausgewählt, um die zentralen Konstrukte der Fallstudie abzudecken. Daraus folgt, dass die Monographie keinen Anspruch auf Vollständigkeit der Agenten-, Governance- oder Edge-LLM-Literatur erhebt. Vor einer Journal-Einreichung sollte der Theorieabschnitt durch eine formal dokumentierte Literaturrecherche erweitert werden.

Metrik	Wert	Definition/Zeitraum	Evidenz
Sendedomains	39	Konfiguration domains.json	A
Postfächer	195	5 Präfixe je Domain	A
Sendungen im Revenue-Report	über 19.000	Stand 21.08.2026	B/C
Klicks	über 1.000	reportdefinierte Aggregation	B/C
Antworten	48	kampagnenbezogen	B/C
kampagnenzuordenbare Termine	0	Revenue-Report	B/C
manuelle Übergabeliste	303 Leads	Klick + >=3x nicht erreicht oder warme Antwort ohne Termin	C
Follow-up-Beispiel	540 an einem Tag	04.08.2026	B
Klick-Follow-ups 30 Tage	293	Systemreport	B
Call-Eskalationen 30 Tage	275	Systemreport	B
Rewarm 30 Tage	307	Systemreport	B
Reminder 30 Tage	355	Systemreport	B
Metrik	**Wert**	**Definition/Zeitraum**	**Evidenz**
aktive Marken	6	Stand 22.08.2026	A
Tagesziel	18 Posts	6 Marken x 3	A
Queue-Einträge	einem dreistelligen Bestand	04.-22.08.	B
davon posted	dem überwiegenden Teil	Queue	B
pending	10	Queue	B
error	9	Queue	B
Posting-Versuche gesamt	496	02.07.-22.08.	B
Erfolge	456	daypost.log	B
Fehler	41	daypost.log	B
Publishing-Erfolgsquote	91,7 %	Erfolge / Versuche	B
Posts/Tag 04.-11.08.	12,0	96 in 8 Tagen	B
Posts/Tag 12.-22.08.	3,3	36 in 11 Tagen	B
Tage mit Soll 18	2	08. und 10.08.	B
Zeichensatzfehler	13	07.-09.08.	B

Datei nicht gefunden	8	08. und 20.08.	B
Token abgelaufen	5	ab 19.08.	B
Metrik	**Wert**	**Definition/Zeitraum**	**Evidenz**
protokollierte Outcomes	über 1.500	14.07.-12.08.	B
nicht erreicht	nahezu allen	99,94 %	B
falscher Ansprechpartner	1	0,06 %	B
Termine	0	gesamter protokollierter Zeitraum	B
Rückrufwünsche	0	gesamter protokollierter Zeitraum	B
Twilio failed	dem weit überwiegenden Teil	89,7 % aller Versuche	B
no-answer	51	3,3 %	B
busy	43	2,7 %	B
IVR/Hold/Voicemail	66	4,2 %	B
Call-Outcomes nach 12.08.	0	bis Erhebungsstichtag	B
Chefsache-Liste	300	Kappungsgrenze erreicht	B
pending Leads	87	Stand 22.08.	B

Control	**Bereich**	**Zweck**	**Erzwingung**	**Status**
Copy-Status-Gate	Mail	draft verhindert Live-Send	erzwingend	produktiv
Verify-Gate	Mail	nur sendbare Adresszustände	erzwingend	produktiv
Bounce Guard	Mail	Auto-Pause bei Schwelle	erzwingend	produktiv
Warmup Caps	Mail	Volumenbegrenzung je Inbox	erzwingend	produktiv
Wochenend-/ Zeitfenster	Mail	zeitliche Begrenzung	erzwingend	produktiv
Opt-out Vorfilter	Mail	Abmeldung vor Modellentscheidung	erzwingend	produktiv
Demo-Link-Platzhalter	Reply AI	verhindert erfundene URLs	erzwingend	produktiv
Browser-confirm	Reply Send	Einzelbestätigung	nur clientseitig	produktiv
Content Review	Content	manuelle	deaktiviert	nicht produktiv

