Prompt Engineering im Systems Engineering

Potenziale und Grenzen moderner großer Sprachmodelle im V-Modell

Aschot Hovemann,*
Isaac Mpidi Bita,
Abed Alrahman Aldade,
Oliver von Heißen und
Roman Dumitrescu

Große Sprachmodelle wie GPT-4 bieten erhebliche Potenziale für das Systems Engineering. Prompt-Engineering ermöglicht einen flexiblen Einsatz im Anforderungsmanagement, Systementwurf und in der Integration, Verifikation und Validierung ohne aufwendiges Modelltraining. Die Formulierung von Prompts und die Anwendung fortschrittlicher Techniken erfordern jedoch tiefes Domänenwissen. Der Beitrag zeigt Potenziale und Herausforderungen dieser Technik auf und illustriert praktische Anwendungsbeispiele

Einleitung

Die fortschreitende Digitalisierung und der Einsatz von Künstlicher Intelligenz (KI) eröffnen faszinierende Potenziale im Systems Engineering. Große Sprachmodelle (engl. Large Language Models, LLMs) wie GPT-4 von OpenAI können Entwicklungsprozesse bereichern, indem sie generalisiertes Wissen und logische Fähigkeiten bereitstellen. LMs sind auf großen Textkorpora trainierte KI-Modelle, die in der Lage sind, menschenähnliche Sprache zu verstehen, zu generieren und kontextbasiert Antworten auf komplexe Anfragen zu liefern [1]. Sie unterstützen Aufgaben wie die automatisierte Anforderungserstellung oder die Optimierung von Systementwürfen, sind jedoch in ihrer Grundform nicht für domänens pezifische Engineering-Tätigkeiten ausgelegt.

Um die Einsatzmöglichkeiten von KI im Systems Engineering vollständig auszuschöpfen, sind eine Domänenanpassung bzw. das Anreichern mit domänenspezifischem Wissen erforderlich. Hierfür existieren verschiedene Techniken, wie das ressourcenintensive Fine-Tuning, bei dem das zugrunde liegende KI-Modell auf spezifische Daten (weiter-)trainiert wird. Eine schnellere und flexiblere Alternative stellt jedoch das Prompt Engineering dar [2]. Dabei werden präzise formulierte Eingabeaufforderungen (Prompts) verwendet, um das vorhandene Wissen eines Modells gezielt zu aktivieren. Ein Prompt ist eine spezifische Anweisung oder Frage, die das Modell auf eine gewünschte Aufgabe ausrichtet. Durch die Anreicherung der Prompts mit zusätzlichem Wissen bzw. Kontext können auch spezialisierte Aufgaben effektiv gelöst werden – ohne den erheblichen Aufwand einer umfassenden Modellanpassung [3].

Bild 1 zeigt die Phasen des V-Modells und hebt zentrale Anwendungsfelder sowie exemplarische Einsatzpotenziale für generative KI im Systems Engineering hervor. Durch den gezielten Einsatz von LLMs können verschiedene Aufgaben – von der automatisierten Anforderungsextraktion bis hin zur Konsistenzprüfung und Testfallgenerierung – effizient unterstützt werden. Im Gegensatz zum Stand der Technik, der nur elementare Prompts für spezifische Aufgaben empfiehlt [4], geht dieser Beitrag einen Schritt weiter und zeigt Empfehlungen fortschrittlicher Prompt-Engineering-Techniken.

Zunächst werden die theoretischen Grundlagen des Prompt Engineerings sowie Empfehlungen für spezifische Systems-Engineering-Tätigkeiten erläutert. Darauf aufbauend erfolgt eine systematische Analyse praxisnaher Anwendungsbeispiele. Abschließend werden die Einsatzmöglichkeiten und Limitationen bewertet und ein Ausblick auf den zukünftigen Forschungs- und Entwicklungsbedarf gegeben.

* **Korrespondenzautor**
Aschot Hovemann, M. Sc.; Fraunhofer-Institut für Entwurfstechnik Mechatronik IEM; Zukunftsmeile 1, 33102 Paderborn; Tel.: +49 (0) 5251 5465-447, E-Mail: aschot.hovemann@iem.fraunhofer.de

Weitere Autoren
Isaac Mpidi Bita, M. Sc.; Fraunhofer IEM, Paderborn
Abed Alrahman Aldade, M. Sc.; Fraunhofer IEM, Paderborn
Oliver von Heißen, M. Sc.; Fraunhofer IEM, Paderborn
Prof. Dr.-Ing. Roman Dumitrescu; IEM & Universität Paderborn

Hinweis
Bei diesem Beitrag handelt es sich um einen von den Advisory-Board-Mitgliedern des ZWF-Sonderheftes wissenschaftlich begutachteten Fachaufsatz (Peer-Review).

