MORe® unter Belastung

Drei Praxisbeispiele, ihre Grenzen und der Weg von Informationsarchitektur zu operativer Steuerung

Ein Framework wird nicht belastbar, weil es auf Folien funktioniert. Es wird belastbar, wenn es an der Praxis scheitern darf - und daraus Regeln ableitet.

Dieses Whitepaper zeigt nicht drei perfekte Showcases. Es zeigt drei reale Entwicklungsstände: ein bewusst überfrachtetes Gegenbeispiel, einen deterministischen Durchstich von Spezifikation bis Code und Nachweis sowie einen KI-Anwendungsfall mit verteilten führenden Systemen. Genau die Defizite sind Teil der Bewertung.

Autor: Reiner Schindler
Organisation: MORe® Framework / projektkram
Version: v0.6 · September 2026
Praxisbasis: ZOHO ONE · ELSTER USt VA · AI-UC-017 · Jira-Operationalisierung

Abgrenzung: Gezeigt wird der Mechanismus, nicht die vollständige Implementierung. Die Beispiele sind Arbeitsstände aus Confluence und enthalten bewusst auch Unschärfen und Überstrukturierung.

Inhalt

1. Management Summary

MORe adressiert ein reales Problem moderner Projekte: Nicht fehlende Information, sondern verlorener Zusammenhang. Die entscheidende Frage ist nicht, ob Anforderungen, Entscheidungen, Tests und Abnahmen irgendwo dokumentiert sind. Entscheidend ist, ob ihre fachlichen Beziehungen und gültigen Stände noch nachvollziehbar sind.

Die drei Praxisbeispiele verändern die Bewertung von MORe deutlich. Einzelne Bausteine - Use Cases, Traceability, Decision Records, Konfigurationsmanagement, Single Sourcing - sind nicht neu. Eigenständig wird der Ansatz durch ihre Kombination zu einer fachlichen Informationsarchitektur mit vier wiederkehrenden Mechanismen:

- Information wird an einem eindeutig führenden Ort gepflegt - nicht in mehreren Dokumenten parallel.
- Beziehungen zwischen Informationsobjekten tragen fachliche Semantik und sind Teil der Nachweiskette.
- Ableitungen müssen die für ihren Zweck relevanten fachlichen Invarianten erhalten und überprüfbar machen.
- Versionen und Baselines machen verbindlich, welcher Stand zu welchem Zeitpunkt galt.
- Dokumente und andere Publikationsformen werden aus geführten Informationen erzeugt oder referenzieren sie, statt eine zweite Wahrheit zu bilden.

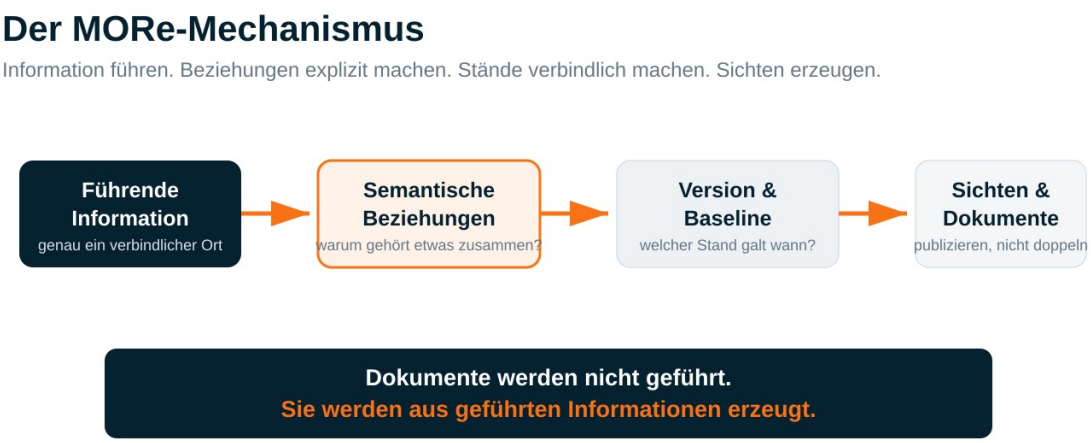


Abbildung 1: Der MORe-Mechanismus. Nicht die Werkzeuge bilden die Klammer, sondern führende Informationen, semantische Beziehungen und verbindliche Stände.

MORe ist kein neues Requirements-Engineering-Verfahren. Es ist ein Ordnungsmodell für Projektwissen.

Die Praxis zeigt zugleich Grenzen. MORe verhindert keine falsche Information, keinen vergessenen Link und keine überladene Vorlage. Es macht diese Probleme jedoch lokalisierbar. Daraus folgt eine nüchterne Einsatzvoraussetzung:

MORe braucht drei Dinge: Disziplin, Disziplin und den Willen, es wirklich zu tun.

2. Was dieses Whitepaper prüfen will

Eine Framework-Beschreibung kann schlüssig klingen und trotzdem in der täglichen Arbeit scheitern. Deshalb werden die Beispiele nicht danach bewertet, ob sie dem Idealbild entsprechen, sondern ob der MORe-Mechanismus im jeweiligen Kontext trägt.

2.1 Die drei Belastungsproben

Beispiel	Charakter	Prüffrage
ZOHO ONE - Meeting	kleine, einfache Anforderung	Verhindert MORe Überdokumentation - oder baut es nur alte Vorlagen in Confluence nach?
ELSTER UStVA	deterministische Fachlogik, XML, XSD, Code, Test	Kann eine führende Spezifikation tatsächlich Umsetzung, Prüfung, Abnahme und Publikation verbinden?
AI-UC-017	KI-System mit verteilten Artefakten und Plattformen	Funktioniert der Ansatz noch, wenn Information bewusst in Confluence, Jira, Git, Data Repository, Model Registry und Monitoring verteilt ist?

2.2 Bewertungsmaßstab

- Ist ein führender Ort für die jeweilige Information erkennbar?
- Werden Informationen verbunden oder kopiert?
- Ist die Beziehung fachlich verständlich oder nur technisch verlinkt?
- Bleiben bei Ableitungen die für ihren Zweck relevanten fachlichen Eigenschaften erhalten?
- Ist ein historischer Stand rekonstruierbar?
- Entsteht im Alltag zusätzlicher Ballast oder reduziert die Struktur Rekonstruktionsarbeit?
- Lässt sich der Ansatz mit bestehenden Werkzeugen operationalisieren?

> Wichtig: Die Beispiele sind keine normativen Templates. Gerade das ZOHO-Beispiel ist als Gegenbeispiel wertvoll, weil es zeigt, wie MORe nicht umgesetzt werden sollte.

3. Der Kern: Informationsgraph statt Dokumentenbaum

Confluence, Jira, Git oder ein ALM-System bringen jeweils eine eigene Ordnung mit. Diese Ordnungen sind nützlich, aber sie sind nicht automatisch die fachliche Ordnung des Projekts. Ein Seitenbaum ist Navigation. Ein Backlog ist Arbeitssteuerung. Ein Repository ist Versions- und Konfigurationsspeicher. Der fachliche Zusammenhang entsteht erst durch die definierten Beziehungen zwischen den Informationsobjekten.

Der Seitenbaum ist eine Sicht. Das Modell lebt in den Beziehungen.

Damit wird auch der Begriff „Single Source of Truth“ präziser. In einer modernen Toollandschaft gibt es selten eine einzige globale Quelle. Sinnvoller ist: Für jede Information gibt es einen führenden Ort. Die Feldspezifikation kann in Confluence geführt werden, der ausführbare Prompt in Git, das Modell in einer Registry und der produktive Datenstand in einem Repository. MORe verlangt nicht, alles in ein Werkzeug zu ziehen. MORe verlangt, dass klar ist, was wo verbindlich ist.

