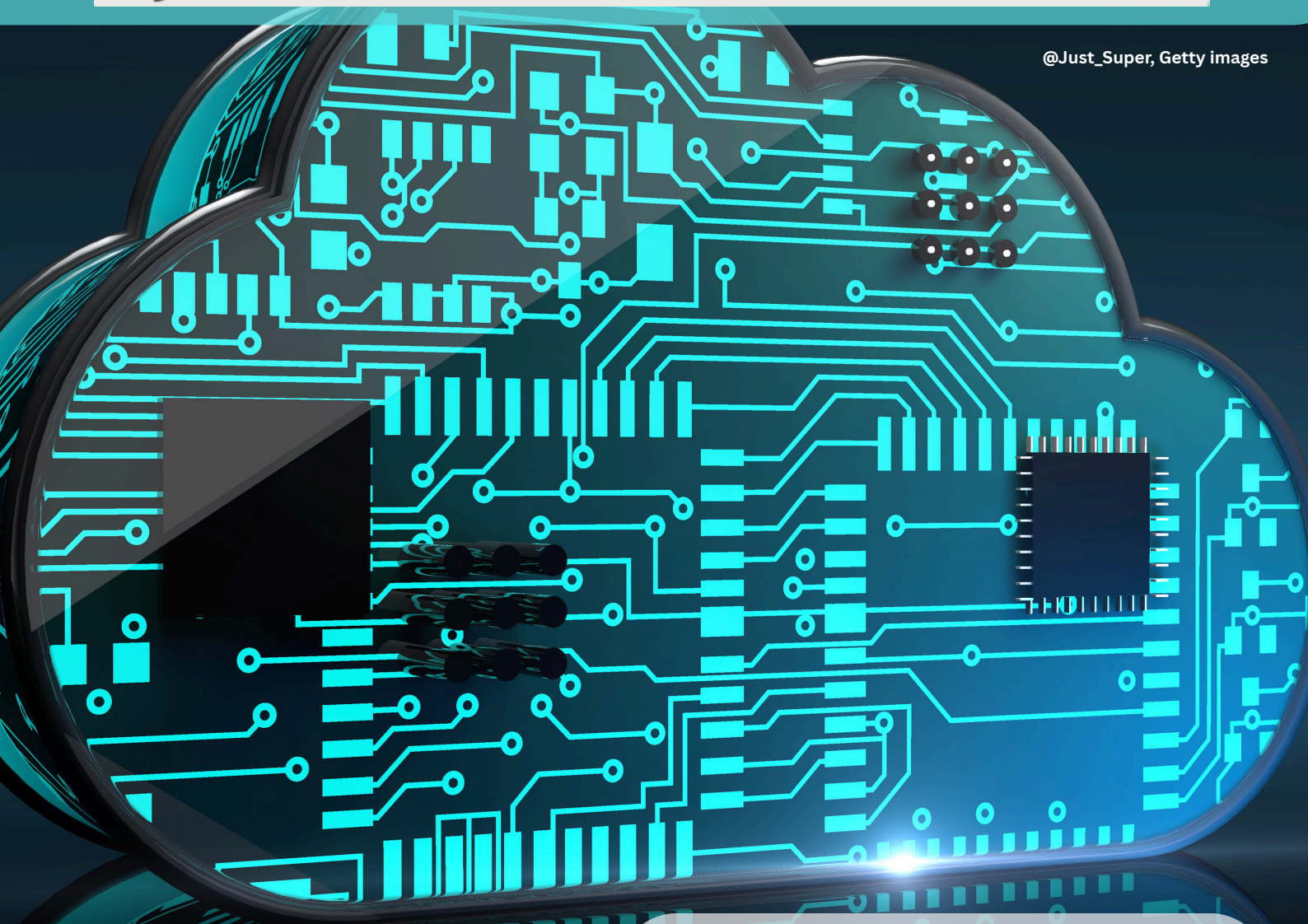
SIGS.DE

Ausgabe 4/2026

IT Spektrum

Digitaler Wandel & Software-Architektur für Profis



@Just_Super, Getty images

20 Jahre Cloud

Europäische Cloud 2026

Cloud-Kosten sind Architekturentscheidungen

Zusammen in der Cloud – Wie kollaborieren wir richtig?

