cover.mesh

Conversational Sales Intelligence

Lösungsansätze und Empfehlung zur prototypischen Umsetzung

Behrus Tabari

01.10.2025

Inhaltsverzeichnis

1. Vorwort

Dieses Dokument soll Entscheidungsträgern mit technischer Affinität sowie technischen Stakeholdern die notwendigen Informationen liefern, damit sie fundierte Entscheidungen über die Umsetzung einer oder mehrerer Lösungen treffen können.

2. Projektziele

Zielsetzung: Die Versicherungsmakler im Schunck-Vertrieb (KAMs) sollen durch KI-Agenten unterstützt werden, um **schnellen Zugriff auf kundenbezogene Geschäftsdaten** zu erhalten und **komplexe Fragen in natürlicher Sprache** beantworten zu lassen. Dadurch sollen Vertriebsteams effizienter arbeiten, um datengestützte Entscheidungen treffen zu können.

Beispiel-Fragestellungen:

- *„Wie entwickelt sich die Schadenquote bei Kunde X?“* – Verlauf der Schadensquote für einen bestimmten Kunden.
- *„Wie haben sich Prämien und Erlöse bei Kunde Y entwickelt?“* – Entwicklung der Versicherungsprämien und Erlöse über die Zeit für einen Kunden.
- *„Gibt es auffällige Vertragskonstellationen oder Potenzial für Cross-Selling?“* – Identifikation von ungewöhnlichen Vertragssituationen oder neuen Verkaufschancen im Kundenportfolio.

Diese **KI-basierten Antworten** sollen **konkret und datenbasiert** sein. Die Agenten dienen als *Conversational BI*-Tool: Anstatt Berichte manuell zu sichten, stellen KAMs Fragen und die KI liefert strukturierte Antworten oder Visualisierungen.

2.1. Technologie-Fokus

Das Projekt soll **Microsoft-Technologien** einsetzen, u.a.:

- **Microsoft Copilot** (Überbegriff für generative KI-Funktionen in Microsoft-Produkten, z.B. Dynamics 365 Sales Copilot, Power BI Copilot, Microsoft 365 Copilot).
- **Azure AI Foundry** (neue Azure-Plattform als „KI-Agentenfabrik“ zur Entwicklung und Verwaltung von KI-Anwendungen und -Agenten).
- **Dynamics 365 CRM-Copilot** (speziell Copilot-Funktionen innerhalb von Dynamics 365 Sales/CRM).
- **Power BI Copilot** (Copilot in Microsoft Fabric/Power BI für natürliche Sprache zu Datenvisualisierungen).
- **Microsoft Fabric Data Agents** (neue *Conversational Data* Agents in Microsoft Fabric, die natürliche Sprache in Datenbankabfragen übersetzen).

Durch **vergleichende prototypische Umsetzung** mit diesen Technologien soll festgestellt werden, **welcher Ansatz die beste Performance und Eignung** im Sales-Umfeld bietet.

3. Aktuelle technische und infrastrukturelle Gegebenheiten

Die für das Projekt relevanten Infrastrukturen des Gesamtkonzerns sind aktuell auf drei separate, voneinander unabhängige Azure-Cloud-Tenants verteilt. Diese Mandantenstruktur prägt die möglichen Lösungen maßgeblich, da Daten und Systeme verteilt sind.

3.1. Ecclesia-Tenant

Ecclesia-Tenant soll perspektivisch alle zentralen Anwendungen hosten. **Dynamics 365 CRM** ist bereits in einer produktiven Umgebung der Power Plattform im Einsatz. Teilweise werden **Daten aus dem Schunck-Datenplattform** schon **in das Ecclesia-CRM repliziert** (z.B. Kundendaten) um sie im CRM verfügbar zu machen. Aktuell läuft auch die **Migration des Verzeichnisdienstes (Entra)** vom Schunck- in den Ecclesia-Tenant. Mittelfristig sollen **sämtliche Anwendungen und die Datenplattform** nach Ecclesia-Tenant migriert werden.

Herausforderung: Ecclesia-Tenant hat bislang **keine separate Test-/Sandbox-Umgebung** für Experimente. Änderungen wirken direkt auf die Produktion, was heikel ist, da die Cloud-Umgebung sich noch im Aufbau befindet. Größere prototypische Entwicklungen dort könnten den Rollout der Umgebung verzögern. Zudem enthält CRM eventuell **nicht alle relevanten Daten** (viele Daten liegen nur im Schunck-Datenplattform).

3.2. Schunck-Tenant

Dieser Tenant beherbergt aktuell den Großteil der relevanten **Datenquellen**. Zentrales Element ist eine **Azure Data Factory** mit mehreren Azure SQL-Datenbanken, in denen sämtliche **Rohdaten aus Legacy-On-Premise-Systemen** gesammelt und zu **analytischen Datensichten** aufbereitet werden. Daraus speisen sich **mehrere produktive Power BI-Reports**, die Kennzahlen zu Schäden, Prämien usw. darstellen.

Herausforderung: Schunck-Tenant hat, wie Ecclesia-Tenant **keine separate Test-/Sandbox-Umgebung** für Experimente.
Außerdem ist er auf dem **Weg der Ablösung** (Migration aller Assets in den Ecclesia-Tenant, inkl. geplanter Aufbau einer neuen Datenplattform voraussichtlich auf **Microsoft Fabric**). Prototypische Arbeiten direkt hier tragen das Risiko, **Migrationsprojekte zu verzögern** oder die laufenden Prozesse zu stören. Anpassungen an der bestehenden Data Factory könnten komplex sein und müssten später ggf. in Fabric nachgezogen werden.

3.3. cover.mesh-Tenant

Dieser Mandant hat ein **Experimentier-Labor**. Hier können **beliebige Azure-Ressourcen** unkompliziert bereitgestellt werden, inkl. **Microsoft Fabric**, **Azure AI Services**, **Copilot, Power Plattform-Umgebung** und mehr. **Daten** aus dem Schunck-Tenant können mit vertretbarem Aufwand **hierher repliziert** werden.

Besonderheit: Es handelt sich um eine **isolierte Umgebung** – d.h. **Dynamics 365 CRM ist hier nicht verfügbar**, da CRM nur im Ecclesia-Tenant läuft. Die prototypischen Lösungen in cover.mesh könnten zwar technisch überzeugen, **müssten für Produktivbetrieb** später **in den**

Ecclesia-Tenant migriert werden (sobald dort die nötigen Dienste bereitstehen). Das Lab bietet jedoch maximale **Flexibilität**, ohne Risiko für die laufende Produktion.

