

Lösch & Partner GmbH hat mehr als 25 Jahren Erfahrung und Praxiswissen in das KIZAM Forschungsprojekt eingebracht. Das Hauptaugenmerk zu Beginn lag in der Koordination und im Alignment der verschiedenen Projektparteien. Ein gemeinsames Verständnis, speziell in der Industrialisierung des Forschungsbeitrags, musste kreiert werden. Dazu wurde im Q4 2021 ein gemeinsamer Workshop gehalten, in dem der Anforderungsprozess erklärt wurde - mit dem gemeinsamen Ziel, den Prozess zu vereinheitlichen und zu quantifizieren. Das sollte als Basis dienen, um KI-Potentiale im Entwicklungsprozess zu identifizieren und entsprechend den gesamthaften Business Value besser abzuschätzen.

Lebenszyklus einer Anforderung

Der Lebenszyklus einer Anforderung wurde wie folgt im Detail dargestellt. Die Quantifizierung und das Zahlenmaterial war exemplarisch zu verstehen und hatte noch nicht den Anspruch auf Richtigkeit. Es wurde von einer Grundmenge von 100 Anforderungen ausgegangen, die beispielhaft in 30 einfach, 50 mittel und 20 schwer geartet waren.



Abb. 1: Schritt 1: Anforderungserstellung

Die Anforderungserstellung steht am Beginn des Lebenszyklus einer Anforderung. Hier recherchiert der Anforderungssteller die Grundgegebenheiten, formuliert und prüft diese. Daraufhin wird die Anforderung fachbereichsintern abgestimmt und anhand der Relevanz und der verbundenen Kosten im Produktionsprozess priorisiert. Die Vervollständigung der Anforderung schließt diesen Prozessschritt ab und stellt die Grundlage für die weiteren Schritte her.

1.2 Neu-Formulierung und Prüfung (fehlt: Beschreibung des Verständnisses des Prozessschrittes)							
Aufwände							
Anzahl Anforderungen	Art	Anteil	Aufwand	Kosten extern	Kosten intern	Gesamtkosten extern	Gesamtkosten intern
30	Einfach	100%	0,25 h	70,00 €/h	150,00 €/h	525,00 €	1.125,00 €
50	Mittel	100%	0,75 h	70,00 €/h	150,00 €/h	2.625,00 €	5.625,00 €
20	Schwer	100%	1,5 h	70,00 €/h	150,00 €/h	2.100,00 €	4.500,00 €
Summe						5.250,00 €	11.250,00 €

Abb. 2: Kalkulation bei der Neu-Formulierung und Prüfung einer Anforderung

Auch der Kostenaspekt wurde in Betracht gezogen. Der zentrale Punkt war, wie viel Arbeit kann die KI übernehmen und was muss noch händisch gemacht werden.

Zudem wurde noch zwischen internen und externen Kosten unterschieden.

Besonders im Zusammenspiel von OEMs, Zulieferern und dem Dienstleistungssektor macht diese Unterscheidung Sinn, um hier das Einsparpotenzial von KI genauer schätzen zu können. Im Vergleich sind die externen Kosten für die Bearbeitung 100 Anforderungen in Schritt 1.2 bei 5.250,00€ und die internen bei 11.250,00€. Unter Berücksichtigung der Aufteilung von KI-Arbeit und verbleibender händischer Arbeit ergibt sich hier ein hohes Einsparpotenzial.

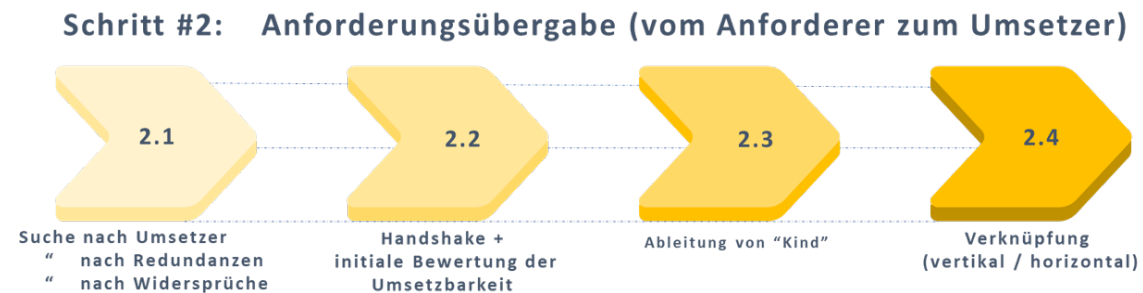


Abb. 3: Schritt 2: Anforderungsübergabe

Die Anforderungsübergabe stellt die Verbindung des Anfordernden zum Umsetzenden her. Nach gemeinsamer Prüfung geht es an den Handshake und die Bewertung der Umsetzbarkeit. Die Anforderung wird in weitere Kind-Anforderungen abgeleitet und vertikal und horizontal verknüpft.

Hier stellt sich alleine die Suche nach dem Umsetzenden als komplex dar und benötigt möglicherweise viele Anläufe. Hinzu kommt die Prüfung potenzieller Redundanzen und Widersprüche in den gestellten Anforderungen. Das KI-Potenzial wurde auch hier hoch eingeschätzt, um den zeitlichen Aufwand grundsätzlich zu verringern und die richtigen Ansprechpartner schneller zu identifizieren. Am Beispiel der 100 Anforderungen wurden die externen Kosten auf 9.940,00€ und die internen auf 21.300,00€ geschätzt.

2. Anforderungsübergabe (vom Anforderer zum Umsetzer)							
2.1 Suche nach Umsetzer							
Aufwände							
Anzahl Anforderungen	Art	Anteil	Aufwand	Kosten extern	Kosten intern	Gesamtkosten extern	Gesamtkosten intern
30	komplex	100%	3,0 h	70,00 €/h	150,00 €/h	6.300,00 €	13.500,00 €
50	mittel	100%	1,0 h	70,00 €/h	150,00 €/h	3.500,00 €	7.500,00 €
20	einfach	100%	0,1 h	70,00 €/h	150,00 €/h	140,00 €	300,00 €
Summe						9.940,00 €	21.300,00 €

Abb. 4: Kalkulation bei der Suche nach Umsetzer

Der nächste Schritt im Entwicklungs- und Anforderungsprozess ist das Änderungsmanagement.

Schritt #3: Änderungsmanagement

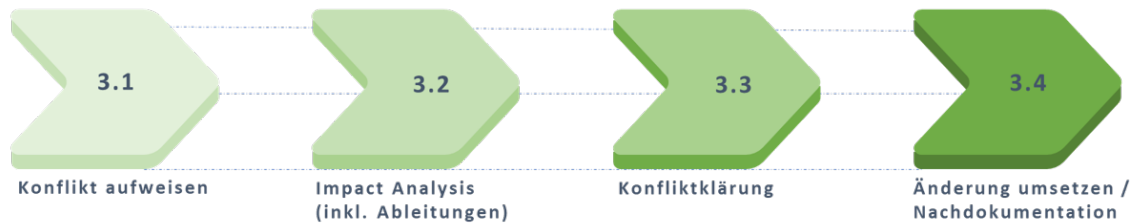


Abb. 5: Schritt 3: Änderungsmanagement

Das Änderungsmanagement beinhaltet nach aktuellen Einschätzungen große Arbeitsanteile, die noch nicht bzw. nicht ausschließlich durch KI durchgeführt werden können. Die Konfliktklärung und das Umsetzen und Nachdokumentieren von Änderungen bedarf viel händischer Arbeit und Diskussionen. Allerdings ergab die Prozessdarstellung die Erkenntnis, dass das Aufweisen von Konflikten – sowie die die Impact Analyse – durch KI-Unterstützung erleichtert werden könnte.

Der letzte Schritt im Anforderungszyklus ist das Lastenheftmanagement.

Schritt #4: Lastenheftmanagement

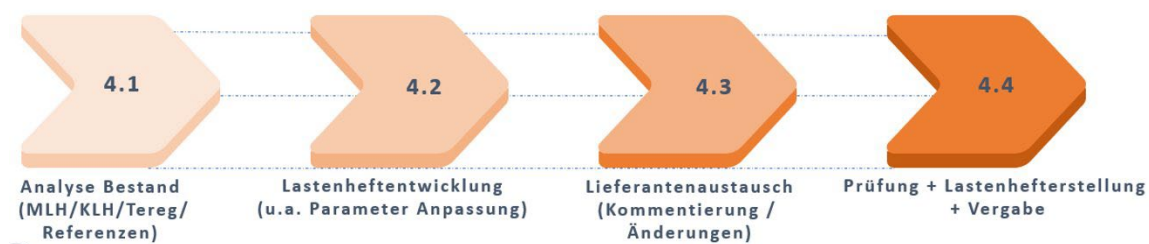


Abb. 6: Schritt 4: Lastenheftmanagement

Hier werden der Bestand analysiert und die Referenzen gezogen, bevor die Lastenheftentwicklung startet. Hier stellte es sich schwieriger dar, das KI-Potential vor allem im quantitativen noch genauer zu analysieren. Nach der Entwicklung steht der Austausch mit potenziellen Lieferanten an, bei denen noch weitere Kommentare und Änderungen ins Lastenheft eingepflegt werden müssen. Als letzter Prozessschritt verstehen

sich die Prüfung, Erstellung und Vergabe des Lastenhefts.

Die Ergebnisse der durchgeführten Analyse ließen sich in das AP1 1.1.1 Ist-Analyse & Potenzialabschätzung – Prozess einordnen. Lösch & Partner führte eigene Regelmeetings durch und nahm an den projektübergreifenden Meetings teil, um in verschiedenen APs noch weitere Informationen aus der Praxis bereitzustellen und um das bestmögliche Projektergebnis zu erzielen.

Zusammenarbeit mit der RWTH Aachen

Das erste Halbjahr 2022 stand ganz im Zeichen der Weiterentwicklung des Anforderungsprozesses. Nach dem konsortialübergreifenden Workshop im Dezember 2021 entwickelte sich ein reger Austausch mit dem WZL der RWTH Aachen. Hier wurde sowohl seitens Lösch & Partner als auch der RWTH eine wissenschaftliche Recherche betrieben, um fundierte Erkenntnisse des Anforderungsprozesses zusammenzutragen.

Die praktischen Erfahrungen von Lösch & Partner galt es wissenschaftlich zu stützen und verschiedene Theorien miteinander zu verknüpfen. Die größten Lücken gab es bei der Verbindung des V-Modells und des Anforderungsprozesses. Kein aus der Praxis abgeleiteter Prozessschritt fand sich auch in der Theorie wieder. Diese Feststellung bildete die Grundlage der Zusammenarbeit im Forschungsprojekt.

Eine der weitreichendsten Erkenntnisse hieraus war, dass ein Lebenszyklus gleichzeitig sowohl bedarfsorientiert als auch linear abläuft. Hieraus ergeben sich dann Pflichtaufgaben, die bedarfsgerecht mit optionalen Tätigkeiten erweitert werden. Die Aufgabe bestand darin, ein gemeinsames Verständnis zwischen Praxis, Theorie und den Werkzeugen des Natural Language Processing zu schaffen.

Anforderungslifecycle und V-Modell

Im zweiten Quartal von 2022 wurde der Anforderungslifecycle beschrieben.

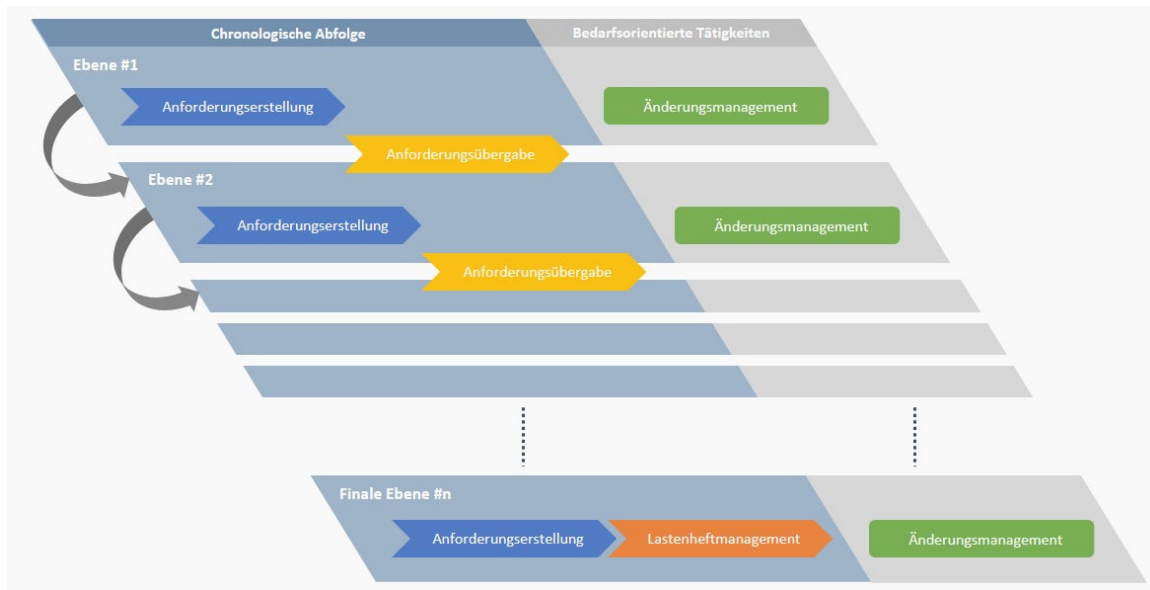


Abb. 7: Phasen des Anforderungsmanagement entlang des V-Modells

Die Grafik veranschaulicht die Zusammenführung der Phasen des Anforderungsmanagement (AM) Prozesses aus der Praxiserfahrung von Lösch & Partner mit dem V-Modell.