3.1 Fünf Grundfragen

Frage	MORe-Antwort
Wo steht die Information verbindlich?	An einem definierten führenden Ort.
Warum gehören zwei Informationen zusammen?	Durch eine benannte, fachlich lesbare Beziehung.
Was muss bei einer Ableitung erhalten bleiben?	Die für den jeweiligen Zweck relevanten fachlichen Invarianten müssen erhalten und überprüfbar sein.
Welcher Stand galt zu einem bestimmten Zeitpunkt?	Durch Version, Freigabe und Baseline.
Wie entsteht eine Dokumentation?	Als Sicht oder Publikation aus den geführten Informationen.

Diese fünf Fragen reichen als Prüfraster für die folgenden Beispiele. Sie sind bewusst einfacher als ein vollständiges Metamodell. Der Anspruch dieses Whitepapers ist nicht, die gesamte MORe-Konfiguration offenzulegen, sondern den Mechanismus prüfbar zu machen.

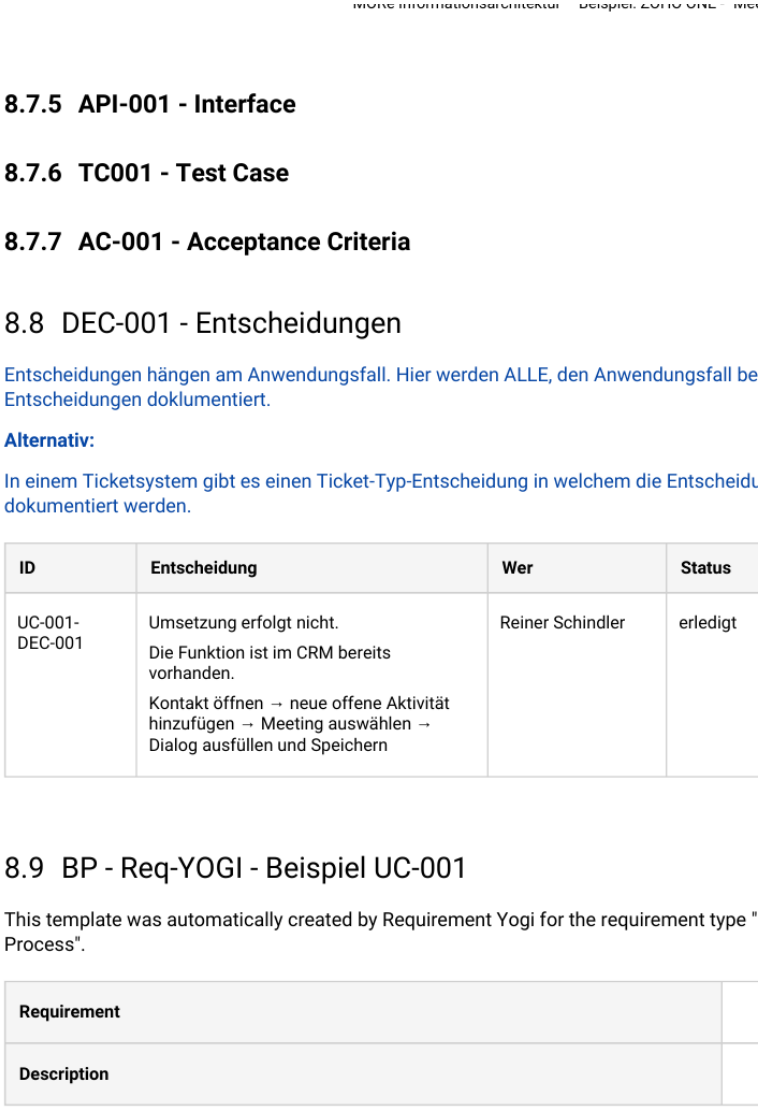
4. Praxisbeispiel 1 - ZOHO: So nicht

Der fachliche Wunsch ist klein: Aus einem CRM-Kontakt soll ein Online-Meeting angelegt werden. Die Analyse zeigt, dass die Standardfunktion bereits vorhanden ist. Die richtige fachliche Entscheidung lautet deshalb: keine Individualentwicklung.

Gerade kleine Anforderungen eignen sich als Belastungsprobe. Große Vorhaben können Überstrukturierung hinter ihrer Komplexität verstecken. Bei einer kleinen Funktion wird sofort sichtbar, ob eine Informationsarchitektur den Sachverhalt klärt oder nur eine Dokumentvorlage reproduziert.

4.1 Was im Beispiel richtig ist

- Der Anwendungsfall bildet die fachliche Klammer.
- Die entscheidende Erkenntnis ist als Entscheidung dokumentiert: Die vorhandene CRM-Funktion reicht aus.
- Die Begründung bleibt am fachlichen Gegenstand erhalten und verschwindet nicht in einem geschlossenen Ticket oder Meetingprotokoll.



8.7.5 API-001 - Interface

8.7.6 TC001 - Test Case

8.7.7 AC-001 - Acceptance Criteria

8.8 DEC-001 - Entscheidungen

Entscheidungen hängen am Anwendungsfall. Hier werden ALLE, den Anwendungsfall be
Entscheidungen doklumentiert.

Alternativ:

In einem Ticketsystem gibt es einen Ticket-Typ-Entscheidung in welchem die Entscheidu
dokumentiert werden.

ID	Entscheidung	Wer	Status
UC-001-DEC-001	Umsetzung erfolgt nicht. Die Funktion ist im CRM bereits vorhanden. Kontakt öffnen → neue offene Aktivität hinzufügen → Meeting auswählen → Dialog ausfüllen und Speichern	Reiner Schindler	erledigt

8.9 BP - Req-YOGI - Beispiel UC-001

This template was automatically created by Requirement Yogi for the requirement type "
Process".

Requirement	
Description	

Abbildung 2: Die fachlich relevante Information des kleinen Beispiels ist im Kern die Entscheidung, keine Individualentwicklung vorzunehmen.

4.2 Was im Beispiel falsch läuft

Der aktuelle Confluence-Stand trägt noch deutlich den Ballast einer klassischen Dokumentvorlage:

Zeichenerklärung, Abkürzungen, Referenzlisten, Prüf- und Abnahmeblöcke, leere Kapitel und vorbereitete Tabellen. Formal ist das ordentlich. Für diesen konkreten Anwendungsfall ist es zu viel.

5 Referenzierte Dokumente

Bezeichnung im Text	Dokument	Version	Verv
[PRINCE2]	*PRINCE2*	??	*Refe*
[P-PLAN]	*Projektplan* *DOK-ID:*		*Refe*
[KM-PLAN]	*Konfigurationsplan* *DOK-ID:*		*Refe*
[QS-PLAN]	*Qualitätssicherungsplan* *DOK-ID:*		*Refe*
[PLD]	*Projektleitdokument* *DOK-ID:*		*Refe*
[AW_FORD]	*Anwenderforderungen* *DOK-ID:*		*Refe*
[PROG-RICHT]	*Programmierrichtlinien* *DOK-ID:*		*Refe*
[PL-RICHT]	*Richtlinie für Projektleiter* *DOK-ID:*		*Refe*

Abbildung 3: Gegenbeispiel - klassische Dokumentlogik wird in Confluence nachgebaut. Für einen kleinen Anwendungsfall entsteht mehr Form als fachlicher Nutzen.

MORe bedeutet nicht: alte Dokumentvorlagen in Confluence nachbauen.

