ChatPLC – Potenziale der Generativen KI für die Steuerungsentwicklung

*Dennis Reinhardt, Jörg Jeschin, Jürgen Jasperneite und Gesa Benndorf**

In diesem Beitrag werden Ansätze untersucht, um ProgrammiererInnen von speicherprogrammierbaren Steuerungen (SPS) entlang des gesamten Softwareentwicklungszyklus durch den Einsatz von generativer künstlicher Intelligenz (KI) zu unterstützen. Konkret wird dabei auf die Umsetzung eines lokalen Sprachmodells für die Generierung von strukturiertem Text (ST) nach IEC 61131-3 und eines Retrieval Augmented Generation-(RAG)-Systeme als Assistent für die Engineering-Software von SPS eingegangen. Diese Beispiele zeigen Potenziale auf, die durch den Einsatz von Sprachmodellen im Umfeld der SPS-Programmierung bereits jetzt greifbar sind.

Einleitung

Die rasante Entwicklung der generativen KI und die damit einhergehenden Veränderungen in Wirtschaft und Gesellschaft durch den Einsatz von großen Sprachmodellen, bzw. Large Language Models (LLM) haben weitreichende Auswirkungen. Insbesondere moderne LLM wie ChatGPT [1], BERT [2], LLaMA [3, 4] und Gemini [5] sind zu zentralen Akteuren in der digitalen Transformation geworden. Mit ihren Fähigkeiten, natürliche Sprache zu verarbeiten, bieten sie das Potenzial, Geschäftsprozesse durch die Analyse umfangreicher Datenmengen zu optimieren. Außerdem können sie die Schaffung personalisierter Kundenerlebnisse und die Automatisierung von Routineaufgaben grundlegend verändern [6]. Diese Potenziale sind insbesondere angesichts aktueller globaler Herausforderungen wie dem Fachkräftemangel und der Deglobalisierung von Bedeutung, da sie die Chance bieten, die Wettbewerbsfähigkeit und technologische Souveränität Deutschlands zu stärken [7]. Ein zentrales Standbein für die technologische Souveränität Deutschlands im Zuge der Industrie 4.0 bildet die Automatisierungstechnik. Sie wird als Integrationswissenschaft verstanden und fungiert als Brücke zwischen den Bereichen Maschinenbau, Elektrotechnik und Informatik [8]. Eine Herausforderung in der Automatisierungstechnik sind manuelle Engineering-Tätigkeiten, die im Bereich der Softwareentwicklung mehr als 10 Prozent der Gesamtkosten für die Planung und Realisierung von Produktionssystemen ausmachen [9]. Um die Effizienz des Entwicklungsprozesses zu steigern, können LLM eine entscheidende Rolle spielen. Die Grundidee, den Softwareprozess durch LLM zu unterstützen, wurde in aktuellen Arbeiten für Teilgebiete der IT bereits untersucht [10, 11], jedoch noch nicht umfassend für die Automatisierungstechnik betrachtet. Es ist daher entscheidend, die tatsächlichen Potenziale und die praktischen Anwendungsmöglichkeiten dieser Technologien im Kontext der Automatisierungstechnik zu ermitteln.

In diesem Beitrag soll daher das Potenzial von LLM innerhalb des Felds der Automatisierungstechnik mit einem Fokus auf die SPS-Programmierung betrachtet werden. Im ersten Schritt erfolgt dazu eine Analyse des gesamten Softwareentwicklungszyklus. Anschließend werden

* **Korrespondenzautorin**
Dr. Gesa Benndorf; Fraunhofer IOSB-INA; Campusallee 1, 32657 Lemgo;
Tel.: +49 (0) 5261 942-9048, E-Mail: gesa.benndorf@iosb-ina.fraunhofer.de

Weitere Autoren
Dennis Reinhardt, B. Sc.; Universität Bielefeld
Jörg Jeschin; Phoenix Contact GmbH & Co KG, Blomberg
Prof. Dr.-Ing. Jürgen Jasperneite; Fraunhofer IOSB-INA

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

zwei konkrete Umsetzungsbeispiele illustriert und schließlich ein Ausblick auf zukünftige Entwicklungen gegeben.