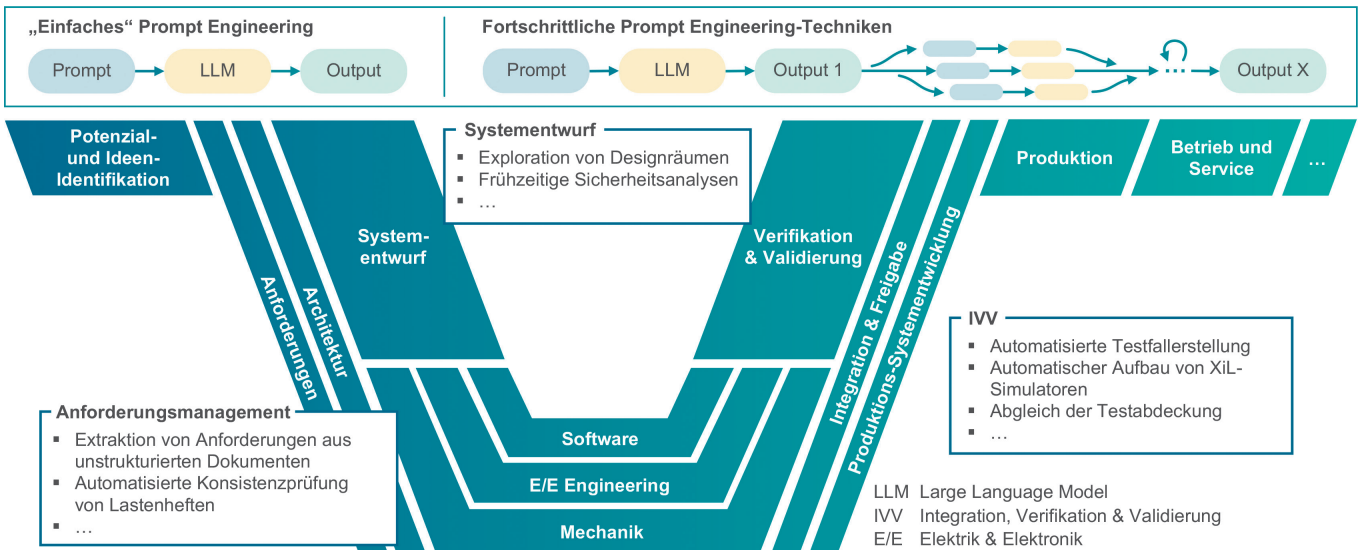

Bild 1. Einsatzpotenziale von Prompt Engineering im V-Modell

Forschrittliches Prompt Engineering

Prompt Engineering ist eine aufstrebende Disziplin, die sich mit der Entwicklung und Optimierung von Prompts befasst. Ziel ist es, die generischen Fähigkeiten großer vortrainierter KI-Modelle (insbesondere LLMs) für spezifische Aufgaben nutzbar zu machen. Die Qualität der generierten Ergebnisse hängt maßgeblich von der präzisen Formulierung sowie einer sinnvollen Kombination bzw. Verschachtelung der Prompts ab [5]. Um diesen Nutzen zu maximieren, wurde eine Vielzahl von Techniken entwickelt, welche komplexere Schemata bzw. Regeln vorgeben. Die in Tabelle 1 dargestellte Kategorisierung strukturiert etablierte Techniken in Anlehnung an [6–8]. Im Folgenden werden ausgewählte Techniken vorgestellt und abgegrenzt. Eine detaillierte Beschreibung kann den zugrundeliegenden Quellen entnommen werden.

Zu den grundlegenden Methoden gehören Zero-Shot- und Few-Shot-Prompting. Beim Zero-Shot-Prompting löst das KI-Modell Aufgaben, für die es nicht explizit trainiert wurde, ohne Beispiele. Beim Few-Shot-Prompting werden dem Modell einige Beispiele präsentiert, um es zur gewünschten Ausgabe zu führen [7]. Fortgeschrittene Ansätze wie Chain-of-Thoughts (CoT) unterstützen die Darstellung eines mensch-analogen Denkprozesses durch das Systematisieren von Zwischenschritten. Dadurch kann das Modell komplexe Probleme schrittweise analysieren und fundierte Lösungen erarbeiten [8]. Aufbauend auf der Zerlegung von Aufgaben in Teilaufgaben wurden Methoden wie Logical-Chain-of-Thoughts (LogiCoT) und Chain-of-Code (CoC) entwickelt. LogiCoT