Der „Macintosh Moment“ für BI

Bernhard Angerer, Bernd Gastermann

In der Software Branche bleibt kein Stein auf dem Anderen. Dieser Artikel beleuchtet Neuro-Symbolic AI und versucht einen Blick auf die – für einen erfolgreichen Einsatz von KI in Unternehmen wichtigen – Kristallisationspunkte zu liefern. Eines vorweg: Die Rolle des Architekten ist wichtiger denn je.

Während die KI-Sprachmodelle mit riesigen Schritten voranschreiten, stellt sich die Situation in Enterprise-Umgebungen anders dar. Hier wird die Neuro-symbolic AI – also die Kombination von statistischer und strukturierter Wissensrepräsentation - die Lösungen bringen. Einer der wichtigsten Komponenten ist in diesem Zusammenhang der Knowledge-Graph, welcher eine flexible und mächtige Art und Weise darstellt, Informationen zu verarbeiten. Selbiger bietet ein adäquates Gegenstück und ist ein „Sparringspartner“ für Large Language Models, welche so hervorragend auf sich ändernde Kontexte in der Fragestellung eingehen können. Auf dem Weg dort hin muss der Fokus auf architekturellen Fragestellungen und Lösungen liegen. Nur mit diesem Fokus kann man überhaupt die notwendige Informationsdichte in den Knowledge-Graphen erreichen. Gleichzeitig ist der Entwurf und die Kapselung durch klare Schnittstellen entscheidend, um auf die bevorstehenden fundamentalen Änderungen in der Softwareentwicklung an sich vorbereitet zu sein (Stichwort: Agentic Engineering).

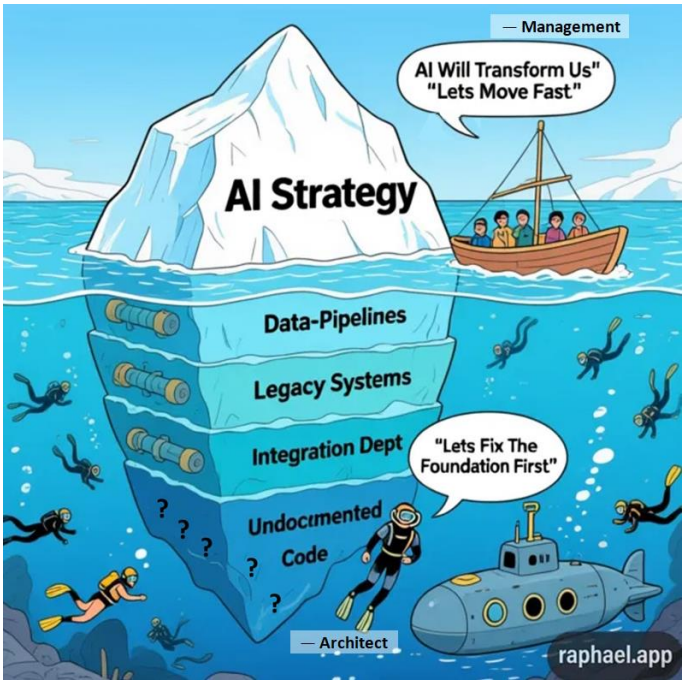
Wendepunkt

Die Entwicklungen der letzten Monate auf dem Schauplatz der Chatbots (ChatGPT/OpenAI, Claude/Anthropic, Gemini/Google, Copilot/Microsoft, Perplexity AI, Grok/xAI, etc) waren beeindruckend. Die Grenze, an der Sprachmodelle den Boden unter den Füßen verlieren und in Halluzinationen abgleiten, ist deutlich weiter nach hinten gerückt. Agentenbasierte Systeme (welche zur Laufzeit je nach Anwort des LLMs zusätzliche Informationen – meist parallel - einholen) zeichnen hier verantwortlich.