Viele theoretische Modelle des AM Prozesses betrachten eine stark vereinfachte oder abstrahierte Produktstruktur. Um jedoch den Übergabeprozess und mögliche Konflikte zwischen Ebenen darstellen zu können, wurde das Ebenen-Modell mit der linken Seite des V-Modells verbunden. In einem Anforderungs-Lifecycle gibt es N Ebenen und AM sieht auf jeder Ebene unterschiedlich aus. Zum Beispiel kann unterschieden werden zwischen HW, Embedded und SW oder auch zwischen High-level Kundenanforderungen und Funktions- und Eigenschaftsanforderungen. In jeder Ebene gibt es eine Abfolge von chronologischen Tätigkeiten sowie mögliche bedarfsorientierte Tätigkeiten. Zu der chronologischen Abfolge gehören feste Bestandteile des AM Prozesses. Je mehr Ebenen es gibt, desto mehr Aufwand wird erwartet. Im Vordergrund der Analyse stand die Quantifizierung der Teilschritte und die Erkennung repetitiver Aufgaben.

Der erste Entwurf des Anforderungslifecycles zeigte vier übergeordnete Tätigkeiten im AM, welche überwiegend auf Theorien in der Literatur über das AM beruhten. Nach der ersten Skizzierung des Anforderungsprozesses wurde der Lifecycle weiter optimiert. Die Ebenen-Struktur sowie die Unterscheidung zwischen verpflichtenden und optionalen Tätigkeiten wurden beibehalten.

Die verbesserten Tätigkeiten im Lifecycle basieren auf dem Generischen Modell für Prozessschritte im AM, die von der RWTH erarbeitet wurde. Anforderungsmanager von LuP

analysierten dieses Modell und schlugen aus eigener Praxiserfahrung Änderungen in den Tätigkeiten vor. Änderungen des Modellentwurfs der RWTH umfassten Umbenennungen, Ausdetaillierungen und Hinzufügen von Tätigkeiten im AM. Außerdem wurden Tätigkeiten rausgenommen, die vom Tooling schon übernommen werden und daher nicht state-of-the-art sind.

Die Anforderungsentstehung und -detaillierung umfasst Tätigkeiten, die in der Literatur oft unter Anforderungserstellung aufgeführt werden. Die Anforderungsübergabe und -änderung wurden von einer Kategorie im Modell der RWTH abgeleitet. Die Lastenhefterstellung wurde als eigene Kategorie im Anforderungslifecycle aufgeführt, da diese erst in der letzten Ebene stattfindet.

Zudem wurde die von der RWTH aufgeführte Anforderungsumsetzung aus dem Lifecycle rausgenommen, da dies out-of-scope von KIZAM ist. Aufbauend aus unserem ersten Entwurf des Anforderungslifecycles wurden die Tätigkeiten modifiziert und mit detaillierten Beschreibungen erweitert. Der optimierte Anforderungslifecycle wurde den Kollegen der RWTH vorgestellt. Die RWTH-Kollegen stimmten den Änderungen zu und sahen das neue Modell ebenfalls als eine verbesserte Darstellung der Prozessschritte im Anforderungsmanagement. Das neue Modell wurde beim Konsortialtreffen im Oktober 2022 vorgestellt und regte eine Diskussion über Prozesse im Anforderungslifecycle an.

Fragebogen RWTH

Des Weiteren entwickelte die RWTH einen Fragebogen zur Prozessanalyse. Die Beantwortung dieser Fragen sollte die im weiteren Verlauf von KIZAM zu bearbeitenden Prozessschritte identifizieren. Der Fragebogen umfasste 11 Fragen, wobei zwei von diesen Fragen mehrere Folgefragen beinhalteten. Beispielsweise beinhaltete der Bogen Fragen über Aufwände, Fehleranfälligkeit und Input/Output-Informationen der Prozessschritte. Dieser Fragebogen wurde im September 2022 an die Usecase-Owner geschickt mit dem Zweck, dass sie Prozessschritt-ID zu den Fragen zuordnen können. Im Oktober 2022 präsentierten die RWTH-Kollegen den Fragebogen bei dem Konsortialtreffen. Hier viel auf, dass die vorgestellten Fragen zu unspezifisch waren. Nach einem ersten Termin Anfang November mit Lösch & Partner und RWTH wurde die Bearbeitung des Bogens kurzzeitig gestoppt. Der Fragebogen wurde von Lösch&Partner analysiert und auf Vollständigkeit, Richtigkeit und Anwendbarkeit überprüft. Fragen bezüglich Aufwands, Kosten und Informationsmengen wurden als zu unspezifisch eingeordnet. Für diese zusammen mit einigen anderen ungenauen Fragen wurden Gegenvorschläge entwickelt, welche bei einem Meeting mit den RWTH-Kollegen nochmal erläutert wurden. In enger Zusammenarbeit verbesserten die Kollegen von Lösch&Partner und RWTH den Fragebogen. Daraufhin wurde der überarbeitete Fragebogen an die Usecase-Owner im November 2022 versendet.

Vorbereitung der Industrialisierbarkeit von KIZAM

Basierend auf der Einschätzung der repetitiven und aufwandintensiven Prozessschritte im gesamten AM-Lifecycle stellte sich nun die Aufgabe zu identifizieren, an welchen Stellen die NLP/KI-Algorithmik den meisten Nutzen bringen kann und welche

Anwendungsmöglichkeiten sie bietet. LuP regte die Diskussion über die Industrialisierung und den Austausch über Lösungen und Ansätze an.

Die Kernfrage war, wo und wie die im Rahmen von KIZAM von den Konsortialpartnern entwickelten Algorithmen in die Breite transferiert werden können. Es galt zu evaluieren, welchen Mehrwert diese Algorithmen im Hinblick auf Zeitersparnis, Beschleunigung der Arbeitsschritte und Qualitätssteigerung bringen können. Das sollte im Idealfall einen Nutzen für alle ALM-Tool-Anbieter darstellen.

Die sich in Arbeit befindende Integration des BMW NLP/KI-Tools Relyze in Codebeamer – über RestAPI – sowie andere bereits existierende KI-Anbindungen wurden von LuP im Hinblick auf ihre Vor- und Nachteile sowie auf ihr Potenzial bewertet. Schwachstellen im Hinblick auf den Datenverlust bei der Modellierung von Metadaten wurden u. A. beim Einsatz der OSLC-Plattform identifiziert. Die Fragen nach einer geeigneten Systemarchitektur, der Passbarkeit der RestAPI (bei der Anbindung von Codebeamer an Relyze) sowie nach dem Einsatz von Plug-Ins als Form der Anbindung wurden erörtert. Als nächster Schritt galten die Vorbereitungen für die Befähigung eines ALM-Tools zur Einbindung von KI.

Ist-Analyse der BMW-Prozesse im Anforderungsmanagement

Der Produktentstehungsprozess bei BMW wurde ausführlich erläutert und in seiner Komplexität aufgezeigt. Die Stärken- und Schwächenanalyse umfasste sowohl Aspekte der Methodik und der Prozesse als auch – was ja im Rahmen von KIZAM besonders relevant ist – das Tooling. Der Vergleich zwischen dem seit langer Zeit bei BMW befindlichen ALM-Tool DOORS und dem neuen ALM-Tools Codebeamer zeigte das Potential vor allem im Hinblick auf wichtige Aspekte wie Attribuierung, Traceability und auf den Automatisierungsgrad. Letzter betrifft zurzeit die laufende Integration BMW-eigenen Programms Relyze in Codebeamer und die geplante weiterführende Automatisierung der FiFa-Qualitätschecks. Darüber hinaus wurde eine Codebeamer-Schulung für die KIZAM-Konsortialpartner gehalten.

Analyse der Integration der BMW-Anwendung Relyze und der technischen Lösungsansätze

Die bei BMW laufende Integration von Relyze in Codebeamer in Form eines externen Widgets wurde im Hinblick auf nützliche und erforderliche Use Cases und Verbesserungen, die sich aus dem Testing der BMW-User ergeben, mitverfolgt. Die Betrachtung lieferte wichtige Inputs, die zu einem späteren Zeitpunkt bei der Integration der technischen Lösungsansätze von KIZAM nützlich sein werden.

Darüber hinaus wurde die Integration auch aus dem Blickwinkel des operativen Anforderungsmanagements von Lösch & Partner betrachtet. Hierzu wurden auch konkrete Tests mit Ad-Hoc-Anforderungen – die häufige Fehlerquellen im alltäglichen operativen Anforderungsmanagement darstellen – durchgeführt. Diese speziellen Use Cases/Fehlerquellen und mögliche weiterführende Einsatzszenarien wurden am beispielhaften Einsatz von Relyze in Codebeamer im operativen Anforderungsmanagement aufgezeigt.

Die Zielsetzungen der technischen Lösungsansätze von KIZAM wurden ebenso aus dem Blickwinkel des operativen Anforderungsmanagements von Lösch & Partner analysiert. Die Überführung von Anforderungen in natürlicher Sprache in eine semiformalisierte Sprache (Structured English) sowie die Überführung von Anforderungen in ein Systemmodell wurden als realitätsnah und von großem Nutzen bewertet, da dies viele menschliche Schritte vereinfachen würde. Die Realisierung der technischen Lösungsansätze von KIZAM „Formalisieren & Prüfen“ und „MSBE-Integration & Virtuelles Testen“ deckt somit einen wichtigen Bedarf im Rahmen des operativen Anforderungsmanagements ab.

Klärungen im Hinblick auf die Integration der KIZAM-Lösungen

Während die Widget-Entwicklung für die Integration von Relyze in Codebeamer von PTC durchgeführt wurde – da die Firma Lösch & Partner im Falle von Relyze nur für die Anbindung des Widget-Docker-Containers an Codebeamer verantwortlich war – wurden Klärungen durchgeführt, wie das Umsetzungsmodell für die technischen Lösungsansätze von KIZAM aussehen soll.

2023 wurde die Frage nach der Einbindung von Lösch & Partner-DevOps für die Integration der drei technischen Lösungsansätze von KIZAM über die Widget-Schnittstelle, die auf der Vorlage von Relyze aufbauen soll, erörtert und zwischen Lösch und BMW geklärt.

Es wurde beschlossen, dass die Integration der drei KIZAM-Lösungen in der geeigneten BMW-Umgebung stattfinden sollte.

Ein Änderungsantrag für die Mehrung der ursprünglich beplanten Ressourcen wurde an den Förderer gestellt.

Lösch & Partner-Mitarbeiter wurden für die anstehenden Aufgaben identifiziert. Die genaue Aufteilung von Aufgaben zwischen BMW, Schäffler, Lösch & Partner und den anderen Konsortialpartnern wurden intensiv besprochen und feingranular ausgearbeitet. Die wesentliche Änderung bestand darin, dass die Firma Lösch & Partner das KIZAM-Projekt nicht nur prozessual unterstützen, sondern einen wichtigen Beitrag bei der Integration der KIZAM-Lösungsansätze in das ALM-Tool Codebeamer leisten sollte.

Die beantragten Umfänge betrafen das Aufsetzen der benötigten Schnittstelle (Widget) aber nicht deren Wartung oder Produktivnahme. Da die Komplexität der KIZAM-Lösungsansätze höher als die des Relyze-Programms ist, war mit einem Ausbau der Funktionalitäten in Verbindung mit dem Widget zu rechnen.

Das Ziel war die Wiederverwendbarkeit der Algorithmen über die reine BMW-spezifische Umsetzung des Relyze-Widgets und somit die Industrialisierbarkeit hinaus. Ein weiterer angedachter (aber optionaler) Schritt war die maximal generische Gestaltung der Anbindung der KIZAM-Lösungen.

Letztendlich wurde die gesamte Umsetzung des „KIZAM-Widget“ unter dem Vorzeichen einer in technischer Hinsicht skalierbaren Anbindung in Codebeamer realisiert.