Kategorie	Techniken
Grundlegende Prompting-Techniken	Zero-Shot (**ZSP**) [7] Few-Shot (**FSP**) [7]
Problemzerlegung und strukturiertes Denken	Chain-of-Thought (**CoT**) [8] Logical-Chain-of-Thought (**LogiCoT**) [7] Structured-Chain-of-Thought (**SCoT**) [8] Least-to-Most (**LtM**) [7] Take-a-Step-Back (**TaSB**) [6]
Optimierung und iterative Verfeinerung	Optimization-by-Prompting (**OPRO**) [6] Self-Consistency (**SC**) [8] Active-Prompting (**AP**) [8] Contrastive Chain-of-Thought (**CCoT**) [6]
Datenintegration und Wissensanreicherung	Retrieval-Augmented-Generation (**RAG**) [7] Chain-of-Knowledge (**CoK**) [8] Chain-of-Table-Prompting (**CoTAP**) [8] Chain-of-Note (**CoN**) [6]
Verifizierung, Validierung und Fehlerreduzierung	Chain-of-Verification (**CoVe**) [8]
Automatisierung und Effizienzsteigerung	Automatic-Prompt-Engineer (**APE**) [7] Automatic-Chain-of-Thought (**Auto-CoT**) [7]
Optimierung der Kommunikation und interaktive Problemlösung	Rephrase-and-Respond (**RaR**) [6] Emotion-Prompting (**EP**) [6] Reasoning-and-Acting (**ReAct**) [7] System-2-Attention (**S2A**) [8]
Computergestützte Modellierung und Simulation	Program-of-Thoughts (**PoT**) [8] Scratchpad-Prompting (**SP**) [6] Chain-of-Code (**CoC**) [7]
Strategische Planung und Entscheidungsfindung	Tree-of-Thoughts (**ToT**) [8] Graph-of-Thoughts (**GoT**) [8]

Tabelle 1. Kategorisierung von Prompt-Engineering-Techniken (i. A. an [6–8])

Phase	Exemplarische Aufgabe	Empfohlene Techniken
Anforderungs-management	Stakeholderanalyse	Empfohlen: ThoT, AP Alternativen: CoK, S2A, RaR
	Systemkontextanalyse	Empfohlen: GoT, CoT, Alternativen: CoK, SC, AP
	Systemanforderungen definieren	Empfohlen: LtM, SCoT Alternativen: CoT, CoC, CoK
	Anforderungsvalidierung	Empfohlen: CoVe, SC Alternativen: S2A, CCoT, SP
Systementwurf	Funktionsmodellierung	Empfohlen: PoT Alternativen: SCoT, GoT, SP, CoC
	Systemarchitektur entwickeln	Empfohlen: GoT Alternativen: ToT, CoK, AP, SCoT
	Alternative Lösungsentwürfe entwickeln	Empfohlen: ToT, CCoT Alternativen: OPRO, GoT, CoT
	Architekturanalyse	Empfohlen: CoVe, CCoT Alternativen: PoT, SC, AP
Integration, Verifikation und Validierung	Integrationsstrategie entwickeln	Empfohlen: GoT, ToT Alternativen: AP, LtM, SCoT
	V&V-Planung und Testfallableitung	Empfohlen: CoVe Alternativen: LtM, CoC, SC, SP
	Auswertung der V&V-Ergebnisse	Empfohlen: CCoT, SC Alternativen: CoVe, AP, SP
	Dokumentation der V&V-Ergebnisse	Empfohlen: CoTAP Alternativen: RaR, CoK, SC, S2A

Tabelle 2. Empfohlene Prompt Engineering-Techniken für ausgewählte SE-Aktivitäten