4.3 Die Konsequenz

Für den konkreten Fall genügt eine sehr schlanke Struktur: fachlicher Wunsch, Ergebnis der Analyse, Entscheidung, gegebenenfalls ein Akzeptanzkriterium und ein Nachweis der vorhandenen Standardfunktion. Alles andere ist nur dann sinnvoll, wenn Risiko, Umfang oder regulatorischer Kontext es tatsächlich verlangen.

> Lernpunkt: Tailoring ist kein Komfortmerkmal. Ohne Tailoring kann auch eine informationszentrierte Methode in Dokumentenbürokratie zurückfallen.

5. Praxisbeispiel 2 - ELSTER: Eine führende Spezifikation, mehrere Ableitungen und Nachweise

Das ELSTER-UStVA-Beispiel ist die stärkste Belastungsprobe für das Generierungsprinzip. Aus strukturierten Buchhaltungsdaten wird eine eXML erzeugt, gegen XSD und fachliche Regeln validiert, getestet und für einen konkreten Stand abgenommen.

Hier reicht eine lose Sammlung von Seiten nicht aus. Felddefinitionen, Datentypen, Nachkommastellen, Ausgabebedingungen, Abhängigkeiten, Regeln, XML-Mapping, Validatoren, Testfälle und Abnahme müssen konsistent bleiben. Genau an dieser Stelle ist eine führende Spezifikation sinnvoll.

5.1 SPEC-001 als führender Ort

Die Spezifikation der Felder und Datentypen ist nicht nur Dokumentation. Sie enthält strukturierte Metadaten wie Kennziffer, Datentyp, Nachkommastellen, Pflicht- und Ausgabebedingung, XML-Element, Validierungsregeln, Abhängigkeiten, Quelle, Gültigkeit und Version. Damit wird sie zur fachlichen Quelle für mehrere nachgelagerte Artefakte.

MORe Informationsarchitektur – ELSTER-UStVA XML-Generierun

6 05_SPEC-001 Spezifikation der Felder und und Datentypen für die XML

6.1 Status

Führendes Artefakt / Single Source of Truth

Aus dieser Seite bzw. einer strukturierten Repräsentation dieser Informationen werden te dokumentarische Artefakte abgeleitet.

6.2 Geltungsbereich

Diese Demo enthält bewusst nur einen Ausschnitt der UStVA-Regeln.

6.3 Feldspezifikation

Kennziffer	Fachliche Bezeichnung	Datentyp	Nachkommastellen	Pflichtbedingung	Ausgabebedingung	XML-Element	Validierungsregeln	Abhängigkeiten	Quelle
KZ 84	Bemessungsgrundlage §13b	Numerisch	0	–	Wenn fachlich relevant	Kz84	RULE-003	KZ85 / KZ67	ELS TEF Spe ifika tior
KZ 85	Steuerbetrag §13b	Dezimal	2	–	Wenn fachlich relevant	Kz85	RULE-004	KZ84 / KZ67	ELS TEF Spe ifika tior
KZ 66	Vorsteuerbetrag	Dezimal	2	–	Nicht bei leer / null /	Kz6	RULE-001	–	ELS TEF

Abbildung 4: Ausschnitt aus SPEC-001. Die strukturierte Feldspezifikation ist der führende Informationsbestand für Mapping, Regeln und Ableitungen.

Eine fachliche Regel wird nicht zusätzlich in Feldkatalog, Codekommentar, Testbeschreibung und Handbuch gepflegt.

5.2 Der Mechanismus trägt

- RULE-Objekte beschreiben fachliche Prüf- und Erzeugungslogiken verständlich.
- Der Generatorcode referenziert die fachlichen Regeln und den Spezifikationsstand.
- XSD-Validierung und fachliche Validierung bleiben getrennt - technische Gültigkeit ist notwendig, aber nicht hinreichend.
- Testfälle weisen konkrete Regeln nach.
- Die Abnahme bezieht sich ausdrücklich auf einen identifizierten Versionsstand.
- API-Dokumentation, Feldkatalog, Prüflogiken und Anwenderdokumentation sind Publikationssichten, keine parallel gepflegten Wahrheiten.

5.2.1 Bedeutungserhaltende Ableitungen

Eine Beziehung allein beweist noch nicht, dass eine Ableitung fachlich korrekt ist. Wird ein Spezifikationselement in ein XSD, eine Validatorregel, eine Implementierung oder einen Testfall überführt, müssen die für den jeweiligen Zweck relevanten fachlichen Eigenschaften erhalten bleiben.

Eine Ableitung ist nur dann belastbar, wenn die für ihren Zweck relevanten fachlichen Invarianten erhalten bleiben und dies überprüfbar ist.

MORe bezeichnet diese zu erhaltenden fachlichen Eigenschaften als Invarianten. Welche Invarianten relevant sind, hängt von der Beziehung ab: Eine technische Repräsentation muss andere Eigenschaften erhalten als eine Implementierung oder ein Nachweis.

Führende Information	Ableitung / Nachweis	Relevante Invariante
Datentyp Dezimal	XSD-Element	Numerischer Typ und zulässiges Format bleiben erhalten.
Zwei Nachkommastellen	Validatorregel	Die geforderte Genauigkeit bleibt prüfbar.
Kennzahlen nur gemeinsam zulässig	Fachregel / Testfall	Die Paarabhängigkeit bleibt erhalten und wird nachgewiesen.
Feld bei leer / null nicht ausgeben	Generator / Testfall	Die fachliche Ausgabebedingung bleibt erhalten.

Lernpunkt: Traceability beantwortet damit nicht nur, ob zwei Artefakte verbunden sind. Sie muss bei relevanten Ableitungen auch sichtbar machen, welche fachliche Bedeutung über die Verbindung erhalten bleiben muss.

5.3 Konfigurationsmanagement macht den Stand verbindlich

16 15_CFG-001_Versionierung_und_Konfiguratio
agement

16.1 Warum diese Seite dazugehört

Eine Informationsarchitektur beschreibt nicht nur, welche Informationen zusammengehö
regulierten Umfeld muss zusätzlich nachvollziehbar sein, **welcher Stand zu welchem Zei**

Damit treffen Informationsarchitektur und Konfigurationsmanagement zusammen.

16.2 Konfigurationselemente

- führende Spezifikation
- Regeln
- Mapping
- Generatorcode
- XSD-/Schema-Stand
- Testfälle
- Abnahme
- erzeugte Dokumentation
- fachliche Beziehungen zwischen diesen Objekten

16.3 Baseline

```
Baseline UStVA-2026-07
SPEC: 1.6
RULE: 1.6
Generator: 1.6
Schema: ERiC v43 / Regelstand 56.0.1
Tests: Build 2026.07.18
Abnahme: AC-001 / 2026-07-19
```

16.4 Historische Rekonstruktion

Abbildung 5: Das Beispiel verbindet Informationsarchitektur und Konfigurationsmanagement. Eine Baseline soll beantworten, welcher Spezifikations-, Generator-, Schema-, Test- und Abnahmestand galt.

Das ist ein wichtiger Schritt über reine Traceability hinaus. Nicht nur Artefakte sind relevant, sondern auch ihre gültigen Beziehungen. Wenn RULE-004 durch TC-001 nachgewiesen wird, ist diese Beziehung selbst Teil des nachvollziehbaren Stands.

Informationsarchitektur schafft Zusammenhang. Konfigurationsmanagement macht seinen Stand verbindlich.