3.4. Ausgangslage

Wir haben **eine verteilte Systemlandschaft**. In **Ecclesia-Tenant** befindet sich das **CRM-System mit Copilot-Integration**, das für KI-Anwendungen in direkter Kundendaten-Nähe genutzt werden kann. Der **Großteil der Daten liegt im Datenplattform des Schunck-Tenants**. **cover.mesh-Tenant** ermöglicht **maximale Flexibilität**.

Die Infrastruktur Verteilung beeinflusst die Lösungsansätze dahingehend, **wo** und **mit welchen Mitteln** die KI-Agenten prototypisch realisiert werden **können**. Je nach Ansatz muss man Daten quer zwischen Tenants bewegen oder auf bestimmte Technologien verzichten.

4. Mögliche Lösungsansätze

Auf Basis der genannten Ausgangslage bieten sich **drei Haupt-Lösungsansätze** für die prototypische Umsetzung der KI-Agenten an. Jeder Ansatz nutzt unterschiedliche Kombinationen der Microsoft-Technologien und hat eigene Vorzüge und Nachteile. Im Folgenden werden die Lösungsansätze vorgestellt.

4.1. Ansatz 1: Copilot-Agent in Dynamics 365 (Ecclesia-Tenant)

Idee: Nutzung der **integrierten Copilot-Funktionen in Microsoft Dynamics 365 Sales (CRM)**. Microsoft bietet *Sales Copilot*[1] Features an, die in der CRM- und Office-Oberflächen als KI-Assistent zur Verfügung stehen. Dieser kann z.B. **CRM-Daten per Sprache abfragen, Inhalte zusammenfassen und Vorschläge generieren**.
Beispiel: Ein Vertriebsmitarbeiter kann Copilot fragen *„Zeige mir alle Deals, die dieses Quartal über 50.000€ abschließen“* – Copilot durchsucht die CRM-Daten und liefert die Antwort ohne manuelle Filtersuche [2]. Ähnlich könnte ein KAM via CRM-Copilot fragen: *„Schadenquote von Kunde X?“*, vorausgesetzt diese Kennzahl ist im CRM vorhanden.

Umsetzung:

- **Dynamics 365 Sales Copilot aktivieren/konfigurieren:** Der Ecclesia-Tenant hat bereits CRM; Copilot-Funktionen müssten lizenziert (ggf. **Dynamics 365 Sales Enterprise Edition** [3]) und konfiguriert werden.
- **Datenintegration ins CRM:** Damit Copilot Fragen zu Schäden, Prämien etc. beantworten kann, müssen diese Daten im CRM **vorhanden oder verknüpft** sein. Evtl. müssen relevante Kennzahlen aus dem Schunck-Datenpool **ins CRM repliziert** werden.

Vorteile:

- **Nahtlose User Experience:** KAMs, die bereits im CRM arbeiten, können den Copilot direkt in ihren Arbeitsabläufen nutzen, da dieser kontextbezogen erscheint – etwa in

[1] Copilot for Sales - Conversation intelligence dashboard

[2] Copilot for Dynamics 365 Sales: AI-Powered Selling in 2025

[3] Preise für alle Dynamics 365-Anwendungen

Kundenakten – und sie bei Aufgaben wie E-Mail-Erstellung, Notizen oder Datenauszügen unterstützt.

- **Wenig Entwicklungsaufwand:** Viele KI-Funktionen (z.B. Zusammenfassen von Kundenkommunikation, Angebotsentwürfe) sind **fertig vorhanden** [4]. Das System ist von Microsoft vortrainiert auf Vertriebsfälle –Aktivierung statt Neuentwicklung.
- **Security & Compliance:** Da alles innerhalb Dynamics/Power Plattform bleibt, greifen die **bestehenden Rechte** und **Datenschutzeinstellungen**. Daten verlassen nicht den Tenant; Compliance (PII-Schutz etc.) ist durch Microsofts Copilot-Implementierung beachtet.

Nachteile:

- **Abdeckung der Use-Cases ungewiss:** Der Sales Copilot ist primär auf Vertriebsfälle ausgerichtet (Leads, Opportunities, Notizen). Ob er komplexe **Kennzahlen wie Schadenquote** korrekt verarbeiten kann, ist fraglich, sofern diese nicht im CRM-Modell vorhanden sind. Ohne **zusätzliche Datenintegration** stößt dieser Ansatz an Grenzen.
- **Datenabdeckung und Integrationsaufwand:** Falls wichtige Kennzahlen (Schadenquote, Prämienhistorie) **nicht im CRM stehen**, muss man sie erst dort verfügbar machen. Eine unvollständige Datenbasis würde zu unbefriedigenden Antworten führen. Copilot im CRM kann **nicht direkt auf das Datawarehouse im Schunck-Tenant zugreifen**.
- **Technische Einschränkungen:** Ecclesia-Tenant hat nur Produktion – **kein Spielraum für Experimente**. Jede Anpassung passiert live. Das birgt Projekt-Risiken (Störungen im Aufbauprozess). Zudem ist CRM ein komplexes System; **Customization** über Plugins/Skripte wäre nötig, wenn man über den Standard hinausgehen will.
- **Eingeschränkte Anpassbarkeit der KI-Logik:** Die **Copilot-Logik ist vordefiniert**. Man hat nur beschränkten Einfluss darauf, wie die KI zu ihren Antworten kommt. Maßgeschneiderte Algorithmen (z. B. spezielles Ranking von Cross-Selling-Indikatoren) lassen sich schwer integrieren – man ist auf das angewiesen, was Microsoft liefert. Der Sales Copilot zielt hauptsächlich auf **Produktivitätssteigerung im Vertrieb** (automatisches CRM-Update, E-Mail Drafts, Meeting-Zusammenfassungen). Das **freie analytische Q&A** („Frage-Antwort über Daten“) ist zwar teilweise möglich, aber ggf. nicht so ausgereift und flexibel wie dedizierte BI-Lösungen. Die Antworten könnten eher **einfach** ausfallen (Copilot könnte z.B. kaum komplexe Berechnungen wie Cross-Sell-Potenziale ohne explizite Datenpunkte liefern).
- **Lizenzkosten:** Die Nutzung von Copilot-Funktionen in Dynamics 365 kann zusätzliche Lizenzkosten bedeuten. Beispielsweise kostet Dynamics 365 Sales **Premium ca. 98 EUR/Benutzer/Monat**. Für alle KAMs summiert sich das merklich.