erweitert die Denkprozesse um formale logische Schlüsse, während CoC das Modell befähigt, eigenen Code zur Lösung von Aufgaben zu generieren und auszuführen [7]. Große Sprachmodelle neigen zu Halluzinationen und erzeugen inkorrekte oder erfundene Informationen. Retrieval-Augmented-Generation (RAG) adressiert dieses Problem, indem relevante externe, verifizierte Wissensquellen in die Eingabeaufforderung integriert werden [7]. In diesem Zusammenhang verbessern Chain-of-Table-Prompting (CoTAP) und Chain-of-Knowledge (CoK) die Genauigkeit der Antworten durch eine erweiterte Integration und Strukturierung des Wissens [8]. Zur Minimierung von Fehlerpotenzial verifizieren Methoden wie Chain-of-Verification (CoVe) den Lösungsprozess sukzessive und erhöhen so die Zuverlässigkeit der Ergebnisse. Techniken wie CoT erfordern oft einen hohen manuellen Aufwand bei der Erstellung von strukturierten Prompts und Beispielen. Um den Prozess zu automatisieren und den Aufwand zu reduzieren, wurden Automatic Chain-of-Thoughts (Auto-CoT) und Automatic Prompt Engineer (APE) entwickelt. Auto-CoT generiert automatisch aufeinander aufsetzende Lösungsbeispiele, wodurch die manuelle Erstellung entfällt. APE erzeugt iterativ Prompts, testet diese und wählt die leistungsfähigsten aus. Methoden wie Tree-of-Thoughts (ToT) und Graph-of-Thoughts (GoT) fördern die Entscheidungsfähigkeit, indem komplexe Probleme durch das parallele Verfolgen verschiedener Denkpfade analysiert werden [8].

Empfohlene Techniken im Systems Engineering

Die ISO/IEC/IEEE 15288 definiert die Prozesse und Aktivitäten, die im Rahmen des Systems Engineering für die Entwicklung komplexer technischer Systeme durchzuführen sind [9]. Im Rahmen dieses Beitrags wurde eine Teilmenge der Aktivitäten analysiert, um Empfehlungen zur Anwendung fortschrittlicher Prompt-Engineering-Techniken zu geben. Die in Tabelle 2 aufgeführten Techniken bieten nur eine grobe Orientierung, da die erfolgreiche Anwendung von vielen Faktoren wie dem konkreten Input und der genauen Umsetzung einer Technik abhängt.

Für das *Anforderungsmanagement*, insbesondere für die Stakeholderanalyse, werden die Techniken ToT und AP empfohlen. Die ToT-Technik fördert das Management der verschiedenen Inputs der Stakeholder und sorgt für ein kohärentes Verständnis. Dies ist besonders wichtig für kollaborative und kreative Aktivitäten. Die AP-Technik fördert die interaktive Kommunikation. Sie ermöglicht es, unterschiedliche Stakeholderperspektiven einzubeziehen und bestehende Unsicherheiten zu klären. Für die Systemkontextanalyse wird die CoT-Technik empfohlen. Sie unterstützt eine schrittweise und strukturierte Analyse der Kontextinformationen, was zu einem umfassenden Verständnis des Systemumfelds führt.

Im *Systementwurf* dominieren kreative und analytisch-logische Tätigkeiten. Für den Entwurf der Systemarchitektur wird die GoT-Technik empfohlen, da sie die Analyse komplexer Beziehungen zwischen Systemkomponenten fördert und sich auch für kreative und konzeptionelle Aufgaben eignet. Eine weitere Schlüsselaktivität ist die Entwicklung alternativer Lösungsentwürfe. Diese Tätigkeit zeichnet sich durch kreatives und gleichzeitig systematisches Denken aus. Die ToT-Technik unterstützt die systematische Untersuchung verschiedener Lösungsansätze und fördert Kreativität und Innovation im Denkprozess. Die zweite Empfehlung für diese Aktivität ist die CCoT-Technik. Sie ermöglicht einen direkten Vergleich zwischen verschiedenen Lösungsentwürfen.

Im Bereich *Integration, Verifikation und Validierung* ist die Ableitung von Testfällen eine zentrale Tätigkeit. Diese Aufgabe erfordert sowohl Kreativität als auch detailliertes analytisches Denken. Als geeignete Methode wird die CoVe-Technik empfohlen. Mit diesem Ansatz können Verifikations- und Validierungsinhalte systematisch erstellt werden und es wird sichergestellt, dass alle Anforderungen durch entsprechende Testfälle abgedeckt werden. Für die Auswertung der Testergebnisse werden zwei Techniken vorgeschlagen. Die CCoT-Technik bietet den Vorteil, dass die Testergebnisse direkt miteinander verglichen werden können, um Abweichungen zu identifizieren und