5.4 Das Beispiel zeigt auch die eigenen Qualitätsrisiken

Die aktuelle Entscheidungstabelle enthält bei DEC-003 einen Widerspruch zwischen Kurzform und eigentlichem Inhalt: Die Kurzform spricht von Übertragung, Begründung und Auswirkung von Nicht-Übertragung. Genau dieser Fehler ist didaktisch wertvoll. Eine Informationsarchitektur verhindert falsche Inhalte nicht automatisch.

13 12_DEC-001_Entscheidungen

ID	Datum	Entscheidung Kurzform	Entscheider	Betroffener Informationsstand	Alternativen	Begründung	Auswirku
DEC-001	03.08.2026	Anwendung von MORe	Reiner	–	nicht notwendig, Anwendung von MORe	**Spezifikation ist führendes Artefakt.** Feldkatalog, Prüflogiken, technische Mapping-Dokumentation und abgeleitete Dokumente werden nicht parallel gepflegt. Maßgeblich ist die strukturierte Spezifikation.	Abgeleitet werden au Spezifikat und nicht fachlich g
DEC-002	16.08.2026	MVP in VBA	Reiner	–	direkte Implementierung als Plugin hätte länger gedauert und die Abgabe der Meldung verzögert.	**MVP in VBA für LibreOffice Calc.** Das MVP wird bewusst in VBA für LibreOffice Calc implementiert.	Die erste Implemen Codegene erfolgen ir LibreOffic → schnell und Feedb Praxis
DEC-003	17.08.2026	KZ83 muss übertragen werden	Reiner	–	keine	**KZ83 nicht übertragen.** KZ83 wird intern berechnet, aber nicht in die eXML aufgenommen.	KZ83 kann verwendet aber nicht der erzeug erscheine
DEC-004	19.08.2026	KZ67 aus KZ85 ableiten	Reiner	–	gilt nur für die DEMO, fachlich ist das noch zu hinterfragen	**KZ67 aus KZ85 ableiten.** Wenn KZ84 und KZ85 gefüllt sind, gilt in der Demo: `KZ67 = KZ85`.	Generator, und Testfä diese Able berücksicl

Abbildung 6: Auch strukturierte Information kann widersprüchlich sein. Der Vorteil ist: Der Widerspruch ist lokalisierbar und kann durch Qualitätsregeln erkannt werden.

Daraus folgt eine zusätzliche Reifeanforderung an MORe: Die Informationsarchitektur selbst muss prüfbar werden - beispielsweise auf verwaiste Regeln, fehlende Nachweise, widersprüchliche Entscheidungen, ungültige Links oder veraltete Baseline-Bezüge.

5.5 Der Automatisierungsgrad ist begrenzt - durch die Werkzeuge, nicht durch das Prinzip

Das Beispiel belegt heute das Prinzip und einen realen technischen Durchstich. Nicht vollständig automatisiert ist die durchgängige Kette von einer geänderten strukturierten Spezifikation über neu erzeugte Implementierungsparameter und Tests bis zu automatisch neu publizierten Dokumentationssichten.

Ein wesentlicher begrenzender Faktor liegt derzeit in der Werkzeuglandschaft. Gleichzeitig muss MORe präziser beschreiben, welche fachlichen Eigenschaften bei automatisierten Ableitungen erhalten und geprüft werden müssen. Es existiert derzeit kein Produkt, das eine fachlich geführte, versionierte Spezifikation entgegennimmt und daraus sowohl die technischen Repräsentationen als auch die Nachweiskette erzeugt. Wer diese Kette heute schließen will, baut sie selbst. Abschnitt 9.3 ordnet diesen Befund ein.

> Lernpunkt: Das Generierungsprinzip ist belastbar, der Automatisierungsgrad aber noch nicht der Endzustand. Entscheidend ist die Unterscheidung: Ein Teil der Lücke liegt in der Werkzeugunterstützung; ein weiterer Teil liegt in der noch zu präzisierenden Semantik der Transformationen.

6. Praxisbeispiel 3 - KI: Der Zusammenhang liegt nicht in einem Werkzeug

Beim Support-Chatbot reicht die klassische Kette Anforderung - Umsetzung - Test nicht mehr. Datenquellen, Corpus, Vektorindex, Prompt, Guardrails, Modellversion, Evaluation, Akzeptanzschwellen und Monitoring beeinflussen das produktive Verhalten.

Gerade dieses Beispiel prüft die Werkzeugneutralität von MORe. Eine zentrale Ablage wäre hier fachlich falsch: ausführbare Regeln gehören in Git, Datenstände in ein Repository, Modelle in eine Registry, Logs in Monitoring und operative Aufgaben in Jira. Confluence ist der fachliche Einstiegspunkt und die Nachweisklammer - nicht der Speicher für alles.

6.1 Führende Orte statt globaler Single Source of Truth

Information	Führender Ort	Rolle von Confluence
Fachlicher Geschäftsbedarf / Scope / Business Rules	Confluence	führt den fachlichen Zusammenhang
Umsetzungsarbeit	Jira	verlinkt auf fachlichen Kontext
Prompts / Guardrails / ausführbare Konfiguration / Code	Git	verweist auf Datei, Commit, PR und fachliche Begründung
Corpus / Test- und Evaluationsdaten	Data Repository / Eval Repository	führt ID, Version, Zweck, Owner und Link
Modellversion	Model Registry	führt Modell-ID, Version und Freigabestatus
Produktive Beobachtung	Monitoring / Logging	verweist auf Dashboard, Schwellenwerte und Verantwortliche

Confluence beschreibt, welche Daten fachlich verwendet werden dürfen. Das Data Repository führt, welche Daten technisch verwendet wurden.

6.2 Die Baseline wird zum fachlichen Nachweis

MORe Informationsarchitektur – AI-UC-017 — Support-Chatbot fi

4.3 Release-Baseline

Baseline-ID	Release	Datum	Inhalt
AIBL-017-2026.09	Support-Chatbot v1.0	offen	Produktiver Ersts

4.4 Baseline-Inhalt

Artefakt	Version / Stand	Link
Anwendungscode	Git Tag `chatbot-v1.0.0`	[Link
System Prompt	Commit `abc123`	[Link
Guardrails	Commit `abc123`	[Link
Retrieval-Konfiguration	Commit `abc123`	[Link
Wissens-Corpus	Corpus Snapshot `corp-2026-07-28`	[Link
Vektorindex	Index Build `idx-017-2026-07-28`	[Link
Embedding-Modell	`embedding-model-x / version y`	[Link
LLM-Version	`model-x / version y`	[Link
Evaluationsdatensatz	Evalset `eval-017-v1.2`	[Link
Evaluationsergebnis	Eval Run `evalrun-2026-07-28-01`	[Link
Akzeptanzentscheidung	DEC-017-004	[Link

Abbildung 7: Eine AI Configuration Baseline verbindet Anwendungscode, Prompt, Guardrails, Retrieval, Corpus, Vektorindex, Modell, Evalset, Entscheidung und Monitoring.

Der interessante Punkt ist nicht die Liste der technischen Komponenten. Entscheidend ist die fachliche Aussage der Baseline: Welche Zusage wurde mit welcher Daten-, Modell- und Konfigurationsbasis produktiv eingelöst? Damit verschiebt sich Abnahme von einem einzelnen Datum zu einem überprüfbaren Zustand.