Vorbereitungen für die Anbindung in Codebeamer

Im zweiten Halbjahr 2023 hat Lösch & Partner verstärkt an der Implementierung eines

externen Widgets gearbeitet, um die Integration der KIZAM-Lösungsansätze, nachfolgend KIZAM-Use Cases genannt, in das ALM-Tool Codebeamer zu ermöglichen. Ausgangspunkt der Diskussion war, dass die Integration der KIZAM-Use Cases über eine Widget-Schnittstelle erfolgen soll, die auf der Vorlage der RELYZE-Integration („Relyze-Widget“) aufbaut. Die bestehende BMW-eigene Anwendung RELYZE wurde im ersten Halbjahr 2023 (vorerst ohne KI) von Lösch & Partner-DevOps über eine Widget-Schnittstelle in der BMW-TEST-Umgebung an das ALM-Tool Codebeamer angebunden. Die ersten Schritte waren eine eingehende Untersuchung des extern zur Verfügung gestellten Widgets, das von der Firma PTC für die Integration von RELYZE in Codebeamer programmiert worden war. Dies betraf den Datenfluss zwischen Software, API, Backend-API sowie den von PTC zur Verfügung gestellten Code. Im Laufe der Zeit stellte sich heraus, dass die durch Lösch & Partner zu implementierende Widget-Schnittstelle nur für zwei der drei KIZAM-Use Cases eingesetzt werden soll, nämlich für „Formalisieren und Prüfen“ und für „Suchen und Finden“.

Für KIZAM wurde der Datenfluss und die Kommunikation zwischen den einzelnen Komponenten wie im Bild dargestellt festgelegt:

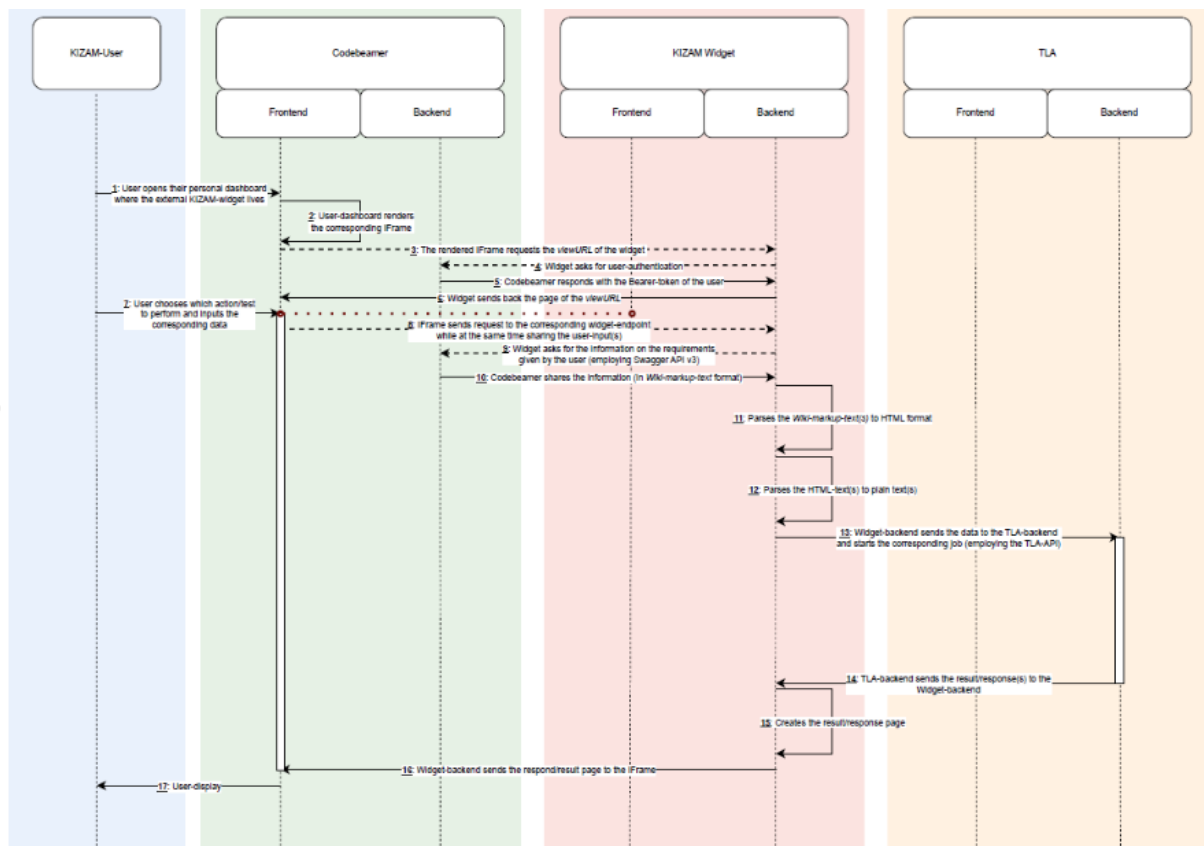


Abb. 8: Sequenzdiagramm für die Interaktion zwischen Benutzer, Codebeamer, Widget, TLA

Das folgende Bild veranschaulicht, welche Informationen zwischen Frontend und Backend des Widgets ausgetauscht werden und wo sich die Schnittstellen (Endpoints) der TLAs befinden:

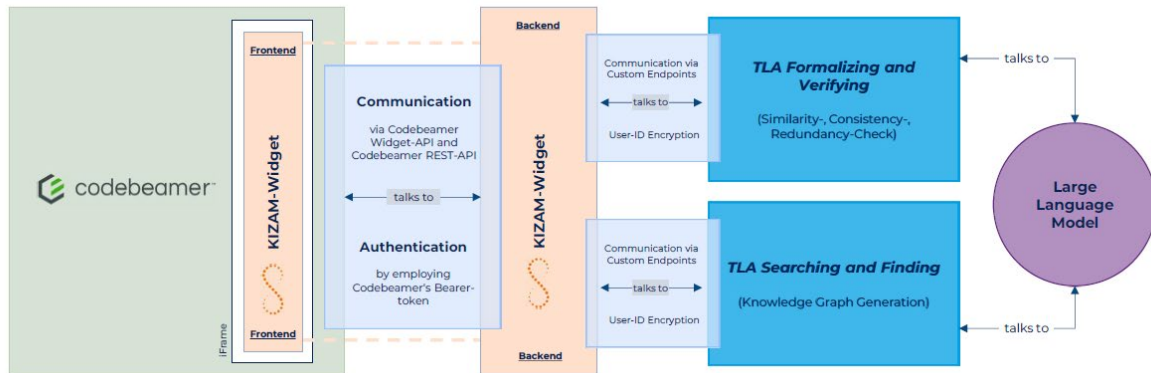


Abb. 9: 1 Gesamtübersicht KIZAM Front- und Backend

Implementierung des Initialaufbaus für die Widget-Schnittstelle KIZAM-KI (Use Case-übergreifend)

Für die Implementierung des Initialaufbaus wurde im Rahmen eines *widget prototype development* zuerst ein Widget-Prototyp, programmiert und dabei die Grundfunktionalitäten mithilfe von Java Standard Library implementiert. Das Backend des Widget-Prototyps wurde in Java Standard Library programmiert. Das Frontend wurde mit HTML, CSS und Javascript realisiert. Es wurde die Frage behandelt, wie die Kommunikation zwischen Frontend und Backend, die in zwischen unterschiedlichen Programmiersprachen implementiert wurden, zu erstellen sei. Es erfolgte die Konzeptionierung des Widget-Backend-Endpoints, um die Kommunikation mit dem Frontend zu ermöglichen. Die Endpoint-Struktur wurde implementiert und dabei wurden verschiedenste Programmierungsfragen gelöst, beispielsweise das Problem mit dem CORS-Mechanismus. Die Schnittstelle (REST-API) zwischen Software-Widget-Backend und dem KIZAM-Use Case „Formalisieren & Prüfen“ wurde in der Theorie durchgesprochen. Alle fünf vorgesehenen Funktionalitäten wurden im Hinblick auf Input und Output spezifiziert:

- Prüfung auf Konsistenz,
- Prüfung auf Ähnlichkeit,
- Prüfung auf Redundanz,
- Anfrage nach Formalisierung nach Structured English,
- Posten von Anforderungen im richtigen Structured English.

TLA Formalisieren und Prüfen

Inputs und Outputs

Der TLA *Formalisieren und Prüfen* hat fünf verschiedene Funktionalitäten bzw. Schnittstellen:

Summary	Input	Output	Bemerkung
<p><u>Prüfung auf Konsistenz</u></p> <p><i>Gibt es Widersprüche in einer Menge von Anforderungen?</i></p>	<p>Eine Menge M von Anforderungen</p>	<p>Eine Teilmenge $S \subseteq M$ von inkonsistenten Anforderungen</p>	<p>Der User <i>solte</i> die Anforderungen der Ergebnismenge S korrigieren und dann nochmal eine neue Inkonsistenz-Prüfung durchführen.</p> <p>Es ist geplant, dass der TLA zusätzlich auch ein (menschlich verständliches) Gegenbeispiel liefert, das die Inkonsistenz der Menge S zeigt.</p>
<p><u>Prüfung auf Ähnlichkeit</u></p> <p><i>Wie ähnlich ist eine Anforderung zu den Anforderungen einer Menge?</i></p>	<p>Eine Anforderung A und eine Menge M von Anforderungen</p>	<p>Nach Ähnlichkeit sortierte Liste der Anforderungen aus M</p>	<p>Die Ähnlichkeit wird hier als eine Zahl zwischen 0 und 1 dargestellt, wobei die Zahl 0 keine Ähnlichkeit bedeutet.</p> <p>Man kann hier einen Threshold einführen, um die Anzahl der zurückgegebenen Anforderungen einzugrenzen; zum Beispiel alles, was unter $0,3$ ist, wird nicht ausgegeben.</p> <p>Nach der Durchführung dieser Prüfung muss der User trotzdem nochmal die ähnlichsten Anforderungen analysieren, um die Ähnlichkeit menschlich zu bestätigen.</p>

Abb. 10: Vereinbarungen bzgl. Input und Output für die Prüfung auf Konsistenz und Ähnlichkeit

Als Input dienen im Wesentlichen eine Menge (Set) an Anforderungen und eine Einzelanforderung. Die Besonderheiten der Output-Ergebnisse sowie mögliche Weiterentwicklungen von „Formalisieren & Prüfen“ wurden ebenfalls erörtert. Basierend auf dieser Feinspezifizierung wurden von Lösch & Partner die passenden Software-Strukturelemente im ALM-Tool Codebeamer identifiziert, die ermöglichen sollten, ein Set an Anforderungen zu erstellen und als Basis für die fünf verschiedenen Funktionalitäten zu verwenden. Es sollten dieselben Elemente für alle fünf Funktionalitäten verwendet werden.

Um eine Menge von Anforderungen zusammenzufassen, die dann an „Formalisieren & Prüfen“ gesendet wird, wurde das Strukturelement „Report“ und die „Report-ID“ ausgesucht. Die Einzelanforderung, die für den Abgleich gegen das Set an Anforderungen eingegeben werden soll, wurde anhand der „Item-ID“ (im Codebeamer entspricht das einer Einzelanforderung) ausgewählt.

Die „Report-ID“ bietet den Vorteil, dass man – auch projektübergreifend – Anforderungen, sprich Items, aus verschiedenen Trackers (Trackers sind ebenfalls wichtige Codebeamer-Strukturelemente) zusammenstellen kann.