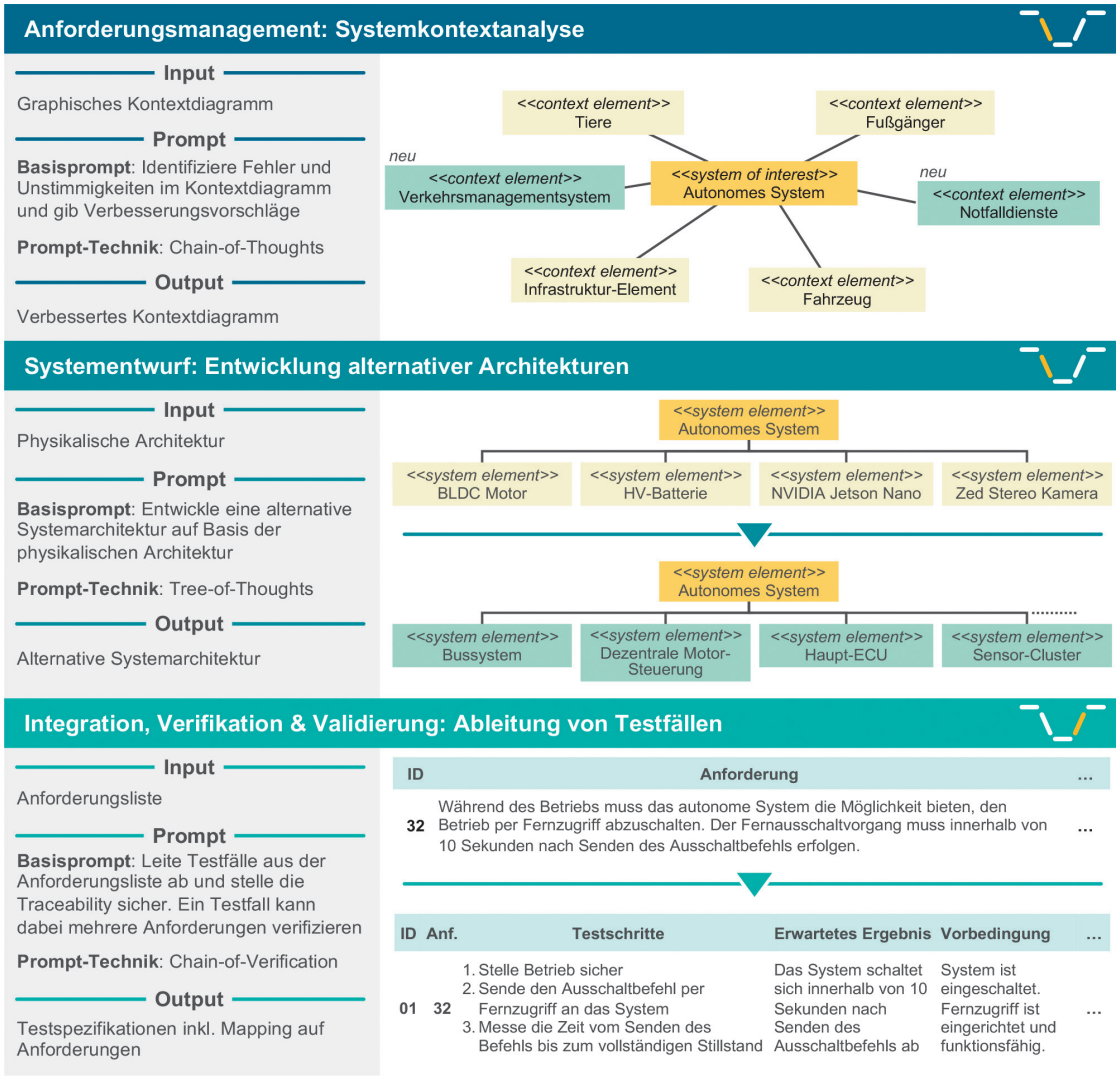

Bild 2. Exemplarische Anwendung von Prompt-Engineering-Techniken

Fehlerquellen aufzudecken. Die SC-Technik wird ebenfalls empfohlen, da sie hilft, die Konsistenz der Auswertungen sicherzustellen und widersprüchliche Interpretationen zu vermeiden.

Exemplarische Anwendung

Um die Einsatzpotenziale von Prompt Engineering im Systems Engineering zu veranschaulichen, werden ausgewählte Aufgaben mit empfohlenen Techniken umgesetzt. Dies erfolgt anhand der in Bild 2 dargestellten konkreten Aktivitäten:

- Verbesserung einer Systemkontextanalyse,
- Entwicklung von Architekturalternativen im Systementwurf und
- Ableitung von Testfällen.