6.3 Produktionsnachweis über Systemgrenzen hinweg

MORe Informationsarchitektur – AI-UC-017 — Support-Chatbot fi

9 11. Produktionsnachweis

Nachweisfrage	Führender Ort/System
Welche fachliche Anforderung ist produktiv?	Confluence AI-UC-017
Welche Jira-Vorgänge wurden geliefert?	Jira
Welche Codeversion ist produktiv?	Git Tag / Deployment
Welche Prompt-Version ist produktiv?	Git
Welche Datenbasis wurde verwendet?	Data Repository / Corpus Snapsh
Welche Modellversion wurde verwendet?	Model Registry
Welche Vektorindex-Version war aktiv?	Vector DB / AI-Plattform
Welche Evaluation war Freigabegrundlage?	Eval Repository
Wer hat freigegeben?	Confluence DEC / Abnahme
Wie wird der Betrieb überwacht?	Monitoring-Dashboard
Welche produktive Baseline gilt?	Confluence Release-Baseline

Abbildung 8: Der Produktionsnachweis beantwortet fachliche Fragen über mehrere führende Systeme hinweg, ohne deren Inhalte in Confluence zu duplizieren.

Das Beispiel widerlegt damit die Kritik, Werkzeugunabhängigkeit müsse theoretisch bleiben. Die Information darf verteilt sein. Die Verantwortlichkeit für den führenden Ort und die semantische Verbindung dürfen es nicht sein.

6.4 Die Schwäche bleibt: Pflege der Beziehungen

Die aktuelle Demo enthält bewusst noch Platzhalterlinks. Das macht den zentralen Aufwand sichtbar: Ein Graph wird nicht belastbar, nur weil Links technisch möglich sind. Beziehungen brauchen klare Semantik, Verantwortlichkeit, Versionierung und Qualitätskontrollen. Ohne diese Disziplin wird aus dem Informationsgraphen schnell ein Netz aus „siehe auch“-Links.

> Lernpunkt: Werkzeugneutralität ist ein Vorteil, aber keine Beliebigkeit. Die eingesetzten Werkzeuge müssen Versionierung, nachvollziehbare Beziehungen, Freigaben und historische Rekonstruktion ausreichend unterstützen.

7. Jira: Von der Informationsarchitektur zur täglichen Arbeit

Die bisherigen Beispiele erklären, wo Informationen fachlich geführt werden. Damit ist noch nicht erklärt, wie Teams sie täglich bewegen. Genau hier übernimmt Jira eine andere Rolle: nicht als zweite Spezifikation, sondern als Arbeits- und Reifegradsteuerung.

Ein gemeinsamer Informationsbestand kann unterschiedliche operative Sichten besitzen. Product Owner, Entwicklung, Test und Release brauchen nicht vier Backlogs mit kopierten Inhalten. Sie brauchen vier Sichten auf denselben Bestand und einen nachvollziehbaren Statusfluss.

Abbildung 9: Jira operationalisiert den fachlichen Zusammenhang. Gezeigt wird der Mechanismus, nicht die vollständige Konfiguration.

7.1 Eine schlanke Definition of Ready

Die Definition of Ready ist kein zweiter Sprint und kein Bürokratie-Gate. Sie markiert den Übergang vom fachlich geführten Anforderungswissen zur umsetzbaren Arbeit. Für einen schlanken Einstieg genügen drei Fragen:

- Sind fachliches Ziel und betroffene Informationen verstanden?
- Sind Akzeptanzkriterien und wesentliche Regeln ausreichend klar?
- Sind wesentliche Abhängigkeiten, Risiken und entscheidungsrelevante offene Fragen so geklärt, dass die Umsetzung beginnen kann?

Die Definition of Ready stellt sicher, dass eine Anforderung umsetzbar ist. Die Definition of Done stellt sicher, dass ihre Umsetzung vollständig, nachvollziehbar und nachweisfähig ist.

7.2 Entscheidungen und Risiken als eigene operative Objekte

Zwei Tickettypen sind für die Operationalisierung besonders nützlich: Entscheidung und Risiko. Sie erlauben, fachliche Kausalität in der Arbeitssteuerung sichtbar zu machen, ohne die Fachinformation zu kopieren.

Nicht jede Beziehung bedeutet dasselbe. MORe unterscheidet in der Praxis mindestens Sichten, Ableitungen, Umsetzung, Nachweise, Entscheidungen und Kausalbeziehungen. Die Beziehungsklasse bestimmt, welche fachliche Aussage - und bei Ableitungen gegebenenfalls welche Invarianten - überprüfbar bleiben müssen.

Beziehung	Rückrichtung	Bedeutung
wird entschieden in	Entscheidung betrifft	Die Entscheidung wird als eigenes Objekt geführt und verweist auf alle betroffenen Informationen.
erzeugt Risiko	Risiko resultiert aus	Das Risiko bleibt kausal an der Information verankert, aus der es entsteht.
wird umgesetzt durch	setzt um	Jira-Story oder Task steuert konkrete Arbeit an einer fachlichen Information.
wird nachgewiesen durch	weist nach	Test, Evaluation oder Abnahme belegt eine Regel oder ein Akzeptanzkriterium.

Die vollständige Link-Semantik ist bewusst nicht Gegenstand dieses Whitepapers. Entscheidend ist das Prinzip: Ein Link muss als fachlicher Satz lesbar sein. Generische Beziehungen wie „related to“ oder „siehe auch“ sind für belastbare Traceability zu schwach.

MORe definiert die Semantik der Beziehung. Jira operationalisiert sie.

7.3 Offene Entscheidung statt Open Point: UC-COP-011

Im Use Case UC-COP-011 – Cut-over-Bestand im Board anzeigen ist noch offen, ob die Boardansicht innerhalb der App oder als normales Jira-Board bereitgestellt wird. Klassisch würde dieser Punkt leicht in einem Open-Point-Register landen. MORe behandelt ihn als das, was er fachlich ist: eine noch nicht getroffene Entscheidung.

Eine offene Entscheidung ist keine Notiz. Sie ist ein fehlender Informationsstand.

Informationsobjekt	Beziehung	Entscheidung
UC-COP-011	wird entschieden in	DEC-COP-001 – Bereitstellung der Boardansicht festlegen

7.3.1 Entscheidung als schlankes Jira-Objekt

Neben den Jira-Standardfeldern genügen zwei zusätzliche Felder. Ein separates Feld „Bezug“ ist unnötig: Die betroffenen Informationsobjekte werden über die semantischen Links bestimmt.

Feld	Typ	Bedeutung
Datum der Entscheidung	Datum	Zeitpunkt der getroffenen Entscheidung.
Entschieden von	Freitext	Person, Rolle oder Gremium.

7.3.2 Open bedeutet: nicht Ready

Für DEC-COP-001 stehen zunächst zwei Alternativen im Raum: Boardansicht in der App oder Jira-Standardboard. Solange der Status Open ist, fehlt eine umsetzungsbestimmende Festlegung. Damit ist die Definition of Ready für UC-COP-011 nicht erfüllt.

Status DEC-COP-001	Informationslage	Folge
Open	Entscheidung fehlt.	UC-COP-011 bleibt vor Ready; keine Umsetzung.
Entschieden	Variante, Entscheider und Datum sind dokumentiert.	Entscheidungsblocker ist aufgelöst; Ready ist bei erfüllten übrigen Kriterien möglich.

Der Unterschied zum Open-Point-Register ist nicht die schönere Liste. Die offene Entscheidung wird Teil des fachlichen Zustands des Use Cases – und beeinflusst

unmittelbar seinen Reifegrad.

MORe definiert die Beziehung; Jira führt den Entscheidungsstatus; die Definition of Ready verhindert, dass eine fachlich noch nicht entscheidbare Anforderung in die Umsetzung gezogen wird.