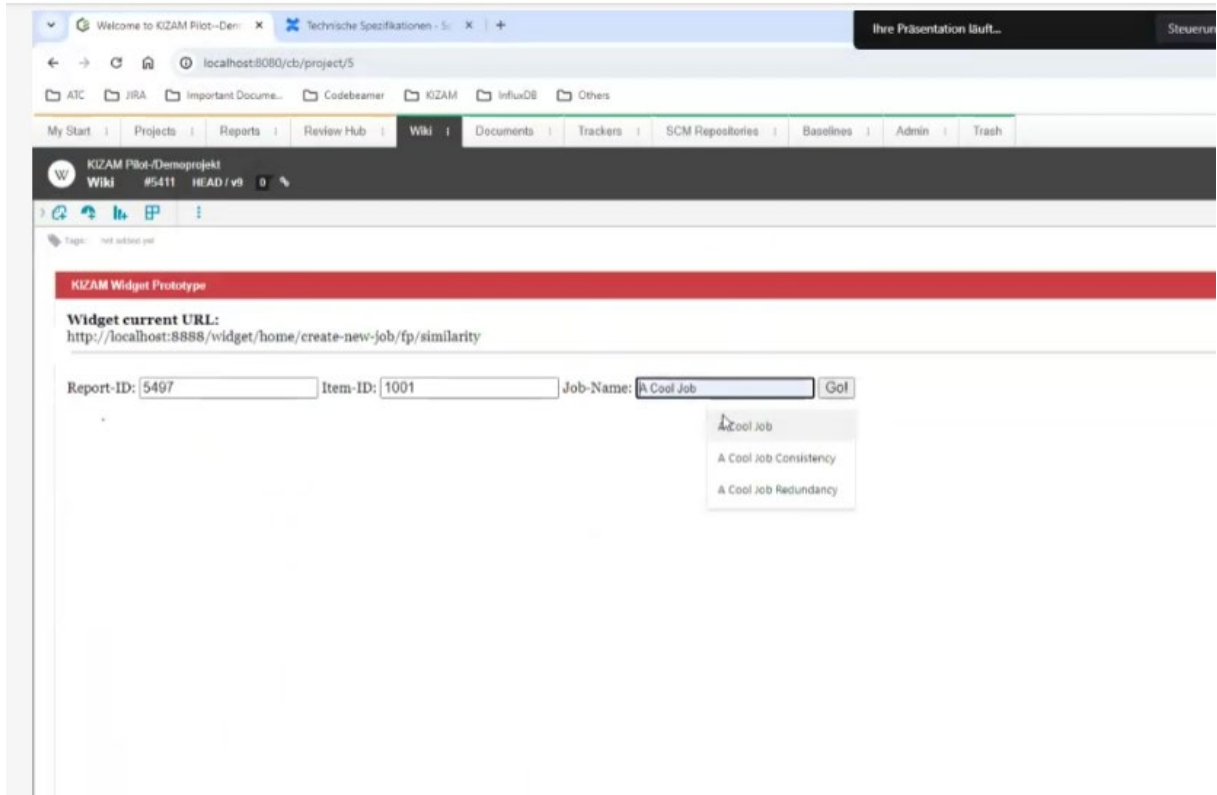


Abb. 11: Beispiel für Anlage eines Jobs im Widget, um verschiedene Funktionalitäten von „Formalisieren & Prüfen“ zu verwenden, Stand Oktober 2023

Nachfolgend ist beispielhaft als Code die Schnittstelle für das Starten eines Jobs abgebildet:

Request POST: in URL

```
{
  "job_type": "consistency|similarity|redundancy|formalization",
  "name": <string>,
  "user_note": <string>,
  "requested": {
    "query_req": {"req_id": <id>, "req": <string>}, | undefined, // undefined in case
of consistency and formalization
    "set": [{"req_id": <id>, "req": <string>}, ...]
  }
}
```

Abb. 12: Mit diesem Request wird ein Job (z. B. Similarity oder Consistency) gestartet.

Es wurde vereinbart, dass der Benutzer die Möglichkeit haben soll, den Status aller Jobs abzurufen, indem er auf die Schaltfläche 'See all Jobs' klickt. Jedes Mal, wenn ein User diesen Button klickt, schickt das Widget-Backend eine Anfrage (GET-Request) zum TLA-Backend, das dann die aktuellen Job-Status zurückgibt. Diese Antwort vom TLA wird dann vom Widget-Backend geparsed und in Codebeamer (also am Widget-Frontend) angezeigt. Hier als GET-Request dargestellt:

```
-----  
Statusabfrage Benutzer (alle Jobs)  
Status of User (all Jobs)  
-----  
Request GET: in URL // 29.01.2024 Agreement @Yosua Lie (ext.) @Haron Nqiri (ext.) : since the Bearer-token is now  
accessible, we do not need any user-id.  
Response:  
  
{  
  "joblist": [  
    {  
      "job_id": <id>,  
      "name": <string>,  
      "user_note": <string>,  
      "pending": "true"|"false",  
      "message": <string>,  
      "job_type": "consistency"|"similarity"|"redundancy"|"formalization"  
    }, ...  
  ]  
}
```

Abb. 13: GET-Request für die Anzeige aller Jobs im Frontend ("See all Jobs")

Umsetzung einer lokalen Anbindung Software-Widget-Backend und Live-Demo für Konsortialtreffen

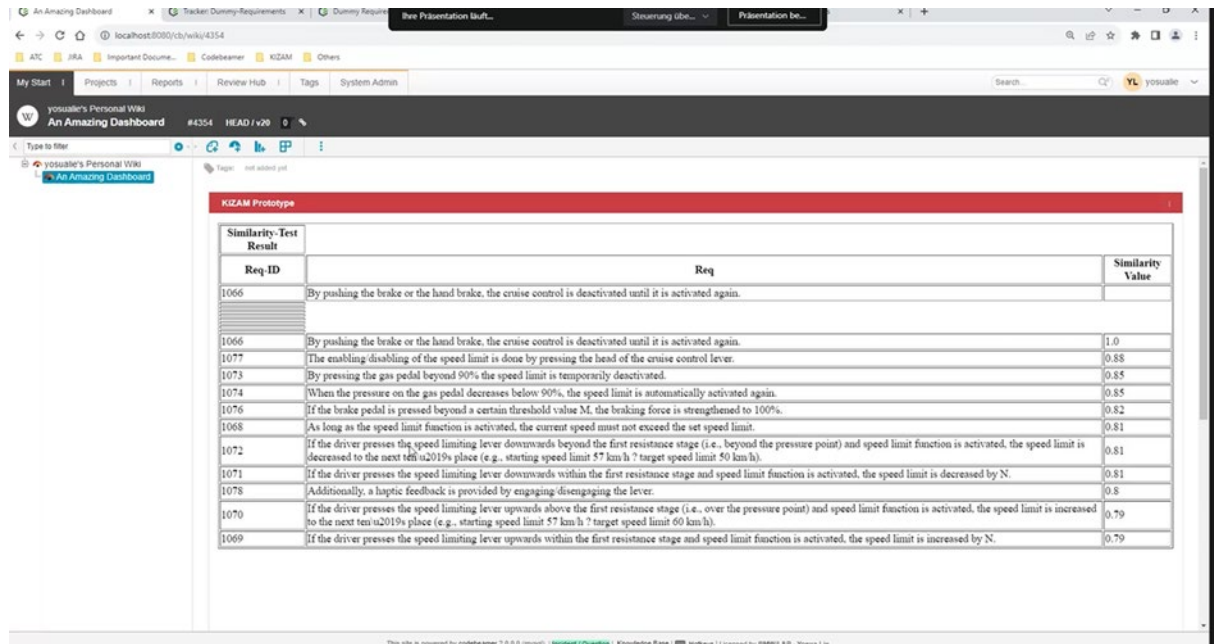
In enger Zusammenarbeit mit SE wurde an Lösungen gearbeitet, um eine lokale Anbindung zwischen Widget und Backend zu realisieren, die es ermöglichte, eine der Funktionalitäten des ersten KIZAM-Use-Case „Formalisieren und Prüfen“ auf einem lokalen Entwickler-Rechner in der BMW-Umgebung zu implementieren. Die lokale Anbindung wurde zu Demo-Zwecken für das Konsortialtreffen im Oktober realisiert und im Rahmen des Treffens vorgeführt.

Es wurde ein MVP (*Minimum Viable Product*) des ersten KIZAM-Use Cases realisiert. Der MVP war in der Lage, 15 Anforderungen anzunehmen und wurde auf dem lokalen Rechner des Entwicklers von Lösch & Partner in der BMW-Umgebung installiert. Endpoints konnten erfolgreich installiert werden, um die Kommunikation zwischen Widget und „Formalisieren & Prüfen“ zu ermöglichen bzw. Informationen durchzugeben, die über das Widget in der Instanz des ALM-Tools Codebeamer angezeigt werden konnten. In diesem Rahmen wurden

etliche Fragen (z. B. Proxy-Einstellungen) angegangen.

Insbesondere ging es bei der Live-Demo auf dem Konsortialtreffen um die Funktionalität „Ähnlichkeitsanalyse“, deren Ergebnisse sinnvollerweise in einer Übersicht mit Prozentwerten angezeigt werden sollten.

Das Frontend musste zu diesem Zweck ergänzt werden. Es musste sichergestellt werden, dass die für die Übersicht benötigten Parameter eingegeben werden können. Die Ähnlichkeitswerte (in %) wurden erfolgreich im Frontend angezeigt.



Req ID	Req	Similarity Value
1066	By pushing the brake or the hand brake, the cruise control is deactivated until it is activated again.	
1066	By pushing the brake or the hand brake, the cruise control is deactivated until it is activated again.	1.0
1077	The enabling/disabling of the speed limit is done by pressing the head of the cruise control lever.	0.88
1073	By pressing the gas pedal beyond 90% the speed limit is temporarily deactivated.	0.85
1074	When the pressure on the gas pedal decreases below 90%, the speed limit is automatically activated again.	0.85
1076	If the brake pedal is pressed beyond a certain threshold value M, the braking force is strengthened to 100%.	0.82
1068	As long as the speed limit function is activated, the current speed must not exceed the set speed limit.	0.81
1072	If the driver presses the speed limiting lever downwards beyond the first resistance stage (i.e., beyond the pressure point) and speed limit function is activated, the speed limit is decreased to the next ten ^u 2019s place (e.g., starting speed limit 57 km/h ? target speed limit 50 km/h).	0.81
1071	If the driver presses the speed limiting lever downwards within the first resistance stage and speed limit function is activated, the speed limit is decreased by N.	0.81
1078	Additionally, a haptic feedback is provided by engaging/disengaging the lever.	0.8
1070	If the driver presses the speed limiting lever upwards above the first resistance stage (i.e., over the pressure point) and speed limit function is activated, the speed limit is increased to the next ten ^u 2019s place (e.g., starting speed limit 57 km/h ? target speed limit 60 km/h).	0.79
1069	If the driver presses the speed limiting lever upwards within the first resistance stage and speed limit function is activated, the speed limit is increased by N.	0.79

Abb. 14: Anzeige der Ähnlichkeitswerte bei der Live-Demo auf dem Konsortialtreffen im Oktober 2023

Testing allgemeiner Funktionalität des Widgets (Use-Case-übergreifend)

Im Nachgang an die erfolgreiche lokale Anbindung zwischen Demo-MVP und Widget mussten weitere Änderungen am Widget vorgenommen werden, um sicherzustellen, dass neben der demonstrierten Funktionalität „Ähnlichkeitsanalyse“ auch die restlichen vier Funktionalitäten von „Formalisieren und Prüfen“ reibungslos angebunden werden können.

Feinspezifizierung der Schnittstelle zwischen Software-Widget-Backend des zweiten KIZAM-Use Case „Suchen & Finden“ (Theorie)

Nachdem der erste KIZAM-Use Case, was die Anbindung über Widget angeht, gut vorangekommen war, stellte sich Frage, ob das erstellte Widget-Sequence-Diagramm auch für den Datenfluss beim zweiten KIZAM-Use Case „Suchen & Finden“ übernommen werden konnte.

Die Schnittstelle wurde im Austausch mit dem Konsortialpartner WZL fein spezifiziert. Input, Output und Regeln der Kommunikation wurden definiert. Die Parameter, die beim zweiten KIZAM-Use Case im Frontend eingegeben werden können, sprich der Input, sind ein Satz, eine neue Anforderung sowie Stichwörter. Auch die Definition des Suchraums gehörte dazu, wenn auch sie erst später realisiert wurde.

Bei den Output-Ergebnissen galt die Anforderung, dass über eine HTML-Datei der Knowledge Graph mit den semantischen Abhängigkeiten zwischen den Graph-Knoten in der CB-Instanz angezeigt werden soll. Es galt zu testen, ob die HTML-Datei vom Widget verstanden wird. Es wurde erfolgreich getestet, dass Chrome in der BMW-Umgebung die HTML-Dateien parsen kann – das Anklicken bzw. Verschieben der Graph-Knoten war möglich.

Es wurde darüber diskutiert, inwiefern – analog zu „Formalisieren und Prüfen“ – ein MVP lokal beim Lösch-Entwickler in der BMW-Umgebung installiert werden könnte. Das Aufbauen von Endpoints für das MVP lokal beim Lösch-Entwickler stellte bei „Suchen & Finden“ eine besondere Herausforderung dar.

Es stellte sich heraus, dass die vom Konsortialpartner WZL angefertigte Lösung in der BMW-Umgebung nicht anzeigbar war, da sie aufgrund von Proxy- bzw. Firewall-Einstellungen von der BMW-Umgebung (lokal beim Entwickler von Lösch & Partner) nicht erreichbar war. Es wurde u. A. versucht, die Proxy-Einstellungen nur auf Request-Ebene – also nicht auf Applikationsebene – dahingehend zu ändern, dass die Kommunikation möglich war. Dies blieb trotz der Versuche vorerst erfolglos. Hier musste noch von WZL ein Endpoint zur Verfügung stellen.

Untersuchung unterschiedlicher Frameworks im Hinblick auf Eignung für BMW-Umgebung und KIZAM-Use-Case-Spezifika

Mit BMW wurden Diskussionen über das passende Framework für das Widget geführt. Da das Widget zuerst in Java Standard Library programmiert worden war, war es notwendig, verschiedene Frameworks in Betracht zu ziehen. Die Auswahl des passenden Frameworks ist deshalb wichtig, weil sie gravierende Auswirkungen auf Kosten, Performance, Skalierbarkeit hat. Recherchen und ein gezielter Auswahlprozess waren erforderlich und wurden ausgeführt. Schließlich wurden vier verschiedene Frameworks untersucht:

- Angular,
- Micronaut,
- Java Standard Library,
- Quarkus.