Ein Schlüsselaspekt in LLM getriebenen Lösungen ist das sogenannte **Grounding**. LLM Grounding (deutsch: Verankerung oder Erdung) bezeichnet den Prozess, bei dem die Antworten eines Large Language Models mit vertrauenswürdigen, strukturierten Datenquellen (wie eine relationale Datenbank) oder der realen Welt verknüpft werden. Ziel des Grounding ist es, die KI daran zu hindern, Fakten zu erfinden, und stattdessen präzise, aktuelle und kontextbezogene Antworten auf der Basis echter Informationen zu liefern. Dieses Grounding kann nun punktuell oder kontextuell erfolgen. Letzteres benötigt einen Knowledge-Graphen. Bevor wir uns jedoch dem Thema Graphen widmen können, ist es wichtig die Datenaufbereitung bzw. die Daten-Pipelines zu verstehen.

Neuro-Symbolische KI

Neuro-Symbolic AI bezeichnet einen Ansatz in der KI, welcher den statistischen Ansatz, der den neuronalen Netzen und somit den Large Language Models (LLMs) innewohnt, mit dem strukturellen Ansatz formaler Logik und der Wissensrepräsentation (Stichwort: Knowledge-Graphs) kombiniert. Es können somit Schwachstellen des reinen Deep Learning Ansatzes wie Nachvollziehbarkeit, Erklärbarkeit sowie mehrstufiges logisches Schlussfolgern adressiert werden. Eine weitere Dimension ist freilich der Brückenschlag zwischen klassischem Software Engineering und Data Science. Letztere Disziplin, aus welcher die aktuellen Errungenschaften der KI hervorgegangen sind, unterscheidet sich fundamental von strukturierter Softwareentwicklung. Die grundlegendsten Vokabel und Wissenskontexte um zwischen diesen beiden Disziplinen effizient Informationen auszutauschen sind nicht gegeben. Gerade im Unternehmens- und BI-Kontext wird es jedoch essentiell werden diese zwei Bereiche nicht nur nahtlos sondern auch effizient und befruchtend miteinander zu verschalten.



LLM Grounding

Die wichtigsten Aspekte des LLM Grounding:

- **Vermeidung von Halluzinationen**: Ein „ungrounded“ Modell antwortet basierend auf seinen Trainingsdaten, die veraltet sein können. Ein "gerundetes" (grounded) Modell nutzt eine externe Quelle als "Fakten-Check".
- **Verknüpfung mit externem Wissen**: Die KI greift auf Unternehmensdatenbanken, aktuelle Nachrichten, Dokumente oder Suchmaschinen zu, anstatt sich nur auf ihr internes Wissen zu verlassen.
- **Verbesserung der Relevanz**: Grounding stellt sicher, dass die KI-Antworten speziell auf den Anwendungsfall (Use Case) zugeschnitten sind.
- **RAG (Retrieval Augmented Generation)**: Eine der gängigsten Methoden für das Grounding ist RAG. Hierbei sucht das System relevante Informationen aus einer Datenbank und übergibt diese Informationen zusammen mit der Benutzerfrage an das LLM, um eine Antwort zu generieren.

Datenaufbereitung / Data-Pipelines

In modernen Enterprise-Umgebungen ist die Datenaufbereitung das Fundament für den Erfolg von KI und Large Language Models (LLMs). Während LLMs beeindruckende kognitive Fähigkeiten besitzen, hängt ihre Präzision direkt von der