8. Die unbequeme Wahrheit: MORe braucht Disziplin

Eine Informationsarchitektur pflegt sich nicht selbst. MORe kann eine schlechte Arbeitsweise sichtbar machen, aber es kann niemanden zwingen, Informationen an der richtigen Stelle zu führen, Entscheidungen zu begründen oder Links aktuell zu halten.

MORe braucht drei Dinge: Disziplin, Disziplin und den Willen, es wirklich zu tun.

8.1 Was Disziplin konkret bedeutet

- Information wird nicht aus Bequemlichkeit in Ticket, Chat, Mail und Dokument kopiert.
- Eine Entscheidung wird als Entscheidung dokumentiert - mit Begründung, Auswirkung und betroffenem Informationsstand.
- Eine Regel ohne Nachweis bleibt sichtbar unvollständig.
- Ein Link hat eine fachliche Bedeutung und wird bei Änderungen mitgepflegt.
- Ein produktiver Stand wird als Baseline nachvollziehbar gemacht.
- Nicht benötigte Artefakte werden durch Tailoring weggelassen, statt leer mitgeführt zu werden.

8.2 Was MORe nicht leisten kann

MORe verhindert keine falsche fachliche Aussage. Es verhindert auch nicht, dass ein Team einen Link vergisst. Der Nutzen liegt darin, dass die Struktur Qualitätsprüfungen ermöglicht: Waisen, widersprüchliche Entscheidungen, fehlende Nachweise, alte Versionen oder Link-Rot können überhaupt erst als Informationsqualitätsprobleme erkannt werden.

Damit verschiebt sich die Diskussion. Dokumentationsqualität wird weniger zur Frage persönlicher Sorgfalt und mehr zur Frage einer prüfbaren Struktur.

9. Kritische Würdigung nach den drei Beispielen

Die ursprüngliche skeptische Einschätzung - reales Problem, aber im Kern nur bekannte RE-Praktiken - greift nach den Praxisbeispielen zu kurz. Die Einzelbausteine sind weiterhin nicht neu. Die Gesamtkonstruktion ist jedoch eigenständiger, als eine reine Methodenliste vermuten lässt.

Aspekt	Stand	Kritische Einordnung
Problemrelevanz	belegt	Informationsfragmentierung, Drift und Rekonstruktionsarbeit sind in allen drei Fällen plausibel sichtbar.
Praktische Umsetzbarkeit	weitgehend belegt	Confluence, Jira, Git und spezialisierte Systeme lassen sich ohne zentrale Monolith-Lösung verbinden.
Generierungsprinzip	teilweise belegt	ELSTER zeigt den Mechanismus und realen Code; die vollständig automatisierte End-to-End-Generierung ist noch Entwicklungsziel.
Konfigurationsmanagement	stark belegt	ELSTER und AI zeigen Baselines und historische Rekonstruktion als integralen Bestandteil.
Link-Semantik	in Arbeit	Die Notwendigkeit ist klar; ein kleiner verbindlicher Beziehungskatalog muss noch normiert und operationalisiert werden.
Tailoring	in Arbeit	ZOHO zeigt eindrücklich, warum nicht jeder Use Case den vollständigen Artefaktumfang benötigt.
Skalierung	offen	Die Beispiele zeigen Mechanismen, aber noch keinen belastbaren Nachweis in einer Landschaft mit Tausenden Informationsobjekten.
Change / Mindset	kritische Voraussetzung	Ohne disziplinierte Pflege und organisatorischen Willen bleibt auch eine gute Architektur nur Struktur auf dem Papier.

9.1 Was an MORe nicht neu ist

Use Cases, Requirements Information Models, Traceability, Decision Records, Living Documentation, Single Sourcing, Konfigurationsmanagement, MBSE und ALM sind etablierte Ideen. MORe sollte seine Eigenständigkeit deshalb nicht über die Neuheit einzelner Artefakte behaupten.

9.2 Wo die Eigenständigkeit liegt

Use Cases, Traceability, Decision Records und Konfigurationsmanagement finden sich in vielen Ansätzen. Der Unterschied liegt nicht in den Artefakten, sondern in der Behandlung ihrer Beziehungen.

Die meisten Werkzeuge behandeln die Verknüpfung als Attribut: Sie existiert oder sie existiert nicht. Man kann fragen, ob A mit B verbunden ist. Man kann nicht fragen, ob A zu einem bestimmten Zeitpunkt mit B verbunden war und wodurch.

In MORe ist die Verknüpfung selbst ein Artefakt.

Sie hat eine Bedeutung, einen Verantwortlichen, eine Version und einen Gültigkeitszeitraum, und sie ist Bestandteil einer Baseline.

Die Semantik einer Beziehung beschränkt sich dabei nicht auf ihren Namen. Bei Ableitungs-, Implementierungs- und Nachweisbeziehungen muss zusätzlich bestimmbar sein, welche fachlichen Eigenschaften beim Übergang erhalten bleiben müssen. Eine bloße Verbindung zwischen zwei Artefakten genügt nicht, wenn ihre fachliche Bedeutung unterwegs verändert wurde.

Traceability ohne semantische Integrität kann einen Zusammenhang nachweisen, der fachlich bereits gedriftet ist.

Damit verschiebt sich, was nachweisbar ist. Nicht nur, dass eine Anforderung heute mit einem Testfall verbunden ist - sondern welche Beziehung zum Zeitpunkt der Abnahme galt, wodurch sie begründet war und wann sie sich geändert hat. Eine gelöschte Verknüpfung ist dann kein spurloser Vorgang, sondern eine dokumentierte Änderung am Nachweisstand.

Das ist der Anspruch. Der heute erreichbare Stand bleibt dahinter zurück, weil kein verfügbares Werkzeug Beziehungen als eigenständige, versionierte Objekte führt. Die Abschnitte 9.3 und 9.4 beschreiben, wie weit man damit kommt und was fehlt.

Information einmal führen. Beziehungen explizit machen. Stände verbindlich machen. Sichten daraus erzeugen.

9.3 Die Werkzeuglücke

Eine naheliegende Rückfrage lautet, warum dieser Mechanismus nicht einfach von einem Werkzeug übernommen wird. Die Antwort ist nicht, dass es keine Werkzeuge gäbe. Es gibt sie - auf beiden Seiten der Grenze. **Keines überbrückt sie**.

- ALM- und Requirements-Management-Systeme wie Polarion, Codebeamer oder Jama Connect verwalten Anforderungen, Beziehungen, Baselines, Freigaben und Testnachweise und decken regulierte Umfelder mit eigenen Vorlagen ab. Sie verknüpfen und verwalten. Sie erzeugen aus einer fachlichen Spezifikation keine technischen Repräsentationen.
- Das spezifikationsgetriebene API-Ökosystem löst die Erzeugung tatsächlich. Aus einer Definition entstehen Dokumentation, Clients, Mocks, Vertragstests und Prüfregeln; Werkzeuge für Linting und Breaking-Change-Analyse sichern die Kette im Build ab. Diese Definition ist jedoch bereits der technische Kontrakt. Die fachliche Ebene davor - Bedeutung, Entscheidung, Abnahme, Nachweis - bleibt außerhalb des Werkzeugs [C1][C2].
- MBSE und SysML behandeln das Modell als maßgebliches Artefakt und erzeugen Sichten daraus. In realen Landschaften mit CAD-, Simulations-, PLM- und Codemodellen wird der Begriff der einen führenden Quelle jedoch unscharf; der Schwerpunkt liegt zudem im Systems Engineering und nicht in der Business-IT [C3][C4].