Generative KI im Softwareentwicklungszyklus

Im vorliegenden Abschnitt wird der gesamte Software-Lebenszyklus aus der Perspektive der Automatisierungstechnik betrachtet und auf Arbeitsschritte hin analysiert, die durch den Einsatz von LLM eine Steigerung hinsichtlich Effektivität, Effizienz und Qualität erfahren können. Die gewonnenen Erkenntnisse sind in Bild 1 zusammengefasst und illustrieren die potentiellen Anwendungsfälle von LLM in den verschiedenen Phasen des Lebenszyklus. Aus der grafischen Darstellung geht hervor, dass sich die Anwendungspotenziale von LLM grundsätzlich in zwei Kategorien einteilen lassen: den generativen und den unterstützenden Bereich (Support). Der generative Bereich umfasst sämtliche Applikationen von LLM, die auf Basis vorhandenen Kontextes neue Inhalte generieren. Dagegen bezieht sich der unterstützende Bereich auf die Aufbereitung existierender Wissensbestände in einer interaktiven, multimodalen Form (z. B. durch Texte, Bilder oder Audio-/Video-Inhalte), was die Zugänglichkeit für Nutzer verbessert.

Im Folgenden werden einzelne Anwendungsbereiche kurz exemplarisch dargestellt, eine tiefergehende Erläuterung erfolgt für die beiden in Bild 1 hervorgehobenen Anwendungsfälle in den nächsten beiden Kapiteln.

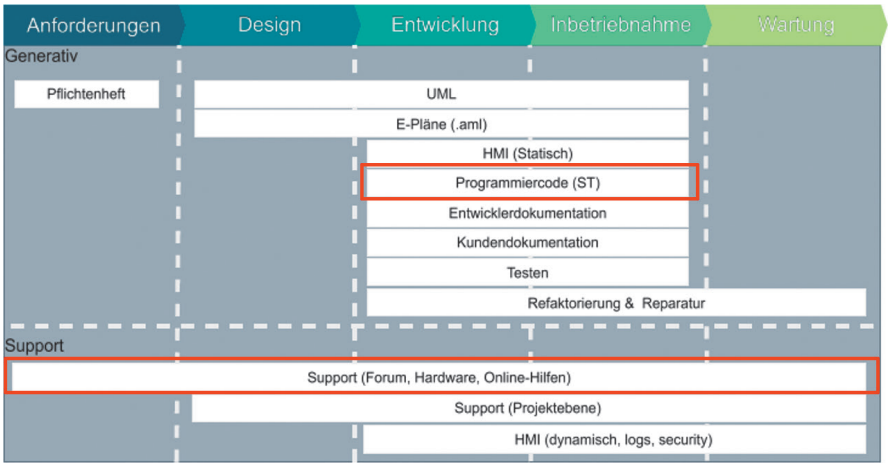

Bild 1. Anwendungspotenziale von LLM entlang des Software-Lebenszyklus von der Anforderungsdefinition bis zum Betrieb. Es wird zwischen dem generativen Bereich (oben) und dem unterstützenden Bereich (unten) unterschieden, je nachdem ob neue Inhalte generiert oder bestehende Inhalte kontextspezifisch aufbereitet werden. Die beiden hervorgehobenen Anwendungsfälle werden in den folgenden Kapiteln näher erläutert

Anforderungsanalyse

Die Anforderungsanalyse ist ein zentrales Element im Prozess der Softwareentwicklung. Sie dient als essenzielles Verbindungsglied zwischen den technischen Spezifikationen der von Entwicklern konzipierten Systeme und den Anwendungszwecken dieser Systeme. In diesem Kontext wird, basierend auf Lastenheften und ergänzt durch spezifisches Wissen aus dem Bereich der Automatisierungstechnik, ein Pflichtenheft erstellt. Dieses Pflichtenheft enthält beispielsweise Unified Modeling Language (UML)-Diagramme mit funktionalen und nicht-funktionalen Anforderungen. Es dient als Basis für die Erzeugung diverser Softwareartefakte wie Programmcode und Testfälle. Da die meisten Pflichtenhefte in natürlicher Sprache geschrieben sind und Grafiken enthalten, besteht das Ziel der Integration von LLM darin, die Fähigkeiten der Textanalyse und -erstellung zu nutzen, um die Qualität der Anforderungsdokumente zu steigern und den Entwicklungsprozess effizienter zu gestalten. Unter dem Begriff Natural Language Processing for Requirements Engineering (NLP4RE) wird der Einsatz von verschiedenen Modellen erforscht [12]. Trotz der multiplen Anwendungsmöglichkeiten von NLP4RE ist die Forschung in diesem Bereich noch nicht weit fortgeschritten [13]. Zum jetzigen Zeitpunkt gibt es Potenzialanalysen [14, 15], welche darauf hindeuten, dass die von ChatGPT generierten Anforderungen vielversprechende Ansätze bieten. Grundlegende Qualitätsmerkmale wie Verständlichkeit, Konsistenz und Korrektheit wurden gezeigt. Jedoch deuten die Studien auch auf Ambiguität in der Formulierung der Anforderungen und potentielle Herausforderungen in der praktischen Implementierung hin.