Die *Systemkontextanalyse* umfasst die Definition der Systemgrenze sowie Schnittstellen und Interaktionen zu weiteren Akteuren. Die konkrete Aufgabe besteht darin, ein bestehendes Systemkontextdiagramm eines autonomen Mobilitätssystems, welches Akteure wie „Tiere“ und „Infrastruktur-Elemente“ enthält, mithilfe der Chain-of-Thoughts-Technik zu verbessern. Hierzu werden Unstimmigkeiten identifiziert und Verbesserungsvorschläge erarbeitet. Dabei wird die Aufgabe schrittweise angegangen: Zunächst werden alle Elemente und ihre Beziehungen überprüft, indem Fragen wie „Sind alle relevanten Akteure berücksichtigt?“ oder „Fehlen wichtige Schnittstellen?“ gestellt werden. Als Ergebnis werden fehlende Akteure wie das „Verkehrsmanagementsystem“ oder die „Notfalldienste“ identifiziert.

Eine zentrale Aufgabe im Systementwurf ist die Exploration des Lösungsraums durch die *Entwicklung von Architekturalternativen*. Als Input dient ein Auszug der physischen Architektur eines autonomen Fahrzeuges, bestehend aus Komponenten wie Sensoren und Steuerungseinheiten. Im dargestellten Beispiel wird die Tree-of-Thoughts-Technik eingesetzt. Zunächst werden Hauptideen generiert, etwa „Modulare Sensorintegration“ für Flexibilität, „Zentralisierte Steuerung“ für Effizienz, „Dezentrale Steuerung“ für Ausfallsicherheit und „Kompakte Integration“ für Energieeinsparung. Anschließend wird jede Hauptidee systematisch analysiert, und kritisch bewertet. Die

Phase und Aufgabe	Variante mit eingesetzter Technik	Kohärenz	Klarheit	Vollständigkeit	Innovation	Gesamteindruck
Anforderungsmanagement: Kontextanalyse	Graph-of-Thoughts	3,5	3,5	3,5	2,8	**3,1**
	Chain-of-Thoughts	3,5	4	4,2	3,7	**3,8**
	Self-Consistency	3,5	2,8	1,8	1,9	**2,3**
Systementwurf: Entwicklung alternativer Lösungen	Tree-of-Thoughts	3,5	3,0	3,0	2,8	**3,4**
	Optimization-by-Prompting	3,2	2,8	2,6	2,4	**2,7**
	Graph-of-Thoughts	3,5	3,5	2,3	2,0	**2,5**
IVV: Ableitung von Testfällen	Chain-of-Verification	3,9	4,2	4,3	3,5	**4,2**
	Least-to-Most	3,6	3,2	2,8	2,4	**2,8**
	Self-Consistency	3,6	3,4	3,0	2,6	**3,1**

Optimization-by-Prompting (Auszug)

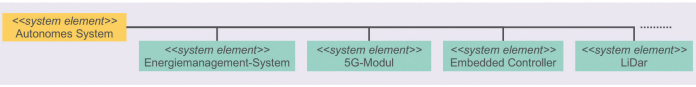

Bild 3. Bewertung der Eignung verschiedener Prompt-Engineering-Techniken

Analyse führt zu einer hybriden Architekturvariante: eine dezentrale Motorsteuerung (Erhöhung der Ausfallsicherheit), kombiniert mit einem modularen Bussystem (einfache Erweiterbarkeit) und einer leistungsfähigen Haupt-ECU für die Bildverarbeitung und weitere hochautomatisierte Fahrfunktionen.

Im Rahmen der IVV wird die *Testfallableitung* exemplarisch KI-gestützt umgesetzt. Eine tabellarische Anforderungsliste dient als Input. Mithilfe der Chain-of-Verification-Technik wird jede Anforderung einzeln und im Gesamtkontext analysiert, um relevante Testkriterien wie zeitliche Abhängigkeiten zu identifizieren. Anschließend werden die Testfälle detailliert formuliert, einschließlich Testprozedur, erwarteter Ergebnisse, Vorbedingungen und der relevanten Anforderungen. Die CoVe-Technik gewährleistet durch ihr schrittweises Vorgehen eine vollständige Abdeckung der Anforderungen und minimiert das Risiko, kritische Aspekte zu übersehen.