Auf beiden Seiten der Grenze existieren ausgereifte Werkzeuge. Über die Grenze hinweg fehlen sowohl durchgängige Werkzeugunterstützung als auch eine explizite Beschreibung, welche fachlichen Invarianten eine Transformation erhalten muss.

Bemerkenswert ist, dass die Diagnose außerhalb des Requirements Engineering dieselbe ist. Eine aktuelle Analyse zur Schema-Drift in API-Werkzeugketten führt das Problem auf mehrere unabhängig gepflegte Kopien desselben Kontrakts zurück und empfiehlt, die Artefakte zu erzeugen statt sie zu pflegen [C1]. Das ist derselbe Satz wie in Kapitel 5, eine Ebene tiefer und auf einen technischen Kontrakt bezogen.

Dieselbe Analyse benennt auch die Kehrseite: Wird die Spezifikation umgekehrt aus dem Code erzeugt, wird sie zum Nebenprodukt statt zu einem prüfbaren Kontrakt, und ein Schritt, an dem Verträglichkeit bewertet werden könnte, entfällt [C1]. Genau diese Grenze zieht MORe bewusst vor dem Anwendungscode.

Der Ansatz ist konsequent - er beginnt aber beim Kontrakt. Fachliche Regeln, die sich in einem Schema nicht ausdrücken lassen, fallen aus der erzeugten Kette heraus. Die Regel, dass zwei Kennzahlen nur gemeinsam angegeben werden dürfen, steht im ELSTER-Fall in der Jahresdokumentation und nicht im XSD; keine schemagetriebene Pipeline erfasst sie. Ebenso wenig erfasst wird, auf welcher Entscheidung eine Festlegung beruht und ob deren Bedingung noch gilt.

Formatdrift und Definitionsdrift werden damit beherrschbar. Regeldrift und Nachweisdrift bleiben offen.

Lernpunkt: Der begrenzte Automatisierungsgrad des ELSTER-Beispiels markiert zwei Lücken: fehlende Werkzeugunterstützung über die fachlich-technische Grenze hinweg und eine noch zu präzisierende

Transformationssemantik. Solange beides nicht geschlossen ist, müssen führende Spezifikation, Ableitungsregeln und Nachweise organisatorisch geführt werden.

9.4 Die zweite Lücke: die Verbindung ist kein Artefakt

Die erste Lücke betrifft die Erzeugung. Die zweite betrifft den Anspruch aus Abschnitt 9.2 und lässt sich an einem Werkzeug zeigen, das auf der Artefaktebene bereits sehr weit ist.

Was auf der Artefaktebene bereits gelöst ist

Requirement Yogi ist im Confluence-Umfeld der Referenzfall und in mehreren MORe-Umsetzungen im Einsatz. Anforderungen sind dort adressierbare Objekte mit Schlüssel, Typ und Eigenschaften. Vorlagen und Anforderungstypen sind konfigurierbar, Regeln sind prüfbar, und eine Anforderung erreicht ihren Freigabezustand nicht, solange die geforderten Angaben fehlen.

Baselines frieren Anforderungen samt Text ein; Abhängigkeiten innerhalb einer Baseline werden mitgespeichert.

Dieser Formalismus ist keine Einschränkung, sondern genau das, was eine belastbare Artefaktebene ausmacht. MORe setzt ihn voraus und muss ihn nicht neu erfinden.

Wo auch dieser Ansatz aufhört

Die Verbindung ist dort keine eigenständige Information, sondern eine Eigenschaft der verbundenen Objekte. Sie wird konserviert, weil das Objekt konserviert wird - nicht, weil sie selbst geführt würde.

- Sie hat keine eigene Identität und damit keinen Schlüssel, keinen Verantwortlichen und keinen Gültigkeitszeitraum.
- Sie trägt keine Begründung. Sichtbar ist, dass A und B verbunden sind, nicht warum und seit wann.
- Sie hat keinen Lebenszyklus. Eine Anforderung kann nicht freigegeben werden, solange Angaben fehlen. Eine Beziehung kann überhaupt nicht freigegeben werden, weil sie keinen Zustand besitzt.
- Eine nicht gezogene Beziehung hinterlässt keine Lücke, weil das Objekt fehlt, das fehlen könnte.

Der Nachweis liegt nicht in den Artefakten, sondern zwischen ihnen. Ist die Verbindung kein Artefakt, bleiben die Gelenke der Nachweiskette ungeführt.

ALM-Systeme kommen von der anderen Seite und bleiben ebenfalls davor stehen. Sie markieren über Suspect-Mechanismen, dass sich ein Ende einer Beziehung geändert hat. Sie versionieren jedoch nicht die Beziehung selbst.

Die Kette ist gebaut - die Gelenke sind es nicht

Der Werkzeugpfad existiert bereits vollständig: von der Anforderung in Confluence über das Ticket in Jira bis zu Commit, Branch und Testlauf in Git und Zephyr. Jedes Werkzeug versioniert seinen eigenen Gegenstand sauber. Keines versioniert die Verbindung eines Gliedes zum nächsten.

Damit ist die Lücke klein, aber sie sitzt an jeder Übergangsstelle und ist überall dieselbe. Man kann jedes einzelne Artefakt in seinem Stand belegen und trotzdem nicht belegen, dass sie zu einem bestimmten Zeitpunkt in dieser Konstellation zusammengehörten. Genau diese Frage stellt eine Prüfung.

Ein Werkzeugwechsel repariert das nicht. Er verschiebt die Bruchstelle nur, weil das neue Werkzeug seinen Gegenstand ebenso gut und das Glied ebenso wenig führt. Jede zusätzliche Werkzeuggrenze ist ein weiteres ungeführtes Gelenk.

Werkzeugneutralität heißt nicht Werkzeugbeliebigkeit.

Daraus folgt keine zentrale Ablage - Abschnitt 6.1 zeigt, warum das bei KI-Systemen fachlich falsch wäre. Es folgt Sparsamkeit: so wenige führende Orte wie fachlich nötig, jeder mit ausdrücklicher Begründung. Ein Werkzeug, auf das verzichtet werden kann, nimmt ein ungeführtes Gelenk mit sich.

Dasselbe Prinzip, anderes Artefakt

Der Entscheidungstickettyp aus Abschnitt 7.2 folgt derselben Logik. Eine Entscheidung im Kommentar ist kein Objekt: Sie ist nicht auswertbar, nicht verknüpfbar, nicht abnehmbar und nicht als fehlend feststellbar. Erst als eigener Vorgangstyp bekommt sie Identität, Status und Beziehungen, Version.

Für die Verbindung gilt genau dasselbe - nur fehlt dafür bisher das Werkzeug.

Was heute möglich ist

Confluence versioniert Seiten. Da eine Beziehung auf einer Seite steht, ist sie mittelbar mitversioniert: Es lässt sich rekonstruieren, dass eine Seite in Version 7 auf einen Testfall verwies und in Version 8 nicht mehr. Das ist der praktikable Kompromiss - unter der Bedingung, dass die fachliche Bedeutung einer Beziehung ausdrücklich benannt und nicht als generischer Verweis geführt wird.

Der Schritt darüber hinaus wäre die Verbindung als eigenes Objekt mit Schlüssel, Typ, Verantwortlichkeit, Version und Baseline-Zugehörigkeit - aufsetzend auf einer vorhandenen Artefaktebene, nicht als deren Ersatz. Ein solches Produkt existiert nicht.

> Lernpunkt: Beide Lücken haben denselben Charakter. Sie sind keine Reifeschwäche des Ansatzes, sondern fehlende Werkzeugunterstützung. Solange sie bestehen, sind führende Spezifikation und Beziehungsführung organisatorische Leistungen und können nicht an ein Produkt delegiert werden.