4.2. Ansatz 2: Copilot-Agent im Schunck-Tenant (Azure + Fabric + Power Plattform)

Idee: Umsetzung eines KI-Agents **direkt im Schunck-Tenant**, wo die relevanten **Datenquellen schon vorliegen** (Azure SQL Datenbanken, Data Factory und bestehende Power BI-Datensätze).

[4] Dynamics 365 Sales-Pläne

Anstatt Dynamics CRM zu nutzen, könnte man z.B. **Power BI und Azure OpenAI** verwenden, um Fragen in natürlicher Sprache über die vorhandenen Datensätze zu beantworten. Auch eine **Power Plattform-Lösung** (z.B. Copilot) ist denkbar, der auf die Daten zugreift.

Mögliche Umsetzungskomponenten:

- **Power BI Copilot:** Microsoft hat Copilot-Funktionen in Power BI (Teil von Microsoft Fabric) eingeführt, die es erlauben, **Fragen an den bestehenden Power BI-Modell** zu stellen. Copilot generiert dann entweder direkt eine Antwort-Visualisierung oder führt Berechnungen (DAX) aus, um die Frage zu beantworten [5]. Da im Schunck-Tenant bereits einige **Power BI Reports** mit den benötigten Kennzahlen existieren, könnte Copilot diese nutzen. Ein KAM könnte z.B. im Power BI Service fragen: *„Wie hat sich die Prämie von Kunde Y in den letzten 5 Jahren entwickelt?“*. Copilot würde auf das entsprechende Dataset zugreifen und z.B. eine Zeitreihe als Ergebnis liefern. Vorteil: **Keine Datenmigration nötig**, Nutzung vorhandener Strukturen. Nachteil: an **Fabric-Features gekoppelt** (muss im Tenant aktiviert sein).
- **Copilot Agent:** Der Schunck-Tenant erlaubt den Einsatz von **Copilot Agents** (Chatbot-Baukasten). Mit *Generative AI*-Features kann ein Copilot mittels **LLM** in natürlicher Sprache Antworten generieren. Über **Tools/Wissen** kann der Bot auf Daten zugreifen. So ließe sich ein Chatbot entwickeln, der KAM-Fragen versteht und dann im Hintergrund z.B. eine Datenbankabfrage durchführt, um Fakten (Schadenquoten, Summen) zu holen, die er in seine Antwort einbettet. Dadurch kann man im Copilot einen Agenten erstellen, der genau auf die Schunck-Daten trainiert ist. Dieser Bot könnte z.B. in **Microsoft Teams** für KAMs zugänglich gemacht werden.
- **Azure OpenAI + Cognitive Search direkt:** Alternativ könnte man auch unabhängig von Power Plattform einen eigenen kleinen Web-Chat oder Teams-Bot bauen, der **Azure OpenAI** ansteuert. Per **Azure Cognitive Search** ließen sich die Schunck-Daten indexieren und dann via **Retrieval-Augmented Generation (RAG)** dem OpenAI-Modell als Kontext bereitstellen. Das bedarf allerdings mehr eigenem Entwicklungsaufwand (Custom Code mit Semantic Kernel o.Ä.).

Vorteile:

- **Daten unmittelbar verfügbar:** Alle relevanten Daten liegen bereits im Schunck-Tenant **an zentraler Stelle**. Es entfällt komplizierte Replikation oder Synchronisation in andere Tenants. Die KI kann auf **echte Produktivdaten** zugreifen – Ergebnisse sind realitätsnah.
- **Flexibilität bei der Umsetzung:** In Azure und der Power Plattform gibt es **viele Integrationsmöglichkeiten**. Man kann den Weg wählen, der am besten passt – sei es ein Power BI-integrierter Ansatz, ein Copilot Agent oder eine maßgeschneiderte Azure-Lösung. Diese könnten genau auf die KAM-Fragen zugeschnitten werden.
- **Unabhängigkeit vom CRM:** Auch wenn das Dynamics CRM nicht benutzt wird, könnten KAMs Antworten auf ihre Fragen erhalten. Dies **umgeht eventuelle Limits** von CRM-Copilot und nutzt stattdessen die Stärken der BI-Plattform (tiefe quantitative Auswertungen).

[5] Copilot in Power BI tutorial: Discover data and ask questions

- **Keine Beeinflussung des CRM-Projekts:** Da man im Schunck-Tenant bleibt, tangiert man den im Aufbau befindlichen Ecclesia-Tenant und das CRM vorerst nicht.

Nachteile:

- **Fehlende CRM-Kontextfeatures:** Da dieser Ansatz außerhalb von Dynamics 365 stattfindet, fehlen die **kontextuellen Vorteile** dort. Beispielsweise könnte der Sales Copilot im CRM direkt Aktionen ausführen (z.B. einen Kontakt aktualisieren oder ein Meeting zusammenfassen) – solche CRM-spezifischen Automatisierungen entfallen hier. Es geht rein um analytische Q&A-Fähigkeiten, keine CRM-Automatisierung.
- **Implementierungsaufwand:** Im Vergleich zu Ansatz 1 muss hier **mehr neu entwickelt oder konfiguriert** werden. Ein Copilot erfordert das Definieren von Themen/Flows und die Integration der Datenquelle. Auch die **Power BI Copilot-Funktion** erfordert vorbereitende Arbeit (Erstellung eines geeigneten Datenmodells oder *Verified Answers* im Dataset). Es gibt mehr **Freiräume**, aber damit auch technischen Aufwand, die KI präzise auf die richtigen Daten zu lenken.
- **Produktiv-Umgebung nutzen:** Der Schunck-Tenant ist produktiv; obwohl wir keine Endnutzer-Störung riskieren, arbeiten wir dennoch **in der laufenden Umgebung mit echten Daten**. Fehler in Abfragen oder hohe Last durch KI-Nutzung könnten schlimmstenfalls laufende Prozesse beeinflussen. Ohne dedizierte Testumgebung muss man sehr vorsichtig vorgehen.
- **Zukunftssicherheit:** Dieser Tenant wird perspektivisch abgebaut. Ein Prototyp hier hätte ggf. **kurze Lebensdauer**, da langfristig alles rüber nach Ecclesia soll. Man müsste die entwickelte Lösung später portieren (z.B. in Fabric im neuen Tenant).
- **Feature Verfügbarkeit:** Einige modernere Features (Fabric, Copilot in Power BI, Azure OpenAI) müssen im Schunck-Tenant **erst enabled/zugänglich** gemacht werden. Wenn der Tenant noch nicht auf dem neuesten Stand oder berechtigt für Preview-Features ist, kann das Hindernisse schaffen.