Analyse der Eignung und Ausblick

Um die Eignung und Effektivität verschiedener Prompt-Engineering-Techniken im Systems Engineering zu bewerten, wurde eine zusätzliche Vergleichsstudie durchgeführt. Dabei wurden die drei zuvor vorgestellten Aufgaben – Systemkontextanalyse, Entwicklung alternativer Lösungen im Systementwurf und Ableitung von Testfällen – näher untersucht. Wie in Bild 3 dargestellt, kamen pro Aufgabe drei unterschiedliche Prompt-Engineering-Techniken zum Einsatz. Die resultierenden Artefakte wurden von 13 Systems Engineers mit 3 bis 15 Jahren Berufserfahrung anhand von fünf Kriterien bewertet: Kohärenz und Widerspruchsfreiheit (Konsistenz der Ergebnisse), Klarheit und Verständlichkeit (wie klar ist die Darstellung), Vollständigkeit (Abdeckung aller relevanten Aspekte), Innovation (Einbringen neuer oder kreativer Ansätze) und Gesamteindruck (subjektive Gesamtbewertung). Jedes Kriterium wurde auf einer Skala von 1 (niedrig) bis 5 (hoch) bewertet.

Die Ergebnisse zeigen, dass die Effektivität der Techniken je nach Aufgabenstellung stark variiert. Bei der Systemkontextanalyse erzielt Chain-of-Thoughts mit einem Gesamteindruck von 3,8 die höchsten Bewertungen, insbesondere in Klarheit (4,0) und Vollständigkeit (4,2), und übertrifft damit Graph-of-Thoughts (3,1) und Self-Consistency (2,3). In der Entwicklung alternativer Lösungen zeigt Tree-of-Thoughts mit einem Gesamteindruck von 3,4 eine leichte Präferenz gegenüber Optimization-by-Prompting (2,7) und Graph-of-Thoughts (2,5), wobei höhere Werte in Kohärenz und Klarheit erzielt werden. Im direkten Vergleich der Architekturausschnitte von Bild 2 und Bild 3 wird deutlich, dass Tree-of-Thoughts ein insgesamt stimmigeres Bild erzeugt. Die klare Struktur und die Integration innovativer Komponenten, wie z. B. ein Sensorcluster mit Multisensorfusion, heben die Architektur deutlich hervor. Dies ermöglicht eine umfassendere und robustere Wahrnehmung der Umgebung im Vergleich zu einem einfachen LiDAR-System. Hinsichtlich der Vollständigkeit und Kohärenz überzeugt Tree-of-Thoughts durch übergreifende Steuergeräte und eine klar abgestimmte Systemstruktur, die Kommunikations-, Steuerungs- und Wahrnehmungsmodule effizient und skalierbar kombiniert. Bei der Ableitung von Testfällen dominiert Chain-of-Verification mit einem Gesamteindruck von 4,2 und hohen Bewertungen in Kohärenz (3,9), Klarheit (4,2) und Vollständigkeit (4,3), was die Eignung für Aufgaben mit hohen Genauigkeitsanforderungen unterstreicht.

Die Analyse verdeutlicht, dass strukturierte Techniken, die iterative oder rekursive Prozesse sowie explizite Qualitätskriterien einbeziehen, tendenziell bessere Ergebnisse liefern. Dies spiegelt sich in den hohen Bewertungen von Chain-of-Thoughts und Chain-of-Verification wider. Ein eindeutiger Favorit für alle Aufgaben lässt sich jedoch nicht identifizieren, da die Effektivität stark von der spezifischen Aufgabe, dem verfügbaren Input und der Umsetzung abhängt. Die heterogene Natur der Aufgaben im Systems Engineering, die kreative und analytische Elemente umfasst, erschwert eine Generalisierung.

Bemerkenswert ist, dass einige Teilnehmende von der Qualität der Ergebnisse überrascht waren und in Zukunft LLMs mit fortschrittlichem Prompt Engineering einsetzen wollen, da sie darin insbesondere bei „Greenfield"-Aufgaben eine erhebliche Arbeitserleichterung sehen. Die Integration von LLMs in bestehende Werkzeuge erleichtert die Bedienung und wird als besonders förderlich für die Immersion und Usability wahrgenommen [10].

Forschungsbedarf besteht in der spezifischeren Anpassung der Techniken an die Art der Engineering-Aufgabe und die individuellen Rahmenbedingungen. Da die Techniken als Schemata fungieren, die von Anwender:innen unterschiedlich umgesetzt werden, ist es nur begrenzt möglich, ihre Wirksamkeit objektiv zu bewerten oder klare Favoriten zu identifizieren. Daraus ergeben sich drei zentrale Forschungsrichtungen: Erstens die Entwicklung dedizierter Techniken für spezifische Herausforderungen im Engineering. Zweitens die Optimierung durch kontextsensitive Kombination bestehen-

der Techniken, um deren Potenziale gezielt zu nutzen. Drittens die Analyse der Variabilität in der Anwendung, um deren Einfluss auf die Lösungsqualität besser zu verstehen. Eine kombinatorische Betrachtung und Klassifizierung von Engineering-Aufgaben nach ihrem Charakter – kreativ, analytisch oder interaktiv – könnte die systematische Eignungsbeurteilung zusätzlich fördern. Insgesamt zeigt die Analyse, dass fortschrittliche Prompt-Engineering-Techniken bereits heute ohne großen Aufwand erheblichen Mehrwert in Systems-Engineering-Aufgaben bieten können. Die genannten Forschungsbereiche sind jedoch essenziell, um das Potenzial der Techniken langfristig auszuschöpfen.