Elektroplanung

Elektropläne (E-Pläne) und Rohrleitungs- und Instrumenten (R&I)-Fließschemata stellen zentrale Dokumente in der Designphase des Software-Lebenszyklus dar, denn in ihnen werden die verschiedenen Engineering-Disziplinen IT, Elektrotechnik und Maschinenbau zusammengeführt und in Einklang gebracht. Ein E-Plan umfasst die elektrische Verkabelung sowie das Kommunikationssystem einschließlich der zugehörigen Hardware-Komponenten. Zudem werden entsprechende Signallisten für die Steuerungstechnik erzeugt. Um eine Kompatibilität mit verschiedenen Engineering-Tools im Fertigungsbereich zu ermöglichen, wurde der Standard AutomationML entwickelt, welcher auf einem Extensible Markup Language (XML)-Datenformat beruht und Informationen gemäß dem objektorientierten Design anordnet [16].

In der Literatur existieren verschiedene Ansätze, Graphen und Schemata automatisiert zu erzeugen [17, 18]. Die automatische Erstellung von E-Plänen in der Automatisierungstechnik ist durch diese Vorarbeiten und die aktuellen Entwicklungen im Feld der multimodalen LLM greifbarer geworden, jedoch aktuell noch relativ weit von der praktischen Anwendung entfernt. Dies liegt nicht zuletzt daran, dass für die Implementierung dieses Ansatzes zunächst die Erstellung umfassender Datensätze erforderlich ist, die E-Pläne, technische Spezifikationen, Hardwaredokumentationen und Normen beinhalten. Weiterhin ist zu beachten, dass die automatische Erstellung von E-Plänen durch LLM eine sorgfältige Abwägung von Faktoren wie Genauigkeit, Generalisierbarkeit und Anpassungsfähigkeit an spezifische Projektanforderungen erfor-

dert. Darüber hinaus stellt die Sicherstellung der Einhaltung von Industriestandards und Normen eine weitere Herausforderung dar. Angesichts dessen befindet sich die Forschung und Entwicklung in diesem Bereich noch in den Anfangsstadien und bedarf weiterführender Studien und experimenteller Validierungen.

Human-Machine Interface (HMI)

HMIs dienen als Schnittstellen zwischen Mensch und Maschine in der Automatisierungstechnik und ermöglichen es den Werkern, Maschinen zu steuern, Prozessinformationen zu überwachen und auf Systemereignisse zu reagieren. Zur Erstellung wird zuerst eine Designphase durchlaufen. Anschließend wird dieses Design durch verschiedene Editoren umgesetzt. Zum Schluss werden Variablen an die grafischen Elemente gebunden und das fertige Produkt üblicherweise durch eine Webtechnologie bereitgestellt. Die Komplexität der HMI-Designprozesse kann durch den Einsatz von LLM reduziert werden. Gegenwärtig sind zwei Szenarien für HMI-Designprozesse vorherrschend. Die erste Methode umfasst das Nachbauen eines bereits durch den Kunden gelieferten Prototypen. Die zweite, häufiger auftretende Methode ist die freie Erstellung von HMIs, die insbesondere dann zum Tragen kommt, wenn auf Kundenseite kein Designprototyp vorhanden ist. Beide Ansätze beinhalten zahlreiche Routineaufgaben, wie z. B. das Erstellen von Buttons und das Hinzufügen von Bildern oder die Verknüpfung der Signale, welche an Variablen gebunden sind, mit den entsprechenden grafischen Elementen. Für die grafische Umsetzung und das Verbinden der Variablen stehen verschiedene Editoren zur Verfügung, die von XML-basierten Verfahren bis hin zu HTML5 und OPC UA reichen. HTML5 ist die aktuellste Version der Hypertext Markup Language, mit der sich digitale Dokumente strukturieren und hierarchisieren lassen. OPC UA steht für Open Platform Communications Unified Architecture und ist ein industrieller Kommunikationsstandard. Beide Methoden können durch Technologien der generativen KI automatisiert und beschleunigt werden. Für die Umsetzung lassen sich dabei die Konzepte der Generierung von Programmiercode übertragen, da die zugrunde liegenden Sprachen Skript- oder Programmiersprachen sind. Ergänzend mit einer umfassenden Datensammlung lassen sich LLM nachtrainieren, sodass spezifische Designs für den Sektor der Automatisierungstechnik generiert werden können. Eine besondere Herausforderung besteht bei der Analyse von grafischen Elementen. Zum Beispiel können Rechtecke mit einem Symbol und Text sowohl Fensterelemente als auch Bedienfelder darstellen. Eine weitere Herausforderung stellt die Verknüpfung von Variablen mit grafischen Elementen dar. Bei diesem Aspekt der HMI-Generierung besteht derzeit ebenfalls noch Forschungsbedarf.