Beispielsweise kam Angular in die engere Wahl, weil die Herstellerfirma von Codebeamer, PTC, dieses Framework für die Implementierung des externen Widgets für Relyze verwendet hatte. Angular war bereits in den Anfängen der Konzeptionierung der technischen Implementierung durch Lösch & Partner untersucht worden.

Das Widget wurde in einem hinreichenden und notwendigen Umfang – ca. 80% – in allen vier Frameworks implementiert, um die beste Wahl zu treffen. Die Wahl fiel schließlich auf Quarkus. Auch die Arbeiten an der Live-Demo auf dem Konsortialtreffen waren nützlich und

relevant, um die finale Entscheidung zu treffen.

Kriterien, die ebenfalls eine Rolle bei der Auswahl gespielt haben, waren, dass ein externes Widget eine kleine Applikation ist, die serverlos, sprich funktionsbasiert, arbeitet, ein- bzw. ausgeschaltet werden kann und keine Ressourcen verbraucht, wenn sie nicht genutzt wird. Solche Überlegungen waren im Rahmen einer Umsetzung z. B. mit dem Cloud-Provider Microsoft Azure erforderlich. Zudem ist Quarkus entwicklungsfreundlich. Beispielsweise kann man damit Hot-Coding durchführen. Quarkus ist für serverlose Applikationen, für Microservices, geeignet und hat z. B. eine bessere Startup-Time. Darüber hinaus ist Quarkus mit existierenden Applikationen wie z. B. Java Libraries integrierbar. Das führte dazu, dass keine neuen Libraries verwendet werden mussten. Das Widget wurde demzufolge vollständig in Quarkus programmiert. Die Technologie im Backend wurde geändert. Umfangreiche Tests waren erforderlich, um zu gewährleisten, dass alle (externen) Funktionalitäten mit der neuen Backend-Technologie reibungslos arbeiten können. Implementiert wurde letztlich ein *lightweight widget*. Am Aussehen hatte sich nichts geändert, aber im Hintergrund hat ein *refactoring* des Widgets stattgefunden, sprich eine Verbesserung der Backend-Technologie.

Erster Versuch auf Codebeamer Test-Umgebung; Packaging und Containerization; Auswahl Server; Pilot in der Test-Umgebung mit Dummy-Anforderungen

Vorbereitend für die Implementierung in der BMW TEST-Umgebung wurde an den kleinen, aber wichtigen Unterschieden gearbeitet, die unweigerlich zwischen der lokalen Implementierung auf dem Entwickler-Rechner in der BMW-Umgebung und der BMW TEST-Umgebung existieren.

Da die Anbindung über eine REST-API geht, musste man die API-Throttling-Einstellung der TEST-Umgebung berücksichtigen. Hierzu wurde die Methode zum Verschicken und Empfangen der Requests durch das Backend verbessert.

Die API-Throttling-Einstellung der TEST-Umgebung überprüft, wie viele Anfragen und von welchen Usern pro Minute (oder Sekunde) an Codebeamer verschickt werden.

Codebeamer API-Throttling bedeutet eine Beschränkung der Anzahl der REST-API Requests pro Minute und pro Stunde. Dies ist notwendig, da gleichzeitig viele REST-API Requests verschickt werden können. Wenn dies eintritt, kann es passieren, dass Codebeamer eine Fehlermeldung generiert (Error-Code 429, „Too many requests“). Um dieses Problem zu behandeln wurde ein Retry-Algorithmus implementiert, bei dem das Programm eine bestimmte Zeit wartet, bevor es wieder versucht, die Requests zu schicken.

```

|{
  "apiThrottling" : {
    "urlPatterns" : "/rest/**, /api/**",
    "bandwidthConfigs" : [
      {
        "capacity" : 600,
        "timeUnit" : "MINUTE"
      },
      {
        "capacity" : 36000,
        "timeUnit" : "HOUR"
      }
    ]
  },
  ...
}

```

Abb. 15: Beschränkung der Anzahl der REST-API Requests pro Minute/Stunde

Auch für das Packaging später auf der TEST BMW-Umgebung wurden bei der lokalen Implementierung auf dem Entwickler-Rechner relevante Aspekte von vornherein berücksichtigt.

Für das Weiterkommen in diesem Punkt war Lösch & Partner auf das Deployment der beiden KIZAM Use-Cases in der BMW TEST-Umgebung durch die Konsortialpartner SE, WZL und BMW angewiesen.

Optimierung der aktuellen Implementierung

Im ersten Halbjahr 2024 hat Lösch & Partner verstärkt an der Weiterentwicklung des externen Widgets gearbeitet. Dies erfolgte in engster Zusammenarbeit mit den Forschungsinstituten SE und WZL und war erforderlich, um die TRL-7 Reife zu erreichen. Einige Optimierungen an der User Interface ergaben sich aus den Demo-Sitzungen für die Konsortialpartner, aus den Dry Runs als Vorbereitung für die Experteninterviews im Sommer sowie aus selbstständigen Tests der implementierten KI-Funktionalitäten.

Konzipierung des Deployments

Es wurden rechtzeitig am Jahresanfang verschiedene Wege fürs Deployment auf dem BMW-Server (AWS Instanz) evaluiert. Es war zu diesem Zweck wichtig zu planen, wie das Widget auf dem Server läuft - z. B. als Plattform-Native Binary (z. B., GraalVM native Images) oder als Docker-Images oder sogar als Stand-Alone-Lösung (JAR-Files) mit allen Abhängigkeiten. Es musste evaluiert bzw. überprüft werden, ob das Widget auf einem eigenen Server oder auf demselben Server wie die beiden KIZAM Use-Cases laufen soll bzw. ob ausreichend Ressourcen auf dem Server vorhanden sind und wie die Skalierbarkeit und Performance beeinflusst werden.

Verlauf der Optimierungen am Widget Januar-Mai

Da bis Januar 2024 nur die Schnittstelle für den Similarity-Test angebunden war, aber nicht die Schnittstelle für den Redundanz- bzw. Konsistenz-Test, mussten zuerst intensive Abstimmungen mit dem Forschungsinstitut SE stattfinden. Die Benutzer-Oberfläche für diese beiden Funktionen sowie deren komplette Workflows musste zuerst konzipiert werden.

Da die Widget-Entwicklung bis zu diesem Zeitpunkt noch komplett lokal auf dem Entwicklerrechner stattgefunden hatte, war es erforderlich zu überprüfen, welche Unterschiede zwischen dem Entwicklerrechner und der TEST-Umgebung existierten. Ein Aspekt war beispielsweise bis zu dem Zeitpunkt nicht relevant gewesen - in der Codebeamer TEST-Umgebung ist ein API Throttling-Mechanismus im Einsatz, welcher überprüft, wie viele Anfragen und von welchen Usern pro Minute (oder Sekunde) an Codebeamer verschickt werden. Das Widget musste angepasst werden, um dieser Anforderung gerecht zu werden. Wenn in einer bestimmten Zeitspanne zu viele Anfragen verschickt werden, dann tritt eine (vorkonfigurierte) Wartezeit ein, bis wann nochmal versucht wird, die Anfragen erneut zu verschicken. Außerdem wurde auch ein Retry-Mechanismus eingeführt, der festlegt, wie oft versucht werden muss, falls mehrmals hintereinander der Error Code 429 ("Too many requests") aufkommt.

Es stellte sich heraus, dass in der Codebeamer TEST-Umgebung komplexere Anforderungen verwendet werden als die, die bis zum damaligen Zeitpunkt für die Tests auf dem Entwicklerrechner in Verwendung waren. Es gab zudem eine Besonderheit des Wiki-Textformats, die zu Kodierungsproblemen bei den deutschen Umlauten führte [Die Wiki-Seite ist der Bereich, wo das Widget lebt]. Für die Arbeit mit dem KIZAM-Use Case FP bestand die Anforderung an das Wiki-Textformat, dass der Plain Text ins JSON-Format gebracht werden soll. Es musste deshalb die Formatierung der Wiki-Texte dahingehend abgeändert werden, dass sie auch in JSON-Format übertragen werden. Die Übersetzung der Texte ins JSON-Format muss bestimmte Regeln befolgen, die allerdings vom Wiki-Textformat verletzt wurden. Darum musste letzteres abgeändert werden. Es stellte sich heraus, dass das Problem an einer veralteten Version der Rest-API-Schnittstelle lag. Es musste Debugging betrieben werden, um herauszufinden, warum das Widget nicht mehr richtig funktionierte. Schließlich war diese veraltete Version der Rest-API-Schnittstelle das Problem. Mit Einsatz der neueren Rest-API-Version waren die Kodierungsprobleme für Deutsch auch gelöst.

Im Anschluss wurde ein neuer Demonstrator erstellt, auf dem alle Tests bzw. Jobs (Redundanz, Inkonsistenz, Korrektur der formalisierten Anforderungen) liefen. Zum Ausführen des Widgets wurden prinzipiell zwei Dateien benötigt - die JAR-Datei im Widget selbst und Konfigurationsdateien außerhalb des Widgets. Die JAR-Datei liest aus der Konfig-Datei Informationen wie z. B. Port-Nummer, die URL des KIZAM-Use Case und von Codebeamer.

Mit dieser Art der Implementierung kann allerdings passieren, dass die Informationen in der Konfig-Datei für Dritte zugänglich sind. Die Lösung hierbei war, die Konfig-Datei von außerhalb in die JAR-Datei zu transferieren - um mehr Sicherheit und Schnelligkeit zu erreichen.

Bis zu diesem Zeitpunkt war der KIZAM-Use Case FP noch nicht richtig angebunden und konnte keine echten Antworten liefern. Deshalb musste für alle Antworten auf die Anfragen an den Mockups der KIZAM-Use Cases gebaut werden. Dies war notwendig, um sicherzustellen, dass das Widget in Zukunft in der Lage sein wird, die (echten) Antworten des der KIZAM-Use Cases zu parsen bzw. zu empfangen.

Im Februar setzte die Arbeit durch das Forschungsinstitut SE ein, den KIZAM-Use Case FP in der BMW-Infrastruktur einzurichten, damit nicht mehr mit Mockups gearbeitet werden musste. Dies brachte einen großen Abstimmungsbedarf mit sich.

Eine der zahlreichen Fragen, die geklärt wurden, war, ob die Originalanforderungen, die in einem Report formalisiert wurden - neben der Formalisierung selbst - ebenfalls im Cache aufbewahrt werden sollten. Bis dahin war dies nicht der Fall gewesen, aber das stellte sich als sinnvoll heraus, und zwar für den Fall, dass später exakt die identische Menge an Anforderungen in einem weiteren Report formalisiert werden. In diesem Szenario - sprich durch Abgleich mit dem "alten" Set an Anforderungen - wurde sichergestellt, dass exakt dasselbe Set nicht mehrmals formalisiert werden muss.

Viele Tests waren auch bei der ersten Verbindung zur AWS-Instanz erforderlich. Es gab hierbei Firewall-Probleme. Die AWS-Instanz was von außen nicht erreichbar - obwohl alle Einstellungen richtig konfiguriert waren. Es stellte sich heraus, dass die AWS-Instanz eine Besonderheit bzw. eine zweite Sicherheitsebene vorsieht - ein Tool, das nur innerhalb der BMW-Infrastruktur bedienbar ist, um letztendlich die eingerichteten Ports ein- oder auszuschalten bzw. freizuschalten.

Beim KIZAM-Use Case SF gab es beispielsweise die Herausforderung, dass jedes Ergebnis der Anfragen an den Knowledge Graphen selbst ein Script ist. Dieses Script muss ausgeführt werden, um dem User das Ergebnis des Knowledge Graphen zu zeigen. Um die Funktionsfähigkeit bzw. Anzeigbarkeit der KG-Ergebnisse sicherzustellen, musste dieses Script eingekapselt werden.