4.3. Ansatz 3: Evaluation-Lab im cover.mesh-Tenant (Multi-Technologie-Prototyping)

Idee: Dieser Ansatz setzt auf den **cover.mesh-Tenant als isoliertes Experimentierlabor**, um **parallel mehrere KI-Agenten-Lösungen zu implementieren** – mit allen in Frage kommenden Technologien außer Dynamics CRM. Ziel ist es, in dieser geschützten Umgebung **schnell Prototypen** zu entwickeln, **verschiedene Tools auszuprobieren** (Azure AI Foundry, Fabric Data Agent, Power BI Copilot, Power Plattform Copilot Agents etc.) und **vergleichende Ergebnisse** zu erhalten. Diese **Erkenntnisse** können dann genutzt werden, um die beste Lösung auszuwählen und anschließend gezielt in die produktive Umgebung (Ecclesia-Tenant) zu überführen.

Konkret heißt das: hier können wir **Microsoft Fabric** als Datenbasis nutzen, kombiniert mit dem neuen **Azure AI Foundry Agent Service** für orchestrierte KI-Agenten. Zudem stehen uns dort **Power BI Copilot** und **Fabric Data Agents** zur Verfügung, um natürliche Sprachabfragen auf die Daten durchzuführen.

Datenhaltung mit Fabric: Microsoft Fabric bietet eine einheitliche Analytics-Plattform. In cover.mesh-Tenant könnten wir einen **Lakehouse** oder **Data Warehouse** erstellen und die

erforderlichen Daten aus Schunck-Tenant **dorthin laden**. Das ist unkompliziert, da Fabric die Integration mit verschiedenen Datenquellen unterstützt. Einmal dort, können die Daten von **Fabric-Diensten** und auch **Data Agents** genutzt werden.

- Ein **Beispiel**: Wir richten in Fabric einen Lakehouse oder Warehouse ein, der Tabellen *Kunden*, *Schäden*, *Prämien* enthält. Darauf definieren wir in Power BI einen **semantischen Modell-Layer** (Measures für Schadenquote, Wachstumsraten etc.). Diese Umgebung dient dann als Grundlage für *Conversational BI*.

Umsetzung: cover.mesh bietet maximale Freiheit, daher könnten **mehrere Ansätze parallel gebaut** werden:

- **Fabric Data Agent-Prototyp:** Aufbau einer **Microsoft Fabric** Umgebung mit einem Lakehouse oder Warehouse, in das die relevanten Kundendaten aus dem Schunck-Tenant **repliziert** werden. Dann Erstellen eines **Fabric Data Agent** darauf. Ein Fabric Data Agent erlaubt es, in natürlicher Sprache Fragen an die Daten in Fabric zu stellen – er übersetzt intern die Anfrage in SQL- oder DAX-Queries, führt sie aus und liefert die Antwort zurück [6]. Beispielsweise: *„Was ist die durchschnittliche Schadenquote für Kunde X in den letzten 5 Jahren?“* -> Der Data Agent greift auf die verknüpften Tabellen zu, berechnet die Quote und antwortet mit dem Wert und ggf. Erläuterung. Diese Agents können auch in Power BI oder eigene Apps **eingebettet** werden. Vorteil: Sie nutzen direkt die **vollständigen Datenbestände** (auch Detailtabellen) und bieten *“Transparency”* – man kann die generierte Query einsehen.
- **Power BI Copilot-Prototyp:** In Fabric (oder auch separat) könnte man ebenso **Power BI Copilot** testen, ähnlich wie in Ansatz 2. Mit einem replizierten Datensatz der Schunck-Daten ließe sich ein **Power BI Bericht** bzw. Datenmodell erstellen und Copilot daran ausprobieren. Man kann hier experimentieren, wie gut die **Fragen** der KAMs interpretiert werden und welche Visualisierungen produziert werden. Die Ergebnisse lassen sich mit dem Data Agent vergleichen (z.B. Antwortqualität, Benutzererlebnis).
- **Azure AI Foundry Agent:** Azure AI Foundry ist eine neue Plattform von Microsoft, um **eigene KI-Agenten umfassend zu entwickeln, orchestrieren und bereitzustellen** [7]. In cover.mesh könnte man ein **Foundry-Projekt** aufsetzen: Die Schunck-Daten als Wissensbasis einbinden (Foundry hat Konnektoren zu **über 1.400 Datenquellen inkl. Microsoft Fabric**). Dann einen **Agenten entwerfen**, der z.B. auf **deutschsprachige Anfragen** optimiert ist und Tools/Aktionen definieren, wie er die Daten abfragt. Foundry’s Agent Service erlaubt sogar **Multi-Agenten-Orchestrierung** – z.B. könnte ein Agent Sprache verstehen, ein anderer führt die SQL-Abfrage aus, etc., koordiniert in einem Workflow. Mit Foundry ließe sich auch die Integration in **Microsoft Teams** oder andere Kanäle gleich erproben. Dieser Weg erfordert die meiste **Entwicklungsarbeit und Expertise**, bietet aber potenziell die **mächtigste Lösung**: Eine maßgeschneiderte KI, die auf **eigene Daten und Prozesse** trainiert und voll kontrollierbar ist (Logging, Governance mit Entra ID für Agenten etc. eingebaut).
- **Copilot-Agent:** Vergleichbar mit dem Schunck Tenant Copilot.

---

[6] Microsoft Fabric Data Agents: Conversational AI Insights

[7] Azure AI Foundry: Your AI App and agent factory

- **Weitere Tools:** Im Lab könnten noch weitere Dinge getestet werden, z.B. ein **Einsatz von Azure Cognitive Search** für unstrukturierte Daten, Azure Cognitive Services etc., falls relevant. Hauptaugenmerk liegen aber auf den oben genannten neueren MS-Technologien.