Anwendungsbeispiel 1: Generierung von ST-Code

Ein Anwendungsbeispiel für LLM im Software-Lebenszyklus ist die automatisierte Generierung von Steuerungs-Code. Steuerungen werden in einer nach der Norm IEC 61131-3 standardisierten Programmiersprache geschrieben, wobei ST in Deutschland am weitesten verbreitet ist. Ziel ist es, den Programmierer bei der Implementierung der Steuerung nach dieser IEC-Norm zu unterstützen, indem Code ergänzt, erzeugt oder korrigiert wird. Grundsätzlich können alle diese Anwendungsfälle durch Sprachmodelle realisiert werden, wobei im Folgenden die Generierung von ST-Code im Fokus steht.

Zum Zeitpunkt der Durchführung der Studie im Jahr 2023 erzielte GPT 3.5 im Vergleich zu anderen großen Sprachmodellen (z. B. Bard, Llama 2, Bloom) die mit Abstand besten Ergebnisse für die Erzeugung von ST-Code. Unter dem Gesichtspunkt der Vertraulichkeit der zu verwendenden Daten kam jedoch nur ein Open-Source-Modell für die Nutzung im Projekt in Frage. Llama 2 und CodeT5+ lieferten bereits sehr gute Ergebnisse für die Generierung von Python-Code und können aufgrund ihrer Größe und Verfügbarkeit grundsätzlich lokal gehostet und nachtrainiert werden. Für das weitere Vorgehen wurde daher ein auf Python vortrainiertes CodeT5+-Modell verwendet und mithilfe von unternehmensspezifischen Trainingsdaten auf strukturierten Text nachtrainiert. Nach aufwändiger Datenaufbereitung durch manuelles Dokumentieren der ST-Codeblöcke und mehreren Trainingsiterationen konnte trotz relativ geringer Trainingsdatenmenge (ca. 300 Beispiele) eine akzeptable Modellgüte erzielt werden. Das bedeutet, dass einfache Anfragen für typische Steuerungsfunktionen (z. B. Zeitfunktionen oder Typenumwandlungen) in verschiedenen Varianten valide generiert werden konnten. Die Validität wurde von Experten und durch Ausführung des generierten Codes in der Softwareumgebung