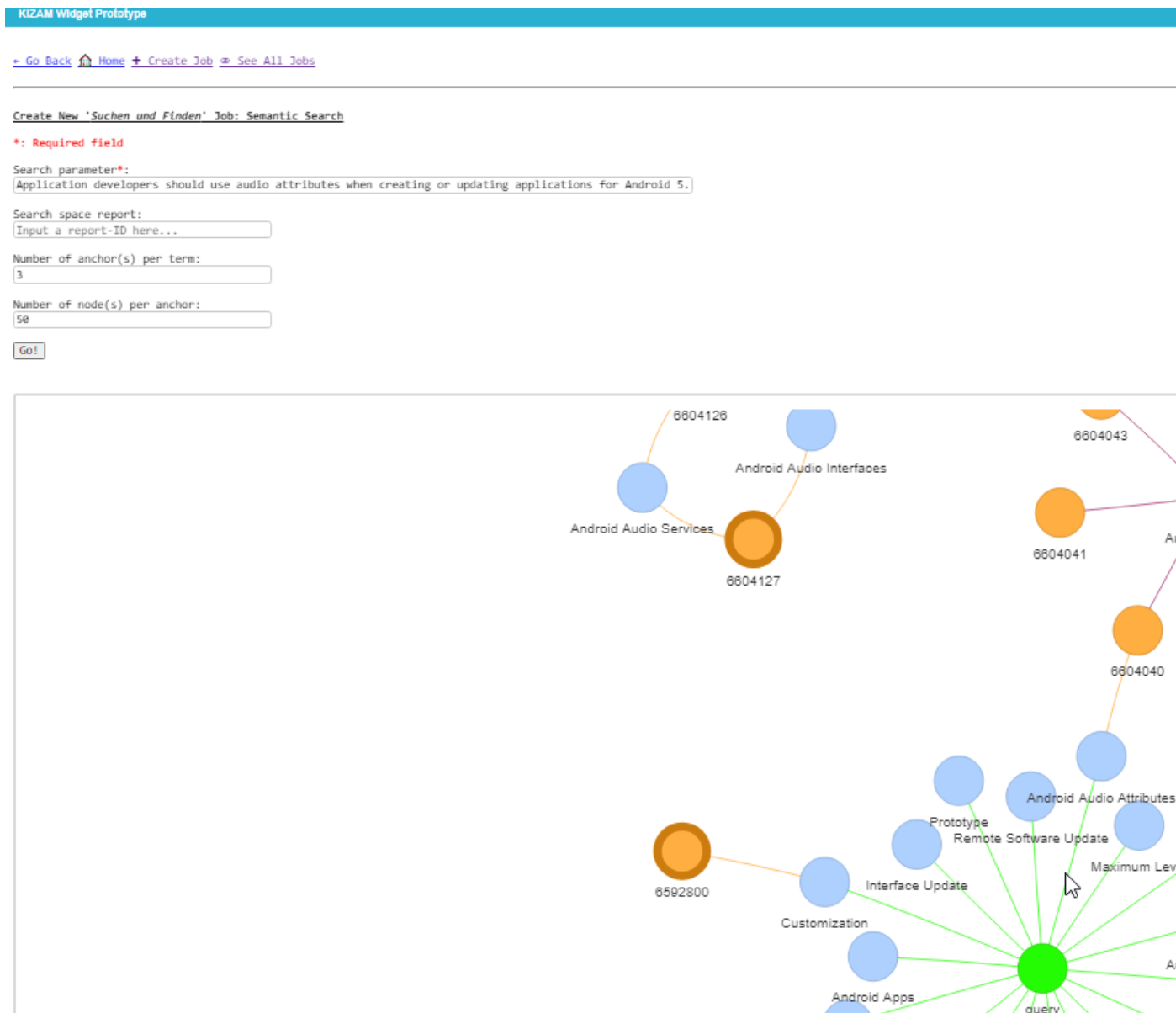


Abb. 16: Anzeige der KG-Ergebnisse in Codebeamer (KIZAM Use Case „Suchen und Finden“)

Die Definitionen der API-Schnittstellen wurden in enger Zusammenarbeit mit dem Forschungsinstitut SE kontinuierlich optimiert. Beispielsweise besteht ein Unterschied in der Implementierung der Redundanz- bzw. Konsistenz-Jobs. Bei der Redundanz kann gegen eine bestimmte Item-ID geprüft werden, während bei der Konsistenz dies nicht möglich ist. Auch die kleinste Änderung erforderte viel Zeit für Abstimmung, Implementierung und Nachtests.

Im Februar/März 2024 fanden die Abstimmungen mit dem fachlichen Product Owner von Codebeamer statt. Es musste wieder eine Demo gebaut werden, um dem Product Owner die

Workflows zu zeigen, damit er diese nachvollziehen kann. Der Product Owner stellte fachliche Anforderungen - wie z. B. die Begrenzung des Suchraum beim KIZAM-Use Case SF. Das bedeutete, dass auch für den KIZAM-Use Case SF erforderlich war, eine Report-ID einzustellen, die das Set an Anforderungen definiert, in dem nach semantischen Ähnlichkeiten gesucht wird. Des Weiteren verlangte der PO für Codebeamer den Einsatz eines bestimmten Templates beim Projekt KIZAM in Codebeamer. Zu einem späteren Zeitpunkt wurde eine weitere Anpassung des Templates angefordert, um das Arbeiten im Projekt KIZAM noch besser zu unterstützen. Auch die Frage, ob die Formalisierung der Anforderungen in Structured English in Zukunft als Attribut zur eigentlichen Anforderung (Item-ID) abgespeichert werden soll, wurde angerissen. Diese Hypothese wurde vorerst verworfen, da die Formalisierung als nicht eindeutig genug eingestuft wurde. Es wurde weiterhin an der User Interface gearbeitet - vor allem im Rahmen des Korrekturvorgangs. Es wurde festgelegt, dass - wenn innerhalb eines Reports, sprich einer Menge, nur einige der formalisierten Anforderungen korrigiert werden - nur diese korrigierten Anforderungen an das Backend des KIZAM-Use Case FP geschickt werden. Als Voraussetzung dafür wurde festgelegt, dass in der Payload die Informationen über die Menge bzw. Set als "Kontext der Formalisierung" steht, zu dem die korrigierten Anforderungen gehören.

Das HTML-Templating von Qute - als sehr performanter Templating-Framework - wurde umgesetzt, um sicherzustellen, dass der dynamische Code in die dafür vorgesehenen Platzhalter (Templates) importiert und von dort ausgeführt wird. Der Zugriff auf das Widget erfolgt aus Sicherheitsgründen nur innerhalb von Codebeamer. Darum muss der Widget-Code - in HTML, Javascript - durch HTML-Templating in die Templates transferiert und dynamisch ausgeführt werden.

Im März/April 2024 fing die Arbeit mit echten Daten und Anforderungen an. Im Rahmen des KIZAM-Use Case SF fiel beim Hochladen der Daten für den UC "Audio" auf, dass die Ergebnisse des Knowledge Graphen mit alten (sprich früher durch Hochladen in Codebeamer generierten) Item-IDs angezeigt werden. Bei jedem Hochladen in Codebeamer werden neue Item-IDs generiert und es ist nicht möglich, die alten Item-IDs beizubehalten. Darum war es erforderlich, beim Erstellen der Payloads ein Mapping zwischen alten und neuen Item-IDs vorzunehmen.

Im März/April 2024 wurden ebenfalls zahlreiche Optimierungen mit dem SE diskutiert und vorgenommen sowie End-To-End-Tests durchgeführt. Bugs worden festgestellt und mussten gefixt werden. Es waren noch etliche Änderungen an der User Interface erforderlich. Die strukturellen Entscheidungen, ob der KIZAM-Use Case FP und das Widget auf demselben Server gehostet werden müssen, wurden diskutiert und getroffen.

Die Demo-Sitzungen für die Konsortialpartner zeigten weitere Anpassungsbedarfe an der User Interface. Für jeden Anpassungswunsch musste eine Machbarkeits- und Aufwandsanalyse gemacht werden. Letztlich wurden alle mit vertretbarem Aufwand eingestuften Änderungswünsche umgesetzt, um die Bedienungsfreundlichkeit des Widgets zu verbessern. Im Rahmen der Demos für die Konsortialpartner kam z. B. der Wunsch auf, im Widget explizit anzugeben, welche Anforderungen bei einem Formalisierungsjob des

KIZAM-Use Case FP geparkt werden konnten und welche nicht. Diese Information wurde ebenfalls im Frontend zur Anzeige gebracht. Es wird im Widget explizit angegeben, welche Anforderungen im Rahmen der Formalisierung ins Structured English geparkt werden konnten („true“) und welche nicht („false“).

ID	Details	Parsable
3387108	<p>Natural language: In this chapter the development and delivery scope is to be described, i.e. all services that are connected to the delivery of the component (including prototypes).</p> <p>Current formalization: Globally, it is always the case that chapter.development_and_delivery_scope_described equals true and chapter.services_connected_to_delivery_including_prototypes_described equals true.</p>	true
3387101	<p>Natural language: All subcomponents, i.e. PCB Assembly, Seal, Leadframe, Groundpin, ... shall be listed on drawing with part no and revision to identify each change.</p> <p>Current formalization: Globally, it is always the case that for each subcomponent, (subcomponent.part_number.listed_on_drawing equals true and subcomponent.revision.listed_on_drawing equals true).</p>	false
3387102	<p>Natural language: This chapter lists the requirements with regard to Material Constraints, i.e. restricted material.</p> <p>Current formalization: Globally, it is always the case that chapter.material_constraints.list_of_restricted_materials equals true.</p>	true

Abb. 17: Die erfolgreiche Formalisierung wird durch einen Wert („true“) bestätigt.

Eine weitere wichtige Änderung - die Freitextmaske für die Formulierung der Suchanfrage beim Similarity-Job - wurde kurzfristig vor dem Konsortialtreffen im April kommuniziert und sehr zeitnah umgesetzt.

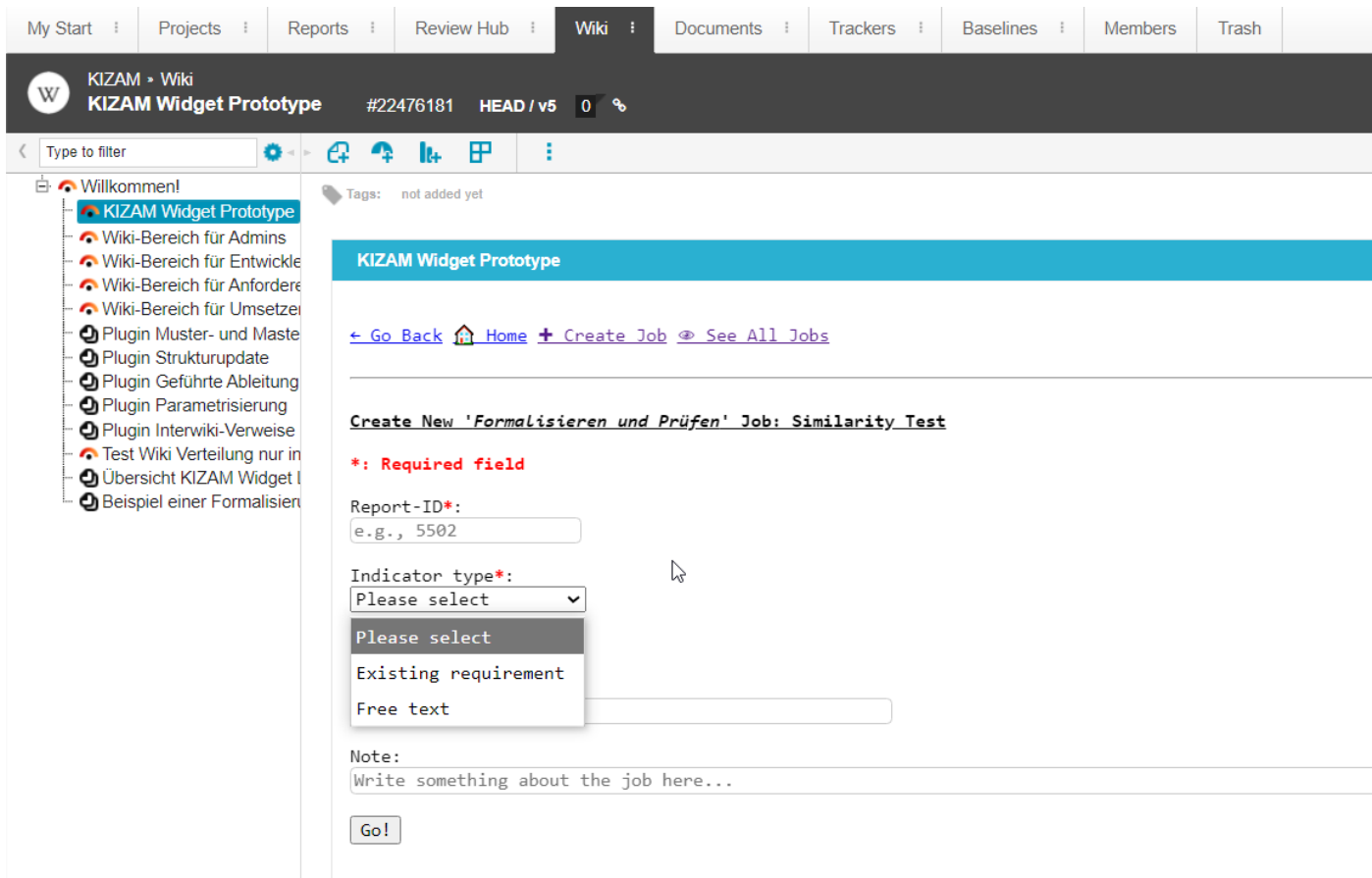


Abb. 18: Auswahl eines Freitextfeldes über Drop-Down-Menü im Similarity-Test

Diese Option im Frontend des Widgets zu bauen war wichtig, da die Ähnlichkeitsanalyse mit einem Sprachmodell durchgeführt wird, die auf Frage-Antwort-Basis funktioniert. D. h. der Benutzer muss am besten eine Frage formulieren. Ähnlichkeitsanalyse liefert – absteigend sortiert – die Anforderungen zurück, die mit der Anfrage am ähnlichsten sind. Der eingesetzte Authentifizierungsverfahren (Bearer-Token) weist das Problem auf, dass jeder zugelassene User die von anderen Usern durchgeführten Jobs im Widget sehen kann. Dies entspricht nicht den realen Arbeitsbedingungen im rechtebasierten ALM-Tool Codebeamer. Der Bearer-Token enthält die Information über den User, der einen Job startet. Um den User zu authentifizieren, wird hierzu ein Service verwendet - OpenID - der alle freigeschalteten User "kennt", und bestätigt, dass der betroffene User für Codebeamer, das Projekt und letztendlich für den Job freigeschaltet ist. Der Token für den einzelnen User ist verschlüsselt - um zu wissen, wer einen bestimmten Job gestartet hat. Um dem Anspruch der rechtebasierten Arbeit in Codebeamer gerecht zu werden, müsste man in der Lage sein, diesen Token zu entziffern bzw. decodieren. Es wurde letztendlich ein Weg gefunden, der für das KIZAM-Nachfolgeprojekt wichtig sein wird. Aber nach Absprache im Konsortium wurde diese Methode im KIZAM-Projekt nicht umgesetzt.