Qualität und Relevanz der zugrunde liegenden Geschäftsdaten ab. Eine erste zentrale Entscheidung liegt hier darin, den ETL (Extract, Transform, Load) oder den ELT (Extract, Load, Transform) Ansatz zu fahren. Aus historischen Gründen ist ETL immer noch sehr verbreitet (vor allem ältere ETL-Tool Hersteller halten an diesem Ansatz fest). ELT ist jedoch eindeutig vorzuziehen, da bei dieser Architektur der – meist aufwendige und entscheidende – Teil der Transformation in ein und der selben Datenbank-Engine vollzogen werden kann. Dieser Ansatz hat klare Performance-Vorteile. Ein weiterer entscheidender Punkt in diesen Architekturen ist es, die Komplexität des notwendigen SQL Codes zu reduzieren. Das geht durch Aufteilung in kleine (modulare) Teile. Hierbei spielt **DBT (Open Source "Data Build Tool")** eine Schlüsselrolle, da es SQL-basierte Transformationen modularisiert und versioniert, wodurch Datenpipelines für KI-Modelle transparent und reproduzierbar werden. In verteilten Systemen (wie z.B. Microservices), ist die Datenkonsistenz jedoch eine Herausforderung. Ebenso sind die sehr verbreiteten **"polyglotten" Architekturen** oft ein Grund, warum der Implementierungsaufwand dieser Pipelines nicht nur massiv unterschätzt wird sondern explodiert [1, 2]. Message Bus Systeme fungieren hier als das „Bindeglied“. Sie ermöglichen es, Datenströme in Echtzeit zwischen isolierten Services zu erfassen und für KI-Anwendungen bereitzustellen, ohne die Systemstabilität zu gefährden. Durch die Kombination von robusten Messaging-Strukturen und präzisen Transformations-Tools wie DBT kann eine bereinigte, konsistente Datenbasis entstehen.

Wissensgraphen aka Knowledge-Graphs

Ein Knowledge-Graph ist ein "Netzwerk" aus Daten, das Informationen nicht einfach nur speichert, sondern sie in einen logischen Kontext setzt. Dinge wie Personen, Orte und Konzepte werden als Knoten dargestellt, die über beschriftete Kanten direkt miteinander verbunden sind. Im Gegensatz zu herkömmlichen Tabellen, die Daten in starre Zeilen und Spalten pressen, bildet ein Knowledge-Graph die semantischen Beziehungen zwischen diesen Objekten ab [3, 4]. Die Relation, welche in einer relationalen Datenbank (RDBMS) erst zur Laufzeit exekutiert wird (nachdem sie ein Entwickler in einem SQL Statement formuliert hat - Stichwort Foreign Keys), wird in einer Graph-Datenbank zum "First Class Citizen" und ist explizit vorhanden und Teil der Daten. Somit können "Netzwerk-Fragen" aber auch Fragen struktureller Natur direkt formuliert werden [5]. An dieser Stelle kann man bereits spüren warum Knowledge-Graphen gerade in Kombination (neuro/symbolic) mit LLMs eine entscheidende Rolle spielen.

Flexibilität durch Duck Typing

Der Begriff Duck Typing („Wenn es wie eine Ente watschelt und quakt, muss es eine Ente sein“) stammt aus der dynamischen Programmierung und beschreibt perfekt die Funktionsweise von Graph-Abfragesprachen wie SPARQL, GQL oder Cypher in einer offenen Welt [6, 7] Im Gegensatz zu SQL, das ein starres Schema voraussetzt, interessiert sich SPARQL & Co beim Pattern Matching nicht primär dafür, ob ein Knoten offiziell als „Mitarbeiter“ deklariert wurde, sondern ob er die Eigenschaften besitzt, nach denen gesucht wird. Wenn man in SPARQL und Co nach Objekten sucht, die ein ex:gehalt und eine ex:personalnummer haben (Turtle Schreibweise), wird die Abfrage alle passenden Knoten zurückgeben – völlig egal, welcher Klasse sie angehören. Diese Form des „strukturellen Typisierens“ macht SPARQL und Co sehr flexibel für die Integration heterogener Daten, da die Struktur der Daten (das Vorhandensein von Prädikaten) schwerer wiegt als eine explizite Typ-Definition. In Kombination mit der „Open World Assumption“ bedeutet das, dass Graphabfragen oft „entdeckerisch“ funktionieren: Sie finden Verbindungen basierend auf vorhandenen Merkmalen, anstatt an vordefinierten Tabellenstrukturen zu scheitern. Dieses dynamischen Charakteristika laufen auch unter den Bezeichnungen Schema-Less, Schema-on-Read oder strukturelle Typisierung.