Vorteile:

- **Maximale Flexibilität & Innovationskraft:** Dieser Ansatz erlaubt es, **sämtliche Optionen ungebremst auszuprobieren**. Man kann **parallel lernen**, welche Lösung am besten funktioniert. Dadurch gewinnt man schnelles Feedback: z.B. zeigt sich, ob der Fabric Data Agent qualitativ bessere Antworten liefert als Power BI Copilot, oder ob Azure AI Foundry Mehrwert gegenüber Power Plattform bietet.
- **Kein Risiko für den Produktivbetrieb:** Alles passiert im vom Tagesgeschäft entkoppelten Tenant. Fehler, Lasttests oder experimentelle Features gefährden nicht die laufenden Systeme.
- **Breite Vergleichbarkeit:** Da alle Prototypen mit **denselben Daten** arbeiten, lassen sich die **Ergebnisse fair vergleichen**. Kriterien wie Antwortqualität, Implementierungsaufwand, etc., können durch direkte Gegenüberstellung der Prototypen evaluiert werden.
- **Zukunftsorientiertes Lernen:** Das Team sammelt Know-how in brandneuen Technologien (Foundry, Fabric etc.), was langfristig für den Konzern wertvoll ist – unabhängig davon, welche Einzellösung final gewählt wird. Viele dieser Fähigkeiten (z.B. Foundry Agentenbau) könnten später auch für andere Use-Cases im Unternehmen genutzt werden.
- **Schnelle Iteration:** In einer Sandbox kann man rasch Änderungen vornehmen, Zwischenergebnisse austesten und Zusatzzugriff erhalten. Bemerkt man z.B., dass ein Bestimmtes Datenfeld fehlt, kann man es in der Replikation ergänzen, ohne monatelange Change-Prozesse. Das **beschleunigt die Entwicklung**.
- **Governance & Sicherheit kontrollierbar:** Interessanterweise bieten die Lösungen trotz aller Flexibilität auch **Enterprise-Governance** (Überwachung, Richtlinien, Identity) out-of-the-box. Das heißt, selbst im Prototyp kann man schon Mechanismen testen, die später für den produktiven Betrieb wichtig sind (Logging aller Agentenentscheidungen, Berechtigungsprüfung bei Datenzugriff etc.). Dies reduziert das Risiko, dass man am Ende eine Lösung hat, die zwar funktioniert, aber nicht compliance-gerecht ist.

Nachteile:

- **Keine direkte Produktivitätswirkung für Endnutzer (kurzfristig):** Da es vom Produktivsystem losgelöst ist, haben KAMs zunächst **keinen Zugriff** auf diese Prototypen im Echtbetrieb. Es dient „nur“ der Evaluation. Sollten schnelle Nutzengewinne gewünscht sein, müsste man parallel dennoch was in den Produktiv-Tenants anbieten.
- **Mehrarbeit für Produktivsetzung:** Was im Lab funktioniert, muss später in den Ziel-Tenant (Ecclesia) übertragen werden. Das heißt, die Integration nachziehen oder auf den neuen Datenplattform-Stack migrieren. Möglicherweise lassen sich nicht alle Prototyp-Komponenten 1:1 überführen. Beispielsweise wenn man im Lab einen Data Agent baut,

aber die Zieldatenplattform anders strukturiert ist, ist Anpassung nötig. Dieser Mehraufwand ist einzukalkulieren.

- **Begrenzter CRM-Bezug:** Da Dynamics 365 im cover.mesh-Lab fehlt, kann man **CRM-spezifische Funktionen** (wie Sales Copilot) nicht testen. Sollte sich herausstellen, dass CRM-Integration unerlässlich ist, müsste man im Nachgang doch noch D365 einbeziehen. Möglichkeit: Ergebnisse aus Lab werden mit kleinerem CRM-Proof-of-Concept ergänzt, um zu validieren, wie gut Copilot dort performt.
- **Ressourcenbedarf:** Die Kehrseite der Freiheit ist der Aufwand. Parallel mehrere Prototypen bauen erfordert ein entsprechend aufgestellten Teams und mehr Arbeitsaufwand. Es müssen Experten für Azure, Fabric, Power Plattform, Azure AI etc. zusammenarbeiten. Das ist nur gerechtfertigt, wenn genug Entwickler/KI-Experten verfügbar sind. Alternativ könnte man sich zunächst im Lab auf ein oder zwei besonders vielversprechende Technologien konzentrieren und die übrigen Kandidaten zu einem späteren Zeitpunkt ergänzen.

5. Risiken der Lösungsansätze

Zusammenfassend lassen sich die zentralen Risiken der Lösungsansätze wie folgt darstellen.

- **Ansatz 1:** *„Bringt Risiken für den neuen zentralen Tenant."* – Da der Ecclesia-Tenant noch im Aufbau ist, könnte das Hantieren mit Prototypen dort **Projekt-Ressourcen binden** und den **Aufbau verlangsamen**. Zudem steht nur die Produktivumgebung zur Verfügung; Experimente könnten unerwünschte Nebeneffekte haben. Weiterhin besteht Unsicherheit, ob das **CRM umfassend genug mit Daten befüllt** ist, um alle KAM-Fragen abzudecken. Falls nicht, ist der Ansatz evtl. von vornherein limitiert. Technologie-seitig ist man auf die Möglichkeiten beschränkt, die **Dynamics 365 + Copilot out-of-the-box** bieten – tiefergehende Analysen (z.B. komplexe Cross-Selling-Mustererkennung) könnten dort schwieriger umzusetzen sein.
- **Ansatz 2:** *„Kollidiert potenziell mit Migrationsplänen."* – Der Schunck-Tenant wird mittelfristig **aufgelöst und migriert**. Ein prototypischer Ausbau dort (etwa Installation neuer KI-Dienste oder größere Datenmodell-Änderungen) könnte die **Migration verzögern** oder zu **Doppelarbeit** führen, wenn alles im Ecclesia-Tenant neu aufgebaut werden muss. Außerdem läuft man Gefahr, die **produktive Datenplattform zu beeinflussen**: Änderungen für den Prototyp (z.B. neue Views, Performance Tuning für KI-Abfragen) wirken auf dieselben Datenbanken, die täglich für Berichte genutzt werden, was Stabilität und laufende Reports beeinträchtigen könnte. Schließlich hat Schunck noch **kein Fabric und keine CRM-Funktionen** – einige modernere MS Copilot-Features wären dort evtl. nicht sofort nutzbar.
- **Ansatz 3:** *„Bietet größtmögliche Flexibilität, aber kein direkter Produktivnutzen."* – Die cover.mesh-Umgebung ermöglicht zwar nahezu **alles auszuprobieren** (was positiv ist), allerdings sind die Ergebnisse zunächst **nur im Labor** verfügbar. Eine spätere **Produktivnahme hängt von externen Faktoren** ab: Der Ecclesia-Tenant muss ausreichend weit sein (z.B. Fabric implementiert, CRM bereit für Integration), und die Schunck-Migration sollte fortgeschritten sein, bevor man ernsthaft ausrollt. In der Zwischenzeit könnten **Parallelwelten** entstehen – z.B. arbeiten KAMs produktiv

weiterhin mit Reports, während im Lab schon Chatbots tolle Antworten geben, aber eben nur Demo-Charakter haben.