		Freigabe		
Caption Validator	Content	formale Gültigkeit	erzwingend	produktiv
Hook Guard	Content	Overlay-Qualität	erzwingend	produktiv
Phasen-Guard	Content	keine Ollama-Aufrufe in Renderphase	erzwingend	produktiv
Zeit-/Rate-Gates	Voice	Anrufzeit und Frequenz	erzwingend	produktiv als Prozess
Booking-State-Machine	Voice	Slot/Name/ Entscheider prüfen	erzwingend	implementiert
Consent/DNC	Voice	Werbe-/ Kontaktkontrolle	fehlend	Lücke
Recording Notice	Voice	Transparenz der Aufnahme	fehlend	Lücke
Sandbox/Test/ Promote	Dev	Live-Übernahme absichern	erzwingend	ungenutzt
Anti-Truncation	Dev	abgeschnittene Dateien blockieren	erzwingend	ungenutzt
LLM-Auditor	Dev	Diff-Risiko signalisieren	nicht Promote-blockierend	ungenutzt
Read-only Fremd-DB	Data	Korruptionsschutz	erzwingend	produktiv
Kontext-Whitelist	LEO	nur freigegebene Lead-Anzeigefelder	erzwingend	implementiert
Secret Scrubbing	LEO Watch	Tokenfragmente aus Logs entfernen	erzwingend	produktiv
externer Tunnel-/Zugriffsdienst Access	Netz	vorgelagerter Zugriffsschutz	außerhalb App	laut Betreiber aktiv
Backend AuthN/AuthZ	Backend	eigene Zugriffskontrolle	unvollständig/ fail-open	Lücke
Backup 04:00/04:30	Recovery	DB- und Gesamtbackup	automatisch	produktiv
Globaler Kill Switch	Gesamtsystem	Notabschaltung	fehlend	Lücke
Git/VCS	Engineering	Diff, Branch, Rollback	fehlend	Lücke

Anhang J – Öffentliche Ergebnisübersicht

Die öffentliche Fassung berichtet Prozessmetriken nur in einer Granularität, die die wissenschaftliche Interpretation ermöglicht, ohne operative Schwellenwerte, interne Kontrollparameter oder reproduktionsnahe Systemdetails offenzulegen.

Outbound: hohes Aktivitätsvolumen und messbare Interaktion, aber im eingefrorenen Untersuchungszeitraum keine belastbare kampagnenzuordenbare Terminwirkung. Content: mehrere hundert dokumentierte Publishing-Versuche mit technischer Erfolgsquote von über 90 Prozent; eine belastbare Content-zu-Umsatz-Attribution fehlt. Voice: über 1.500 protokollierte Wahlresultate bei praktisch fehlender fachlicher Erreichung; der Befund wird als End-to-End-Reliability-Problem interpretiert.

Anhang K – Öffentliche Governance-Zusammenfassung

Starke lokale Kontrollen existieren insbesondere bei Zustandsprüfung, Rate-Begrenzung, Suppression, Validierung und einzelnen Human Gates. Die Untersuchung zeigt zugleich Lücken bei systemweiter Policy-Erzwingung, zentraler Notabschaltung, konsistenter Authentifizierung und der Kopplung technischer Aktivität an fachliche Endzustände. Konkrete interne Kontrollparameter und sicherheitsrelevante Implementierungsdetails bleiben der Private Deep Edition vorbehalten.

Anhang L – Reproduzierbarkeit und Veröffentlichungspaket

L.1 Forschungsfreeze

Der wissenschaftlich untersuchte Zustand ist auf den 22. August 2026 eingefroren. Spätere Änderungen am produktiven Löwen Codex OS dürfen nicht still in Zahlen oder Architekturbehauptungen dieser Version einfließen. Eine neue technische Erhebung erfordert eine neue Dokumentversion.

L.2 Empfohlenes Reproduzierbarkeitspaket

Für eine öffentliche Forschungsablage sollten die Monographie, die vier anonymisierten IST-Exporte, eine Quellenliste, Hashwerte und – soweit ohne Secrets und personenbezogene Daten möglich – ausgewählte Konfigurations- und Codeauszüge versioniert bereitgestellt werden. Ein vollständiger Produktivdump wäre wegen Datenschutz- und Sicherheitsrisiken ungeeignet.

L.3 Externe Reviewstrategie

Die erste Begutachtung sollte zwei Perspektiven kombinieren: Information Systems/Design Science für Forschungsdesign und Wissensbeitrag sowie AI/Software Architecture für technische Plausibilität, Reliability und Governance. Eine juristische Prüfung der Voice- und Outreach-Befunde ist separat sinnvoll, darf aber nicht mit wissenschaftlichem Peer Review verwechselt werden.

L.4 Veröffentlichungslogik

Nach Review und Revision kann die Monographie als unabhängiger Research Output mit dauerhafter Version und DOI veröffentlicht werden. Auf LinkedIn oder im Profil sollte sie als unabhängige Forschungsmonographie bezeichnet werden. Aussagen wie Masterabschluss, M.Sc. oder peer reviewed sind nur zulässig, wenn der jeweilige institutionelle Sachverhalt tatsächlich vorliegt.

Finale Checkliste vor öffentlicher Version 1.0:

- komplette Eigenlektüre und Korrektur des Autors

- mindestens zwei externe Fachreviews dokumentieren
- Literaturangaben und DOI-Daten gegen Originalquellen prüfen
- personenbezogene Daten und Secrets aus allen Anhängen ausschließen
- Hashwerte der veröffentlichten Dateien erzeugen
- Versionsnummer und Veröffentlichungsdatum auf Deckblatt und Metadaten abgleichen
- finales PDF visuell prüfen
- DOI und Lizenz erst in der tatsächlich veröffentlichten Fassung ergänzen