Literatur

1. Naveed, H.; Khan, A. U.; Qiu, S. et al.: A Comprehensive Overview of Large Language Models. Preprint 2024 DOI:10.48550/arXiv.2307.06435
2. Zhao, T. Z.; Wallace, E.; Feng, S. et al.: Calibrate Before Use: Improving Few-Shot Performance of Language Models. Preprint 2021 DOI:10.48550/arXiv.2102.09690
3. Yao, J.; Xu, W.; Lian, J. et al.: Knowledge Plugins: Enhancing Large Language Models for Domain-Specific Recommendations. Preprint 2023 DOI:10.48550/arXiv.2311.10779
4. Rosenberg, D.; Weilkiens, T.; Moberley, B.: AI Assisted MBSE with SysML: An Integrated Systems/Software Approach. Model Based Systems Engineering 4 You, Victoria, 2024
5. Hackmann, S.; Mahmoudian, H.; Steadman, M.; Schmidt, M.: Word Importance Explains How Prompts Affect Language Model Outputs. Preprint 2024 DOI:10.48550/arXiv.2403.03028
6. Sahoo, P.; Singh, A. K.; Saha, S. et al.: A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. Preprint 2024 DOI:10.48550/arXiv.2402.07927
7. Chen, B.; Zhang, Z.; Langrené, N.; Zhu, S.: Unleashing the Potential of Prompt Engineering in Large Language Models: a Comprehensive Review. Preprint 2023 DOI:10.48550/arXiv.2310.14735
8. Vatsal, S.; Dubey, H.: A Survey of Prompt Engineering Methods in Large Language Models for Different NLP Tasks. Preprint 2024 DOI:10.48550/arXiv.2407.12994
9. ISO/IEC/IEEE 15288:2023: Systems and Software Engineering – System Life Cycle Processes. International Organization for Standardization (ISO), Genf 2023
10. Heissen, O. von; Hanke, F.; Mpidi Bita, I. et al.: Toward Intelligent Generation of System Architectures. In: Malmqvist, J.; Candi, M.; Saemundsson, R. J. et al. (Hrsg.): Proceedings of NordDesign. Glasgow 2024, S. 504–513 DOI:10.35199/NORDDESIGN2024.54

Die Autoren dieses Beitrags

Aschot Hovemann, M. Sc., leitet die Abteilung Digital Engineering am Fraunhofer-Institut für Entwurfstechnik Mechatronik IEM in Paderborn.

Isaac Mpidi Bita, M. Sc., ist wissenschaftlicher Mitarbeiter in der Abteilung Digital Engineering am Fraunhofer IEM in Paderborn.

Abed Alrahman Aldade, M. Sc., ist wissenschaftlicher Mitarbeiter in der Abteilung Digital Engineering am Fraunhofer IEM in Paderborn

Oliver von Heißen, M. Sc., ist wissenschaftlicher Mitarbeiter in der Abteilung Digital Engineering am Fraunhofer IEM in Paderborn.

Prof. Dr.-Ing. Roman Dumitrescu ist seit 2015 ein Direktor des Fraunhofer IEM und leitet den Lehrstuhl für „Advanced Systems Engineering“ an der Universität Paderborn.

Abstract

Prompt Engineering in Systems Engineering: Potentials and Limitations of Modern Large Language Models in the V-Model. Large language models such as GPT-4 offer considerable potential for Systems Engineering. Prompt Engineering enables flexible use in Requirements Management, Systems Design and Integration, Verification, and Validation without time-consuming model training. However, the formulation of prompts and the application of advanced techniques require in-depth domain knowledge. The article highlights the potential and challenges of these techniques and illustrates practical application examples.

Schlüsselwörter

Prompt Engineering, Large Language Model, Generative KI, Systems Engineering, V-Modell

Keywords

Prompt Engineering, Large Language Model, Generative AI, Systems Engineering, V-Model

Bibliography

DOI:10.1515/zwf-2024-0139
ZWF 120 (2025) Special Issue; page 101 – 106

ISSN 0947-0085 · e-ISSN 2511-0896