6. Bewertung der Lösungsansätze (Kriterien Vergleich)

Im Folgenden werden die **drei Hauptansätze** systematisch anhand der vorgegebenen **Bewertungskriterien** beurteilt. Die Kriterien sind: **KI-Qualität**, **Umsetzungskomplexität**, **Kosten**, **Erweiterbarkeit**, **Prototypische Umsetzung & Vergleichbarkeit** sowie **Strategische Fit**.
Informationen zu **Risiken** sind in Abschnitt 4 zu finden.

Zur besseren Übersicht zeigt **Tabelle 1** eine vergleichende Gegenüberstellung. Anschließend werden die wichtigsten Punkte je Kriterium erläutert.

Tabelle 1: Gegenüberstellung der Ansätze nach Bewertungskriterien

Kriterium	Ansatz 1 Ecclesia-Tenant	Ansatz 2 Schunck-Tenant	Ansatz 3: cover.mesh-Tenant
KI-Qualität *Antwortgüte, Möglichkeiten*	Mittel bis hoch. Begrenzt auf CRM-Daten. Für Sales-Insights optimiert (z.B. Interaktionen, Opportunities) und generische Hilfen (Zusammenfassungen, E-Mails) Analytische Tiefenfragen zu Kennzahlen nur beantwortbar, wenn im CRM vorab hinterlegt. Kann qualitativ durch CRM-Kontext punkten, aber lässt evtl. Detailtiefe vermissen.	Potenziell hoch. Zugriff auf alle Rohdaten ermöglicht präzise Berechnungen. Antwortqualität ist aber nur so gut wie die Integration (Prompt-Design, Datenaufbereitung, Indexqualität etc.). Power BI Copilot liefert gute Visualisierungen und Fakten aber auf deren Inhalt limitiert.	Voraussichtlich am höchsten, da mehrere moderne KI-Technologien genutzt, kombiniert und optimiert werden können. Foundry-Agenten erlauben Fine-Tuning und orchestrierte Abläufe, was die Antwortqualität steigern kann. Fabric Data Agent kann komplexe Fragen nachvollziehen und aus mehreren Datenquellen kombinieren. Durch iterative Verbesserung im Lab dürfte die beste Antwortqualität erreichbar sein.
Umsetzungskomplexität *Implementierungsaufwand*	Gering bis mittel. Viele Funktionen sind „fertig“ vorhanden, man muss sie nur aktivieren/konfigurieren. Zusätzlicher Aufwand entsteht, wenn Daten erst ins CRM eingebracht oder Copilot angepasst werden muss. CRM-Customization erfordert spezielle Kenntnisse, aber	Mittel bis hoch. Hier ist Integrationsarbeit nötig: egal ob Copilot oder Custom Solution – Datenanbindung, Sicherheit usw. müssen konzipiert werden. Nutzung von existierenden Tools (Power BI, …) spart Code, aber verlangt Konfiguration und ggf. Scripting.	Hoch (aber im kontrollierten Rahmen). Mehrere Prototypen parallel erfordern erhebliche Koordination und technisches Know-how in diversen Services. Foundry, Fabric etc. haben Lernkurven. Allerdings kommt man im Lab-Modus schneller voran, da weniger Abstimmungs- und Freigabeprozess nötig. Die Komplexität zeigt

	insgesamt weniger Neu-Entwicklung.		sich v.a. darin, später die besten Elemente zu einem Endprodukt zu verschmelzen.
Kosten *Lizenz, Betrieb*	Lizenzgetrieben. Zusätzliche Dynamics 365 KI-Funktionalität könnte Lizenz-Upgrades bedeuten (z.B. Sales Premium für jeden Nutzer) Laufende Cloudkosten minimal höher. Keine neuen Azure-Dienste nötig. Kosten transparent pro Nutzer.	Nutzungs- und entwicklungsgetrieben. Azure OpenAI Service läuft pay-per-use. Copilot erfordert geeignete Lizenzen. Power BI Copilot kommt mit Fabric, was Power BI Premium Kapazität erforderlich macht (Kosten abhängig von Kapazitätsgröße).	Entwicklungskosten für Prototyping hoch, aber dafür gezielt investiert für Erkenntnisse. Im Lab selbst fallen v.a. Cloudkosten für die verschiedenen Dienste (Fabric Kapazität, OpenAI Usage, Storage, ...) an. Für eine spätere Prod-Lösung könnte man durch die gewonnenen Erkenntnisse effizienter lizenzieren. Insgesamt zunächst höherer Aufwand, langfristig kann Optimierung erfolgen (z.B. unnötige Dienste werden verworfen).
Erweiterbarkeit *Flexibilität für Änderungen, Zukunftsfähigkeit*	Eingeschränkt. Man ist im Microsoft-Standard des CRM-Copilot gefangen. Anpassungen an neue Anforderungen (andere Datenquellen, neue Use Cases außerhalb Sales) sind schwer, da CRM-Copilot black-box-artig arbeitet. Allerdings wird Microsoft die Copilot-Fähigkeiten im CRM selbst weiterentwickeln (Updates kommen automatisch). Für unseren Zweck heißt das: begrenzt auf alles, was in CRM passt. Nicht direkt wiederverwendbar für andere Abteilungen (z.B. andere Geschäftsbereiche ohne CRM-Zugang).	Moderat. Eine selbst aufgebaute Lösung (Bot oder Ähnliches) kann relativ frei erweitert werden – man kontrolliert die Komponenten. Beispielsweise lässt sich ein Copilot später um weitere Themen oder Datenquellen ergänzen. Auch ein Wechsel der KI (z.B. anderes Modell) wäre unter eigener Regie machbar. Die Kehrseite: Die Lösung hängt stark von der Eigenentwicklung ab – Wartung und Weiterentwicklung liegen beim Team. Dafür kann man sie aber auch gezielt an neue Bedürfnisse anpassen und z.B. in die neue Fabric-Infrastruktur migrieren, wenn Schunck->Ecclesia vollzogen ist.	Sehr hoch. Da in diesem Ansatz verschiedene moderne Tools genutzt werden, ist die Architektur zukunftsfähig. Foundry z.B. ist ausgelegt auf kontinuierliche Erweiterung und hat eingebaute Mehrmandanten-Fähigkeit, Multi-Agent-Kollaboration etc. Neue Datenquellen oder Anwendungsfälle könnten integriert werden (z.B. ein weiterer Agent für Underwriting, der neben KAM-Agenten läuft). Auch skaliert dieser Ansatz perspektivisch am besten, weil er nicht monolithisch ist: Man könnte Teile austauschen. Insgesamt bietet Ansatz 3 die meiste Zukunftssicherheit, da er Technologien einsetzt, welche Microsoft erkennbar als