Um nun LLMs mit Ihrer Kreativität (rein statistische Vorgehensweise welche im Grunde „immer“ halluziniert) auf den Boden der Realität zu holen ist ein Gegenstück notwendig, welches rein strukturell und somit präzise funktioniert, jedoch hohe Flexibilität ausweist, um auf die hyper-dimensionalen Gegebenheiten semantischer Kontexte adäquat reagieren zu können. Knowledge-Graphs sind dieses Gegenstück und bilden in der Kombination mit LLMs die sogenannte Neuro-Symbolic AI.

Agenten

Der Begriff des Agenten gibt es in der Softwareentwicklung seit jeher. Damit sind im Wesentlichen (meist kleine) Programme gemeint welche einen eigenen Lebenszyklus haben – sprich asynchron mit Ihrer Umwelt kommunizieren und stateful sind (d.h. selbstständig oder autonom eine state-machine abbilden oder zu einer beitragen). Die Kommunikation und Koordination erfolgt beispielsweise über einen Message-Bus. Die Orchestrierung meist über spezielle Frameworks, sowie die Aufteilung (Decomposition) durch spezielle Architekturansätze [8, 9, 10]. Nachdem sich in der pre-AI Ära kein Agentframework als Industriestandard etabliert hat (Microservices verfolgen eine stateless Philosophie) gibt es nun eine Explosion an neuen Produkten und Libraries (z.B.: LangGraph/LangChain, CrewAI, Microsoft Autogen, LlamaIndex, Agno oder OpenAI Swarm). Diese Orchestrierungsarchitekturen sind widerum nicht zu verwechseln mit der Orchestrierung von Data-Pipelines oder Machine-Learning Modellen. Hier kommen Tools wie Apache Airflow, Dagster, MLflow oder Kubeflow zum Einsatz. Man sieht schon – der „Big Ball of Mud“ lässt grüßen [11, 12, 13].

Alles spricht von Agentic AI

In modernen KI-Systemen fungieren Agents als die „Exekutive“, die das LLM von einem reinen Textgenerator in ein handlungsfähiges Werkzeug verwandelt. Während ein Standard-LLM nur auf statisches Wissen oder einfache Dokument-Snippets (klassisches RAG) zugreift, übernehmen Agents in GraphRAG-Systemen die Rolle eines intelligenten Navigators.

Hier sind die entscheidenden Gründe für ihre Bedeutung:

- **Iterative Exploration**: In einem Wissensgraph liegen Informationen vernetzt vor. Ein Agent kann entscheiden, einen Pfad zu verfolgen, Zwischenergebnisse zu bewerten und bei Bedarf „abzubiegen“. Er agiert nicht in einem einzelnen Durchlauf, sondern in Schleifen (Loops), um komplexe Zusammenhänge über mehrere Knoten hinweg zu erschließen.
- **Multi-Step Reasoning**: Agents können komplexe Anfragen in Teilaufgaben zerlegen. Bei GraphRAG bedeutet das: „Suche erst nach Entität A, finde deren Verbindung zu

B und prüfe dann die Relation zu C.“ Diese logische Kette ist für ein starres System kaum abbildbar.

- **Werkzeugnutzung (Tool Use)**: Agents können externe Tools ansteuern – etwa eine Graph-Datenbank via Cypher-Queries abfragen oder Web-Suchen starten –, um Lücken im Graph zu schließen. Sie validieren Fakten aktiv, statt nur zu halluzinieren.
- **Kontext-Management**: Da Graphen riesig sein können, filtern Agents die relevantesten Sub-Graphen heraus. Sie verhindern einen „Information Overload“ im Kontextfenster des LLMs, indem sie nur die wirklich kritischen Pfade extrahieren.

Fazit

In unternehmenskritischen Anwendungen (und nicht nur dort) sind Dinge wie Nachvollziehbarkeit, Erklärbarkeit bzw. Validierung sowie mehrstufige logische Schlussfolgerungen essenziell. Knowledge-Graphen im Zusammenspiel mit Agenten können hier profunde Lösungen schaffen. Der Weg dort hin muss jedoch genau unter die Lupe genommen werden. Die Tools und Bibliotheken, welche zu diesen Themen fast wöchentlich auf den Markt kommen, lösen meist punktuell eine bestimmte Angelegenheit, können sich jedoch nicht an einer holistischen End-to-End-Architektur orientieren.