10. Was MORe als Nächstes braucht

10.1 Semantik und operative Steuerung zusammenführen

Der Linkkatalog und die Jira-Operationalisierung gehören zusammen. Die fachliche Beziehung darf nicht nur im Whitepaper definiert sein; sie muss in der täglichen Arbeitssteuerung sichtbar, prüfbar und soweit sinnvoll technisch unterstützt werden.

10.2 Semantische Integrität von Ableitungen

Die nächste methodische Reifestufe liegt nicht nur in einem normierten Beziehungskatalog. Für Ableitungs-, Implementierungs- und Nachweisbeziehungen muss beschrieben werden, welche fachlichen Invarianten erhalten bleiben müssen und wie dies überprüft werden kann.

Damit wird aus Traceability eine stärkere Aussage: nicht nur „A hängt mit B zusammen“, sondern „B bildet die für diesen Zweck relevanten Eigenschaften von A nachweisbar korrekt ab“. Eine formale Methodik für operationale Äquivalenz kann darauf aufsetzen, ist aber nicht Voraussetzung für die praktische Anwendung des MORe-Kerns.

10.3 Tailoring als Bestandteil des Frameworks

Das ZOHO-Gegenbeispiel zeigt, dass „vollständig“ nicht automatisch „gut“ ist. MORe braucht einfache Profile oder Regeln, welche Informationsobjekte bei kleinem, normalem, reguliertem oder KI-geprägtem Kontext tatsächlich benötigt werden.

Profil	Beispielhafter Umfang
Light	Anwendungsfall, Akzeptanz, Entscheidung
Standard	zusätzlich Prozess/Modell, Schnittstelle, Test
Controlled	zusätzlich Versionen, Baseline, formale Nachweisbeziehungen
AI / reguliert	zusätzlich Daten, Modell, Evaluation, Monitoring, Produktionsnachweis

10.4 Die Informationsarchitektur selbst prüfbar machen

Eine nächste Reifestufe sind Qualitätsregeln über die Struktur: Welche Regel hat keinen Test? Welche Entscheidung widerspricht ihrer Kurzform? Welche produktive Baseline referenziert eine alte Spezifikation? Welche Beziehung zeigt auf ein nicht mehr existentes Objekt? Solche Prüfungen machen Informationsqualität messbar.

10.5 Den ELSTER-Durchstich schließen

Das überzeugendste technische Proof-of-Concept wäre eine sichtbare Änderung an SPEC-001, aus der Generatorparameter, Validator, Testfall und mehrere Dokumentationssichten nachvollziehbar neu entstehen. Nicht weil alles zwingend generiert werden muss, sondern weil damit der zentrale MORe-Satz technisch überprüfbar wird.

Nach dem Befund aus Abschnitt 9.3 ist das ein Eigenbau. Die vorhandenen Werkzeuge decken Teilstrecken ab: Erzeugung unterhalb des technischen Kontrakts, Traceability und Baselines oberhalb davon. Der Übergang zwischen beiden ist die eigentliche Arbeit - und zugleich der Punkt, an dem sich zeigen wird, ob der Mechanismus über das Prinzip hinaus trägt.

11. Fazit

Die drei Beispiele zeigen keine perfekte MORe-Welt. Genau deshalb sind sie aussagekräftig. Das ZOHO-Beispiel zeigt, wie schnell eine gute Idee wieder in Dokumentenbürokratie kippen kann. ELSTER zeigt, dass eine führende Spezifikation Regeln, Code, Test, Abnahme und Publikation tatsächlich zusammenhalten kann. Der KI-Chatbot zeigt, dass Informationsarchitektur nicht bedeutet, alles in ein Werkzeug zu ziehen, sondern führende Orte über einen fachlichen Zusammenhang und eine Baseline zu verbinden.

Damit ist MORe weder ein Ersatz für IREB, arc42, Model Cards, Jira, Confluence noch Konfigurationsmanagement. Es verbindet deren Artefakte und führende Orte über semantische Beziehungen und nachvollziehbare Stände. Traceability entsteht nicht am Ende durch Rekonstruktion, sondern während der Arbeit.

Mit der semantischen Integrität kommt eine weitere Anforderung hinzu: Traceability reicht nicht, wenn eine Ableitung zwar verknüpft, aber fachlich gedriftet ist. Für relevante Transformationen müssen deshalb die fachlichen Invarianten benannt und überprüfbar erhalten werden.

Der Zusammenhang entsteht während der Arbeit - nicht hinterher durch Rekonstruktion.

Was MORe jetzt noch beweisen muss, ist Skalierung: viele Informationsobjekte, viele Teams, reale Toolgrenzen und längere Laufzeiten. Die Richtung ist nach den drei Praxisbeispielen klarer als zuvor. Die nächste Reifestufe liegt nicht in mehr Dokumentation, sondern in weniger, besser geführter Information und einer operativen Steuerung, die diese Struktur respektiert.

www.more-framework.org · mail@more-framework.org

Anhang A. Grundlage der Praxisbeispiele

Die in diesem Whitepaper verwendeten Screenshots und Befunde stammen aus drei MORe-Demonstratoren in Confluence, Stand 21.08.2026:

- Beispiel ZOHO ONE - „Meeting“-Schaltfläche, Confluence-Export, Version 9.
- ELSTER-UStVA XML-Generierung für ZOHO Books, MORe Informationsarchitektur.
- AI-UC-017 - Support-Chatbot für Serviceanfragen, MORe Informationsarchitektur.

Zusätzlich wird eine neutralisierte Jira-Workflow-Referenz verwendet, um den Übergang von fachlicher Informationsarchitektur zu operativer Steuerung zu illustrieren. Die veröffentlichte Darstellung zeigt bewusst nicht die vollständige technische Konfiguration.

Anhang B. Einordnung zu bestehenden Ansätzen

MORe baut auf etablierten Denkfiguren auf. Für die fachliche Einordnung sind insbesondere Requirements Engineering und Traceability (IREB / ISO/IEC/IEEE 29148), Living Documentation, Single Sourcing, Architecture Knowledge Management, ALM/Requirements-Management-Systeme sowie Konfigurationsmanagement relevant. Bei KI-Systemen kommen Daten-Governance, Modell- und Evaluationsstände sowie Produktionsbeobachtung hinzu.

MORe ersetzt diese Ansätze nicht. Der Anspruch liegt in der verbindenden Informationsarchitektur: führende Orte, semantische Beziehungen und nachvollziehbare Baselines über bestehende Artefakte und Werkzeuge hinweg.

Anhang C. Quellen zur Werkzeuglandschaft

Die Aussagen in Abschnitt 9.3 beruhen auf einer Sichtung der öffentlich verfügbaren Darstellungen der jeweiligen Anbieter und Fachbeiträge, Stand September 2026. Sie ersetzen keine Werkzeugevaluation für ein konkretes Projekt.

[C1] Fern: Stopping schema drift - a guide to coupling SDKs with documentation. buildwithfern.com, August 2026.

[C2] Zuplo: Best OpenAPI Tools and Documentation Platforms in 2026. zuplo.com, 2026.

[C3] Jama Software: What Is MBSE? Model-Based Systems Engineering Explained. jamasoftware.com, 2026.

[C4] Bajaj, M. et al.: MBSE++ - Foundations for Extended Model-Based Systems Engineering. INCOSE International Symposium, 2016.

[C5] Siemens: Polarion - Requirements und Testmanagement. Produktdarstellung, siemens.com, 2026.

Weiterführend: www.more-framework.org