			strategisch einstuft (Fabric, Foundry).
Prototypische Umsetzung & Vergleichbarkeit	Schwierig. Ohne Testsystem im CRM müsste man in Prod testen. Die Vergleichbarkeit leidet darunter, da man diese Option nicht gleich tief ausprobieren kann wie die anderen.	Begrenzt möglich. Man kann einen POC im Schunck-Tenant aufbauen. Vergleich mit anderen Lösungen ist aber schwer, da man in der Produktionsumgebung nicht beliebig herumprobieren möchte. Insgesamt ist die Ansatzerprobung eingeschränkt, da man vorsichtig sein muss.	Exzellent. Der Ansatz 3 ist darauf ausgelegt, Prototypen zu bauen und zu vergleichen. Man kann in kurzer Zeit mehrere Varianten implementieren und direkt nebeneinander evaluieren. Die Kriterien Einhaltung kann unmittelbarer gemessen werden. Für eine fundierte Entscheidungsgrundlage erfüllt dieser Ansatz das Kriterium am besten – es ist ja sein Kernzweck.
Strategische Fit	Passt zur Zukunft – gute Integration in langfristige Systeme, aber aktuell noch unreif.	Übergangslösung, da Schunck-Tenant perspektivisch migriert wird.	Visionär und zukunftsweisend – entspricht der Strategie, moderne KI und Datenplattformen einzusetzen.

Erläuterungen zu ausgewählten Kriterien:

- **KI-Qualität:** Hier sticht insbesondere die **Datenbreite und Kontextualität** hervor. Ansatz 1 könnte qualitativ hochwertige Antworten im Vertriebs-Kontext liefern. Jedoch für zahlengetriebene Fragen ist er nur so gut, wie das CRM gefüttert wurde. Ansatz 2/3 dagegen können die **vollen historischen Daten** auswerten; ein Data Agent kann z.B. quer über Vertrags- und Schadensdaten suchen, was Copilot allein nie könnte. Ferner erlaubt Foundry eine **Feinabstimmung der Antworten** (z.B. bestimmte Begriffe bevorzugen, Rechenlogiken hinterlegen), was die Qualität steigern kann. Insgesamt wird erwartet, dass **Ansatz 3 > Ansatz 2 > Ansatz 1** in puncto inhaltlicher Tiefe rangiert, aber auch, dass **Ansatz 1 im Anwendungsfeld Sales einige einzigartige Qualitätsaspekte** (CRM-Kontext) hat, die die anderen nicht bieten.

- **Umsetzungskomplexität:** Hier hat **Ansatz 1 klar den Vorteil**, da wenig selbst gebaut wird. **Ansatz 3** ist am komplexesten, weil mehrere Tech-Stacks involviert sind. Wichtig ist aber: Komplexität im Lab ist **beherrschbar**, da dort Ausprobieren zum Prozess gehört, während Komplexität in Prod (Ansatz 1/2) heikler ist, weil dort Fehler ernste Konsequenzen haben können. Also lieber Komplexität in einer Sandbox ausleben, als im Produktivsystem experimentieren – das spricht wiederum für Ansatz 3.

- **Kosten:** Die Kostenmodelle unterscheiden sich deutlich: CRM-Copilot und Copilot kosten pro User (kalkulierbar pro Jahr), während Azure OpenAI & Co. **verbrauchsabhängig** sind. Bei Letzteren hat man jedoch den Vorteil, dass **Kosten direkt mit Nutzung korrelieren** – kein Gebrauch, keine nennenswerten Kosten. Bei

Lizenzen zahlt man auch für ungenutzte Möglichkeiten. In der prototypischen Phase sind Kosten nachrangig, aber für den Rollout muss berücksichtigt werden, was es bedeutet, z.B. Foundry Agents dauerhaft zu betreiben. Insofern könnte eine **Kombilösung** auch aus Kostensicht optimieren: z.B. **seltene komplexe Fragen** laufen über einen Azure-Dienst (verbrauchsbasiert), alltägliche Aufgaben deckt der Copilot (fixe Lizenz) ab.

- **Erweiterbarkeit:** Hier punktet Lösung 3 deutlich. Aufgrund des modularen Aufbaus (separate Agenten und Tools) kann man neue Datenquellen, neue Fragetypen etc. leicht hinzufügen. Lösung 2 lässt sich ebenfalls erweitern, allerdings ist der Zugriff auf die neuesten und Preview-Technologien in der Produktivumgebung nur eingeschränkt möglich. Lösung 1 ist am wenigsten flexibel – hier ist man darauf angewiesen, was Microsoft an Konfigurationsmöglichkeiten bietet.
- **Prototypische Umsetzung & Vergleichbarkeit:** Lösung 3 ist wie geschaffen dafür. Hier kann man iterativ vorgehen, A/B-Tests machen, auch mal scheitern und neu beginnen, was in Produktivumgebungen nicht ginge. Lösung 2 ist okay für Prototyping, muss aber auf produktiven Daten arbeiten. Immerhin kann man dort eingrenzen, dass der Bot nur Lesezugriff hat, was das Risiko mindert. Lösung 1 ist prototypisch am schwierigsten – ohne dedizierte Test-CRM-Umgebung müsste man in Ecclesia-Tenant vorsichtig testen. Insgesamt gewinnt hier also Ansatz 3.