Ein weiteres Problem ergab sich mit der Port-ID bzw. mit der Host-ID. Es muss bei einer Web-Anwendung festgelegt werden, in welchen "Löchern" - Ports, Hosts - vom Server die Anwendung von außerhalb erreichbar sein muss. Dies war zuerst problematisch, weil das Widget über einen lokalen Host und nicht über eine IP-Adresse erreichbar war. Die letzte Hürde für das Hochladen des Widgets in die TEST-Umgebung war ein Genehmigungsantrag, der im Mai gestellt wurde.

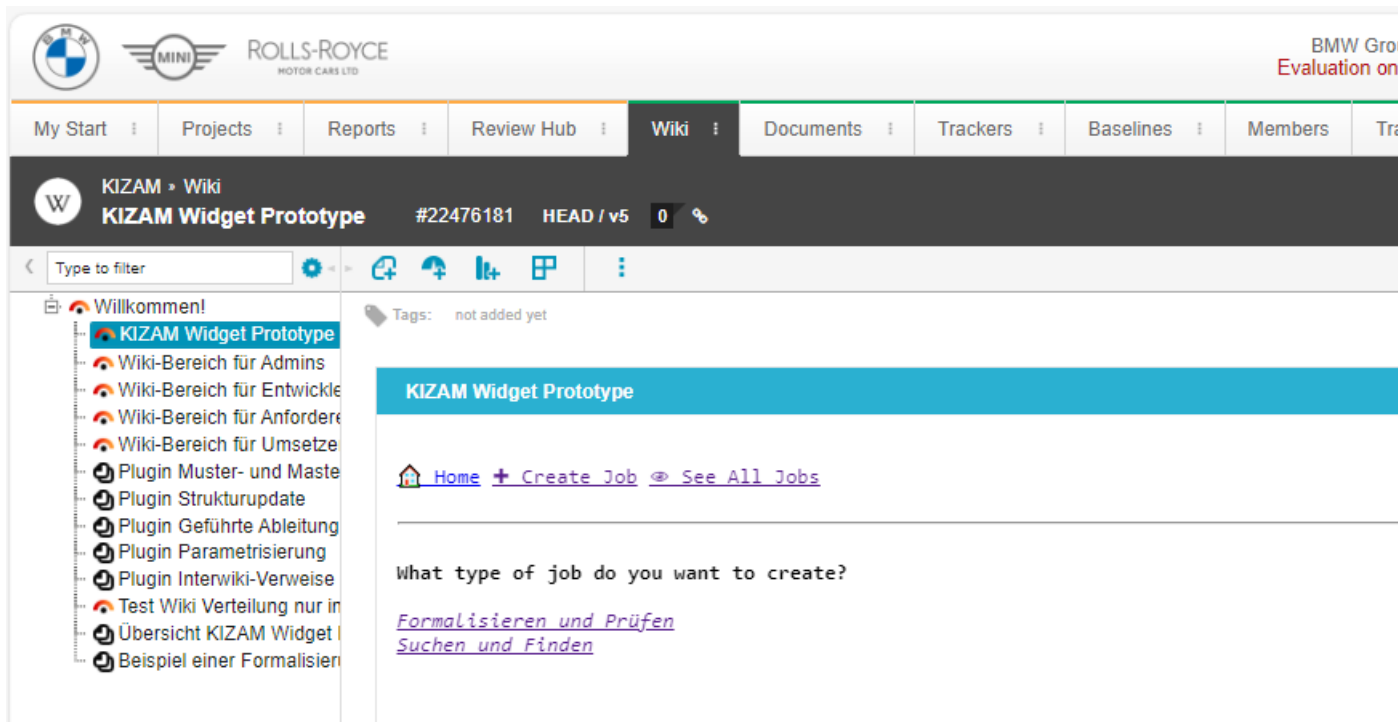


Abb. 19: Auswahl der beiden KIZAM-Use Cases „Formalisieren und Prüfen“ und „Suchen und Finden“ in Codebeamer in der BMW TEST-Umgebung

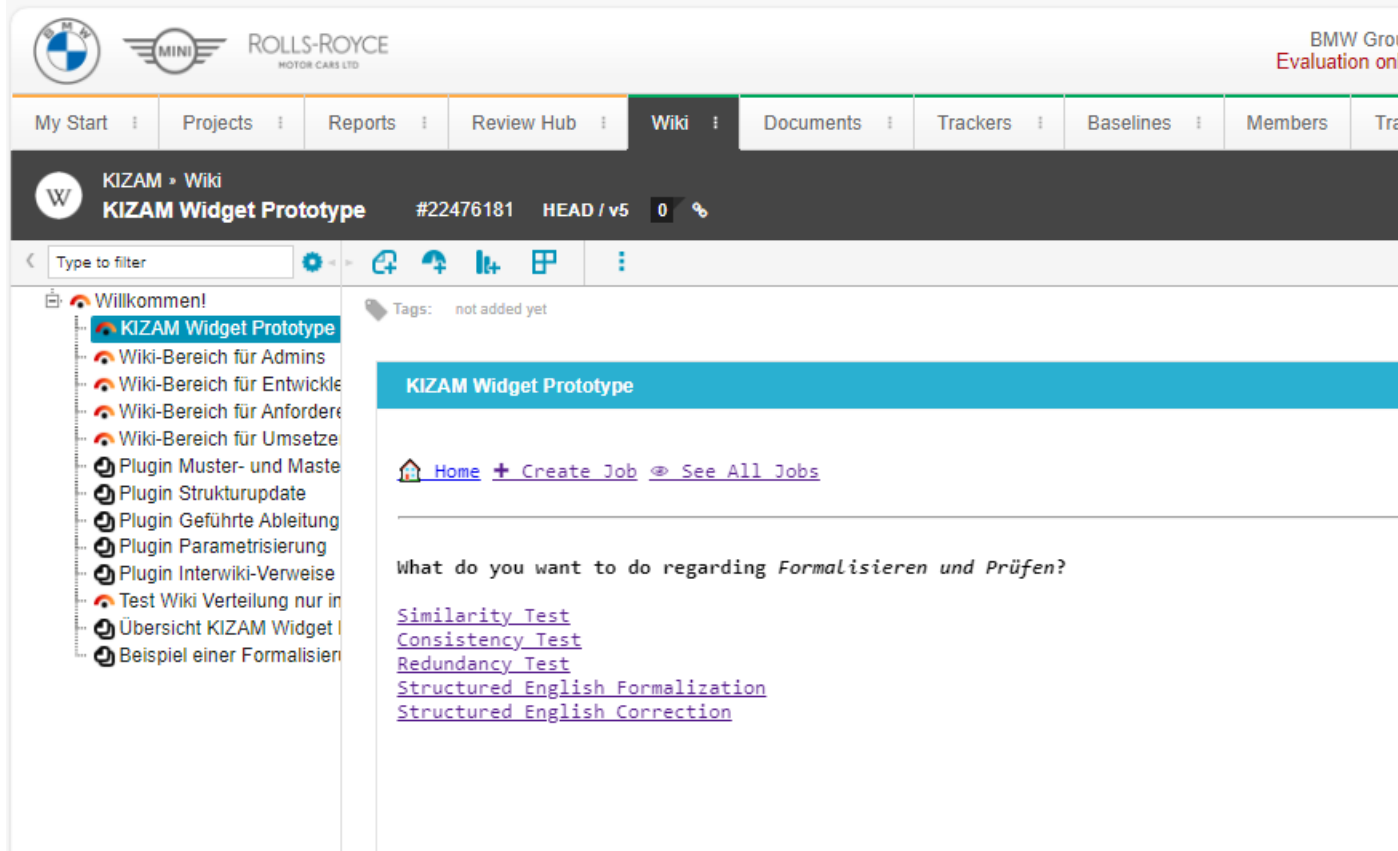


Abb. 20: Auswahl der verschiedenen Funktionalitäten von „Formalisieren und Prüfen“

Im Juli fand eine Einweisung in das Containerhandling für beide KIZAM-Use Case FP und SF statt, damit das Widget auch über den Sommer für die noch anstehenden Interviews mit den Fachexperten bedienbar bleiben und auftretenden Problemen neu angebunden werden können.

Dry Runs und Testing nach dem Hochladen des Widgets in TEST-Umgebung von BMW

Nach dem erfolgreichen Hochladen des Widgets in die TEST-Umgebung von BMW erweiterte sich im Mai/Juni der Schwerpunkt der KIZAM-Aktivitäten sowohl auf die offiziellen Dry Runs durch die UC-Owner (Audio, Frontend, Türen, Licht, TIER1) als auch auf ein generelles selbstständiges Testing der Ergebnisse - vor allem bei den Konsistenz- und Redundanz-Jobs.

Auch im Rahmen der Dry Runs hat Lösch und Partner bei den Vorbereitungen auf die Dry Runs unterstützt - sowohl durch das Anlegen der Tracker als auch durch das Hochladen der Anforderungen für die Tests. Es wurden Trainings für die UC-Owner durchgeführt. Auch ein Dry Run für den UC TIER1 wurde von Lösch und Partner komplett vorbereitet und

durchgeführt. Dieser diente als Basis für den zweiten "offiziellen" Dry Run durch den UC-Owner Schäffler.

TIER1 TEST-SET: CA. 800 ANFORDERUNGEN

S emantische Ähnlichkeitsanalyse

Mögliche Anfragen rum um das Thema "actuator":

- Which parts (*characteristics, elements, mechanical properties*) shall the actuator have?
- Which requirements concern possible damages?
- Which requirements deal with Local Interconnect Network?
- Which requirements describe the risk of corroding?
- When does the risk of rust arise?
- Which signals does an actuator convert?
- How does the actuator perform the conversion? vs. How does the conversion of the input happen?

Codebeamer Report-ID: 23672449

Abb. 21: Erklärung zur Verwendung der semantischen Ähnlichkeitsanalyse beim ersten Dry Run für TIER1

Im Rahmen der Test-Sessions durch die UC-Owner (Audio, Frontend, Türen, Licht, TIER1) traten wiederholt unerwartete Fehlermeldungen auf, die dann bei Ad-Hoc-Terminen mit den UC-Ownern, Lösch&Partner und dem KIZAM-Use Case analysiert und behoben wurden. Auch im Falle einer fehlerhaften Bedienung von Codebeamer durch die UC-Owner hat Lösch&Partner Abhilfe geleistet.

Erstellung von Fehleranalysen

Die Ergebnisse der selbstständigen Tests der Funktionalitäten vom KIZAM-Use Case FP wurden im Detail analysiert und in mehreren Chargen zur weiteren Analyse an das

Forschungsinstitut SE weitergegeben. Es wurde versucht, Muster für fehlerhafte Ergebnisse zu erkennen. Diese wurden auch mit dem SE angerissen und besprochen.