Somit ist der Software-Architekt gefragt, hier das Schlimmste zu verhindern. Neue konsolidierte Abstraktionsebenen (wie z.B. eine API) müssen noch gefunden werden, um den Dschungel an Schnittstellen, Bibliotheken und Tools in den Griff zu bekommen.

Die Autoren

Dr DI Bernhard Angerer hat 35 Jahre Erfahrung als Entwickler, Architekt, CTO und Gründer. Beginnend mit dem ersten IBM-PC ist er fasziniert von dem Umstand, dass die Informatik immer neue Abstraktionsebenen findet (finden muss). Diese jeweils kontextbezogenen „Abstraktionsgrade“, „Modellierungsebenen“ oder „Verallgemeinerungsstufen“ bestimmen dann meist über Jahrzehnte die Interaktion zwischen Maschinen und zwischen Mensch und Maschine. Neben strategischer Unternehmensberatung (Turnaround Management) liegt sein aktueller Fokus in der Entwicklung von Reasoning-Clusters, Positiver-Geometrie sowie neuartigen AI Learning Loops. Bernhard ist Founding Member von Conxious (www.conxious.me), einem Netzwerk für Führungspersönlichkeiten und Gründer:innen, bei dem vertrauensvolle Kooperationen im Mittelpunkt stehen. https://angerer.com

Bernd Gastermann ist als selbstständiger Software Solution Architect tätig und betreibt mit GreenBear IT Solutions e.U. sein eigenes IT-Beratungsunternehmen und wirkt zudem bei Promise IBC s.r.o. als Senior Consultant und Associate Partner mit. Aktuell verantwortet er als Tech- und Teamlead die technische Entwicklung bei einem führenden Online-Plattformbetreiber für den internationalen Kunstmarkt. Mit über 16 Jahren Erfahrung in der Softwareentwicklung und einem akademischen Hintergrund in IT-Security verbindet er tiefes technisches Know-How mit strategischer Unternehmensberatung. Sein aktueller Fokus liegt auf den neuesten Entwicklungen im Bereich KI und Large Language Models sowie der Integration agenter KI-Systeme in Unternehmensprozesse. Bernd lebt und arbeitet in Wien und London.

Referenzen

[1] Robert N. Charette, The Trillion-Dollar Cost of IT's Willful Ignorance: Software Disasters are Predictable and Avoidable, IEEE Spectrum (Volume: 62, Issue: 12, December 2025)

[2] Clean Architecture: A Craftsman's Guide to Software Structure and Design, Robert C. Martin, Pearson, 2017

[3] The What and How of Modelling Information and Knowledge - From Mind Maps to Ontologies, C. Maria Keet, Springer, 2023

[4] Thomas Frisendal, Design Thinking Business Analysis - Business Concept Mapping Applied, Springer, 2013

[5] Juan Sequeda , Ora Lassila, Designing and Building Enterprise Knowledge Graphs, Springer, 2021

[6] Ricky Sun, Jason Zhang, Yuri Simione; Getting Started with the Graph Query Language (GQL): A complete guide to designing, querying, and managing graph databases with GQL, Packt Publishing, 2025

[7] Association of ISO Graph Query Language Proponents, https://www.gqlstandards.org/, 2024

[8] Juval Löwy, Righting Software - A Method for System and Project Design, December 2019, Addison-Wesley

[9] Andrew Harmel-Law, Facilitating Software Architecture: Empowering Teams to Make Architectural Decisions, O'Reilly Media, 2024

[10] Domain-Driven Design: Tackling Complexity in the Heart of Software, Eric Evans, Addison-Wesley Professional, 2003

[11] Brian Foote and Joseph Yoder, Big Ball of Mud. Fourth Conference on Patterns Languages of Programs (PLoP '97/EuroPLoP '97) Monticello, Illinois, September 1997

[12] Dave McComb, Software Wasteland: How the Application-Centric Mindset is Hobbling our Enterprises, 2018

[13] Sam Newman, Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith, O'Reilly Media, 2019