- **Strategische Fit:** Ein Kriterium, das indirekt aus den anderen folgt, aber wichtig ist: Wie fügt sich die Lösung in die Gesamt-IT-Strategie der Ecclesia Gruppe? Ecclesia-Tenant ist die zukunftstragende Plattform. Lösung 1 wäre also später direkt im zentralen System nutzbar – was positiv ist, aber aktuell sind dort die Daten nicht vollständig. Lösung 3 basiert auf dem Innovationshub, was gut zur **cover.mesh-Strategie** (Fokus auf KI und Automatisierung) passt. Die dort gewonnenen Erkenntnisse kann man später nutzen, um in Ecclesia- oder cover.mesh-Tenant produktive Lösungen auszurollen. Lösung 2 hat wenig strategischen Charme, außer dass sie Quick Wins auf aktuellen Daten liefert – aber sie muss vermutlich migriert werden.

- **Risiken:** Alle Ansätze haben beherrschbare technische Risiken, da Microsoft-Technologien verwendet werden, die grundsätzlich **sicher und supportet** sind. Unterschiede liegen eher im **Projekt- und Adoptionsrisiko**. Ansatz 1 und 2 riskieren, durch äußere Umstände (Migration, Tenant-Aufbau) gestoppt zu werden. Ansatz 3 riskiert, viel zu produzieren ohne direkt Business Value zu liefern, was manchmal zu Akzeptanzproblemen bei Stakeholdern führen kann („Forschungslabor ohne greifbares Ergebnis“ – dem muss man entgegenwirken, indem man die Ergebnisse schnell präsentiert und eine Roadmap zur Umsetzung hat).

7. Empfehlung

Empfohlene Vorgehensweise: Auf Basis der Analyse wird **eine Kombination aus Ansatz 3 (Evaluation Lab) und ausgewählten Elementen der Ansätze 1/2** empfohlen. Konkret schlägt das Konzept folgendes vor:

- **Phase 1:** Einrichtung eines KI-Experimentierlabors im cover.mesh-Tenant (Ansatz 3) zur schnellen Umsetzung mehrerer Prototypen (Data Agent, Foundry, etc.) und Vergleich der Ergebnisse.

- **Phase 2:** Basierend auf den Lab-Ergebnissen – Auswahl des besten Lösungsansatzes oder einer sinnvollen Kombination. Diese Lösung **rollout-fähig machen**, d.h. in den Ecclesia-Tenant übertragen (neue Datenplattform und CRM-Integration berücksichtigen).

- **(Optional parallel):** Falls kurzfristige Nutzen erforderlich, eine **kleine produktive Umsetzung** im Schunck-Tenant vornehmen (z.B. Copilot in Power BI aktivieren), um KAMs bereits jetzt einfache KI-Unterstützung zu bieten, ohne das Lab zu ersetzen.

Begründung der Empfehlung:

- **Lernkurve und Innovationsgewinn**: Durch Phase 1 im Lab gewinnt das Team **wertvolle Einsichten, ohne Risiko**. Man sieht, was technisch machbar ist und was die KAM-Fragen am besten beantwortet. Diese Erkenntnisse reduzieren Fehlinvestitionen – ehe man in eine einzige Richtung alles setzt, weiß man schon, ob sie funktioniert.
- **Fundierte Entscheidungsgrundlage**: Die Entscheidungsträger erhalten ein **vergleichendes Ergebnis** der Lösungsansätze (z.B. Demo aller Prototypen, Bewertungs-Matrix), was genau dem Zweck des Projekts entspricht. Damit kann man **objektiv den „Sieger" küren** oder auch entscheiden, dass mehrere Komponenten kombiniert werden sollen.
- **Zukunftssichere Architektur**: Wahrscheinlich wird sich zeigen, dass **moderne cloudbasierte KI-Services (Foundry, Fabric)** den größten Nutzen bieten. Indem man diese im Lab validiert und dann in den **Ziel-Tenant (Ecclesia)** überführt, stellt man sicher, dass die endgültige Lösung auf dem **zukünftigen Betriebsmodell** des Konzerns aufsetzt (zentraler Tenant, ggf. Fabric als Datenplattform, Integration mit CRM). Das vermeidet Aufbau von technischen Schulden in einem alten Umfeld.
- **Integration von CRM im richtigen Moment**: Die Empfehlung schließt CRM nicht aus, sondern verschiebt es in Phase 2: Sobald die KI-Lösung klar ist, kann man schauen, **wie man CRM-Daten andockt**. Beispielsweise könnte die finale Umsetzung ein Foundry-Agent sein, der sowohl Fabric (für die Kennzahlen) als auch Dataverse (CRM-Daten) als Quellen hat – so etwas ließe sich nach Lab-Phase designen. Oder man entscheidet, CRM-Copilot bleibt separat für seine Stärken, und man führt zwei Co-Piloten parallel (Sales Copilot + Data Copilot). Diese Entscheidung kann nach evidenzbasierter Abwägung getroffen werden.
- **Quick Win Möglichkeit**: Falls politisch oder praktisch nötig, kann mit wenig Aufwand **bereits jetzt eine Teilfunktion eingeführt** werden (z.B. Q&A in Power BI, was nur Konfiguration ist). Dies würde den KAMs zeigen, dass das Projekt Fortschritte bringt, und ihr Feedback einholen. Die Empfehlung schätzt aber, dass die KAMs durchaus auf die Lab-Ergebnisse warten können, wenn sie wissen, dass am Ende eine robustere Lösung steht.

8. Fazit

Das **primäre Ziel (KI-Agent Prototyp)** und das **sekundäre Ziel (Technologie-Vergleich)** lassen sich am besten erfüllen, indem man **kontrolliert experimentiert (Lab)** und dann **skalierbar implementiert**. Unter den gegebenen Bewertungskriterien erscheint **Ansatz 3** als *Hauptempfehlung*, angereichert durch pragmatische Elemente aus Ansatz 1 und 2 in der finalen Ausgestaltung. Diese Strategie bietet die **höchste Erfolgswahrscheinlichkeit**, um sowohl **kurzfristig Antworten** auf die Projektfragen zu liefern als auch **langfristig die optimale Lösung** für das Unternehmen zu finden.