Diese Analysen betrafen mehrere Aspekte: Qualität und Form der Varianten in der Formalisierung, Konsistenz der Formalisierung innerhalb eines Sets (Report-ID), Festlegung der Kriterien für Redundanz und (In)konsistenz und deren Erkennbarkeit durch die KIZAM-Use Case FP -Pipeline (Solver), Parsing-Quote, Erkennungsquote für Redundanzen und Inkonsistenzen. Letztlich lieferte das SE selbst eine Analyse der Ergebnisse aus den Dry Runs und erkannte als Verbesserungsmaßnahme die Notwendigkeit, Änderungen an der Parsing-Grammatik vorzunehmen, damit diese an mehreren Stellen - z. B. bei den Steuerzeichen, den Anführungszeichen, den unterschiedlichen Ausprägungen von Parametern - mehr "Schreibvarianten" zulässt (Stichwort: Relaxte Grammatik) und die Anzahl der korrekt geparsten und somit formalisierten Anforderungen erhöht.

Das Testing der Relaxten Grammatik wurde von Lösch und Partner übernommen. Es wurden mehrere Reports für die Use Cases Frontend, Türen, Licht, TIER1 mit einer unterschiedlichen Anzahl von Item-IDs erstellt, die alle möglichen schwierigen und fehleranfälligen Fälle umfassten, bei denen die ursprüngliche Parsing-Grammatik fehlgeschlagen war. Die Reports wurden zuerst vollständig mit der ursprünglichen Grammatik formalisiert und sowohl die erfolgreichen als auch die fehlerhaften Formalisierungen mit ihren Fehlermeldungen festgehalten. Im Nachgang wurde die neue Relaxte Grammatik an das Widget angeschlossen und dieselben Report wurde alle mit der Relaxten Grammatik formalisiert. Auch für die Relaxte Grammatik wurden die Fehlermeldungen festgehalten. Die gesamte Auswertung bzw. Gegenüberstellung der Fehlermeldungen für die alte bzw. Relaxte Grammatik wurden im Juli zur Analyse und Fehlerbehebung an das SE übergeben.

1	A	B	C	D	E	F
Use Case	Report-ID	Standard-Grammatik				
		Item-ID	Details	Parsable	Error message	
3	TUEREN	24079841	Natural language: Components shall be flame resistant. Current formalization: Globally, it is always the case that components.flame_resistant equals true.	true		
4	TUEREN	24079841				
5	TUEREN	24079841				
6	TUEREN	24079841				
7	TUEREN	24079841	Natural language: Adjacent components must not be damaged at appropriate disassembly. Current formalization: Globally, if disassembly.appropriate equals true, then in response adjacent_components.damaged equals false.	true +		
8	TUEREN	24079841				
9	TUEREN	24079841				
10	TUEREN	24079841				
11	TUEREN	24079841	Natural language: During the flame exposure within 15, the burning distance must not be longer than 38. Current formalization: Globally, if flame_exposure_time less or equal than 15, then in response burning_distance less or equal than 38 within 15 minutes.	false	Message from TLA: At least one structu english requirement could not be pars Requirement 3345841: [ERROR] tempFile11417306011721164942.ser 22>: no viable alternative at input 'if flame_exposure_time less or equal th then in response burning_distance les equal than 38 within 15 minutes' in ru stack: [Specification, Property, Pattern	
12	TUEREN	24079841				
13	TUEREN	24079841				
14	TUEREN	24079841				
15	TUEREN	24079841	Natural language: Adjacent component must not be damaged at disassembly.	true		
16	TUEREN	24079841				
17	TUEREN	24079841				
18	TUEREN	24079841				
19	TUEREN	24079841	Natural language: Adjacent component must not be damaged at disassembly.	true		
20	TUEREN	24079841				
21	TUEREN	24079841				
22	TUEREN	24079841				
23	TUEREN	24079841	Natural language: Adjacent component must not be damaged at disassembly.	true		
24	TUEREN	24079841				

Abb. 21: Gegenüberstellung der Ergebnisse „Standard Grammatik“ vs. „Relaxte Grammatik“

Im Nachgang wurde – unabhängig von der verwendeten Grammatik – eine systematische konsolidierte Auswertung und Kategorisierung der menschlich erkennbaren Fehler in der Formalisierung sowie im fehlenden Erkennen von Redundanzen bzw. Inkonsistenzen in den einzelnen Reports durchgeführt. Verwendet wurden hierbei zahlreiche reale Anforderungen aus den Dry Runs für alle Use Cases sowie aus den Experteninterviews. Wiederkehrende Fehlermuster und mögliche Ursachen wurden im Vorfeld der gemeinsamen Diskussion erkannt und festgehalten. Diese Fehlerkategorisierung wurde schließlich mit dem SE besprochen.

Ergebnis	Item-ID	Anforderung/Report-ID	Form
			25758667
redundanz nicht erkannt	3420300	Soldering pads and wetttable surfaces shall be separated with solder resist.	Globalt
		The soldering pads shall be isolated from other wetttable surfaces using solder resist.	Globalt solderin
	3420281		

Abb. 22: Beispiel aus der konsolidierten Auswertung der Funktionalitäten vom KIZAM-Use Case FP

Die Auswertung und die daraus gewonnenen Erkenntnisse sollen im Hinblick auf das Nachfolgeprojekt KIMBA als Basis für eine systematische Ergebnisoptimierung und die Erarbeitung alternativer technischer Lösungsansätze dienen. Als Verbesserungspotential wurde ein geeignetes Preprocessing als Vorstufe zur Formalisierung und Auswertung via Solver identifiziert.

Gefördert durch:



**Bundesministerium
für Wirtschaft
und Klimaschutz**

**aufgrund eines Beschlusses
des Deutschen Bundestages**

Berichtsblatt

1. ISBN oder ISSN [Nicht zutreffend]	2. Berichtsart (Schlussbericht oder Veröffentlichung) Schlussbericht
3. Titel Erfolgskontrollbericht KIZAM – Lösch & Partner	
4. Autor(en) [Name(n), Vorname(n)] Scheuerpflug, Gabriel Colombo, Roberta Dudok, Nick	5. Abschlussdatum des Vorhabens 30. September 2024
	6. Veröffentlichungsdatum [Nicht zutreffend]
	7. Form der Publikation [Nicht zutreffend]
8. Durchführende Institution(en) (Name, Adresse) Lösch & Partner GmbH Ridlerstr. 33 80339 München	9. Ber. Nr. Durchführende Institution
	10. Förderkennzeichen 19I21029B
	11. Seitenzahl 34
12. Fördernde Institution (Name, Adresse) Bundesministerium für Wirtschaft und Klimaschutz (BMWK) 53107 Bonn	13. Literaturangaben
	14. Tabellen
	15. Abbildungen
16. Zusätzliche Angaben	
17. Vorgelegt bei (Titel, Ort, Datum)	

18. Kurzfassung

Beitrag zu förderpolitischen Zielen: Lösch & Partner hat maßgeblich dazu beigetragen, ein gemeinsames Verständnis des Anforderungsmanagementsprozesses zu schaffen. Durch die Koordination und das Alignment der verschiedenen Projektparteien wurde ein einheitlicher Ansatz erarbeitet, der die Grundlage für die erfolgreiche Umsetzung des Projekts bildete. Ein Workshop im vierten Quartal 2021 diente dazu, den Anforderungsprozess zu erklären und zu vereinheitlichen. Dies ermöglichte es, die Potenziale der Künstlichen Intelligenz (KI) im Entwicklungsprozess zu identifizieren und den gesamthaften Business Value besser abzuschätzen.

Wissenschaftlich-technisches Ergebnis: Lösch & Partner hat mehr als 25 Jahre Erfahrung und Praxiswissen in das KIZAM Forschungsprojekt eingebracht. Das Hauptaugenmerk zu Beginn lag in der Koordination und im Alignment der verschiedenen Projektparteien. Ein gemeinsames Verständnis, speziell in der Industrialisierung des Forschungsbeitrags, musste kreiert werden. Dazu wurde im vierten Quartal 2021 ein gemeinsamer Workshop gehalten, in dem der Anforderungsprozess erklärt wurde. Der Lebenszyklus einer Anforderung wurde detailliert dargestellt. Die Zusammenarbeit mit der RWTH Aachen und die Entwicklung eines speziellen Widgets zur Testung der technischen Lösungsansätze waren wesentliche Schritte des Projekts.

Fortschreibung des Verwertungsplans: Im Rahmen des KIZAM-Projekts sind keine Schutzrechtsanmeldungen oder Erfindungen aufgekommen. Die wirtschaftlichen Erfolgsaussichten nach Projektende müssen unter der rasanten Entwicklung im Bereich der KI und GenAI stark beobachtet werden. Ein Folgeprojekt namens KIMBA wurde ins Leben gerufen, in dem weitere starke Industriepartner in die Entwicklungen involviert sind.

Arbeiten, die zu keiner Lösung geführt haben: Das Änderungsmanagement und das Lastenheftmanagement konnten nicht vollständig automatisiert werden. Die Integration von Relyze in Codebeamer stellte technische Herausforderungen dar, die nicht vollständig gelöst werden konnten. Trotz intensiver Tests und Optimierungen der Parsing-Grammatik traten wiederholt Fehler auf.

Präsentationsmöglichkeiten: Im Zuge des Folgeprojekts KIMBA sind Messeauftritte geplant, bei denen die Ergebnisse und Fortschritte des KIZAM-Projekts präsentiert werden. Interessierte können sich jederzeit gerne mit uns in Verbindung setzen, um mehr über das KIZAM-Projekt zu erfahren.

Einhaltung der Kosten- und Zeitplanung: Im Rahmen des KIZAM-Projekts wurden die Kosten- und Zeitplanung weitgehend eingehalten. Eine Budgeterhöhung wurde beantragt und genehmigt, um vorher unbekannte Umfänge der Widgetentwicklung zu übernehmen. Das Gesamtprojekt wurde kostenneutral um 3 Monate verlängert. Im Gesamtbudget wurden etwa 10% mehr Mittel benötigt.

19. Schlagwörter

Anforderungsmanagement, Künstliche Intelligenz, Generative KI, Industrialisierung, Forschung, Entwicklung, KIZAM, KIMBA, Lösch & Partner

20. Verlag

[Nicht zutreffend]

21. Preis

[Nicht zutreffend]

Document Control Sheet

1. ISBN or ISSN Not applicable	2. type of document (e.g. report, publication) report
3. title Success Control Report KIZAM – Lösch & Partner	
4. author(s) (family name, first name(s)) Scheuerpflug, Gabriel; Colombo, Roberta; Dudok, Nick	5. end of project 30. September 2024
	6. publication date Not applicable
	7. form of publication Not applicable
8. performing organization(s) (name, address) Lösch & Partner GmbH, Ridlerstr. 33, 80339 Munich	9. originator's report no. Not applicable
	10. reference no. 19I21029B
	11. no. of pages 34
12. sponsoring agency (name, address) Bundesministerium für Wirtschaft und Klimaschutz (BMWK) 53107 Bonn	13. no. of references
	14. no. of tables
	15. no. of figures
16. supplementary notes	
17. presented at (title, place, date)	
18. abstract <p>Contribution to funding policy goals: Lösch & Partner significantly contributed to creating a common understanding of the requirements management process. Through coordination and alignment of the various project parties, a unified approach was developed, forming the basis for the successful implementation of the project. A workshop in the fourth quarter of 2021 served to explain and standardize the requirements process. This made it possible to identify the potential of artificial intelligence (AI) in the development process and better assess the overall business value.</p> <p>Scientific and technical results: Lösch & Partner brought more than 25 years of experience and practical knowledge to the KIZAM research project. The main focus at the beginning was on coordinating and aligning the various project parties. A common understanding, especially in the industrialization of the research contribution, had to be created. To this end, a joint workshop was held in the fourth quarter of 2021 to explain the requirements process. The lifecycle of a requirement was detailed. Collaboration with RWTH Aachen and the development of a special widget for testing the technical solutions were key steps in the project.</p> <p>Continuation of the utilization plan: No patent applications or inventions emerged in the KIZAM project. The economic prospects after the project must be closely monitored due to the rapid development in the AI and GenAI fields. A follow-up project called KIMBA has been initiated, involving other strong industry partners in the developments.</p> <p>Work that did not lead to a solution: Change management and requirements specification management could not be fully automated. The integration of Relyze into Codebeamer posed technical challenges that could not be fully resolved. Despite intensive testing and optimization of the parsing grammar, errors repeatedly occurred.</p> <p>Presentation opportunities: In the course of the follow-up project KIMBA, trade fair appearances are planned where the results and progress of the KIZAM project will be presented. Interested parties can contact us at any time to learn more about the KIZAM project.</p> <p>Compliance with cost and time planning: The cost and time planning of the KIZAM project was largely adhered to. A budget increase was requested and approved to cover previously unknown scopes of widget development. The overall project was extended by 3 months at no additional cost. Approximately 10% more funds were needed in the overall budget.</p>	
19. keywords Requirements management, Artificial Intelligence, Generative AI, Industrialization, Research, Development, KIZAM, KIMBA, Lösch & Partner	

20. publisher
Not applicable

21. price
Not applicable