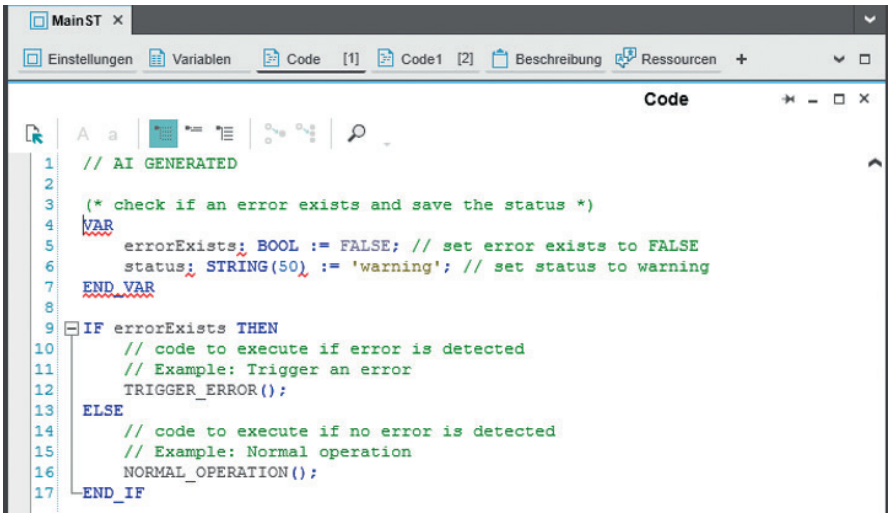

Bild 2. Ansicht des generierten ST Codes in der Steuerungsentwicklungsumgebung am Beispiel des PLCnext Engineer von Phoenix Contact. Der Kommentar (**) entspricht der Anweisung an das LLM, der darunter stehende Baustein dem produzierten ST Code. Die Kommunikation zwischen Entwicklungsumgebung und LLM erfolgt über das Editieren bzw. Neuladen einer Text-Datei

geprüft. Komplexere Anfragen, die größere Programme mit mehreren verknüpften Funktionen beinhalteten, lieferten teilweise noch unzureichende Ergebnisse. Dennoch konnte gezeigt werden, dass grundsätzlich kleine Modelle für spezifische Aufgaben lokal trainiert und im Fall von CodeT5+ sogar auf einem herkömmlichen Laptop ausgeführt werden können. Dies ermöglicht die Auslieferung spezifischer trainierter Modelle beispielsweise an den Nutzer von Steuerungsentwicklungsumgebungen, wobei die Daten jederzeit lokal bleiben und der Modellaufruf ohne signifikante Latenz erfolgt. Eine mögliche Kopplung der ST-Generierung durch das CodeT5+-Modell mit einer Steuerungsentwicklungssoftware ist in Bild 2 illustriert – das generierte Ergebnis wird (nach Neuladen der Datei) direkt im Editor angezeigt.

Es ist zu erwarten, dass die Güte des lokalen Modells durch eine Erweiterung des Trainingsdatensatzes deutlich gesteigert werden kann. Für Ausgaben, die keine Spezifika bzgl. Nomenklatur oder Stil beinhalten, kann auch eine Anreicherung der Daten mit generischen Beispielen vorgenommen werden. Eine weitere Verbesserung der Performance und Validität kann durch eine integrierte Feedbackschleife mit dem Nutzer bzw. der Software (bspw. Berücksichtigung von Fehlermeldungen bei Ausführung des Codes oder Registrierung von Änderungen durch den Nutzer) realisiert werden.

Anwendungsbeispiel 2: RAG-basiertes Assistenzsystem für Engineering-Software

Ein weiteres Beispiel stellt die direkte Unterstützung des Nutzers der Engineering-Software durch die Beantwortung spezifischer Fragen (z. B. durch ein Chatfenster in der Software) dar. Der Aufbau eines derartigen RAG-Systems ist in Bild 3 schematisch gezeigt. Das System nutzt ein Sprachmodell (z. B. GPT-4) zur Verarbeitung von Anfragen, wobei die Anfragen mithilfe einer Datenbank durch spezifischen Kontext ergänzt werden. Im vorliegenden Fall werden als spezifischer Kontext Hilfeseiten und Online-Tutorials zu einer bestimmten Softwareentwicklungsumgebung in einer Vektordatenbank hinterlegt. Bei jeder Abfrage werden dann ähnliche Textstellen aus den hinterlegten Dokumenten in den Prompt eingefügt, so dass das Sprachmodell, auch wenn die Texte nicht Teil der Trainingsmenge waren, spezifische Antworten zu diesen Dokumenten generieren kann. In diesem Fall kann ein großes kommerzielles Sprachmodell verwendet werden, da die verwendeten hinterlegten Dokumente ohnehin öffentlich zugänglich sind. Für die Umsetzung des RAG-Systems wurden verschiedene Methoden für die Vorverarbeitung der Daten, das Retrieval und die Generierung getestet und gegenübergestellt. Beispielsweise wurde bei der Vorverarbeitung die Aufteilung der Dokumente in Abschnitte (Chunks) entweder nach einer fixen Größe, satzweise oder entsprechend dem semantischen Kontext vorgenommen. Beim Retrieval wurden u. a. die Anzahl der relevantesten Chunks und das verwendete Ähnlichkeitsmaß und bei der Generierung die Reihenfolge der ermittelten Chunks variiert. Zur Evaluation diente ein Satz von Fragen, der durch Experten beantwortet und deren generierte Antworten auf ihre Richtigkeit hin geprüft wurden. Ein Großteil der Fragen konnte,

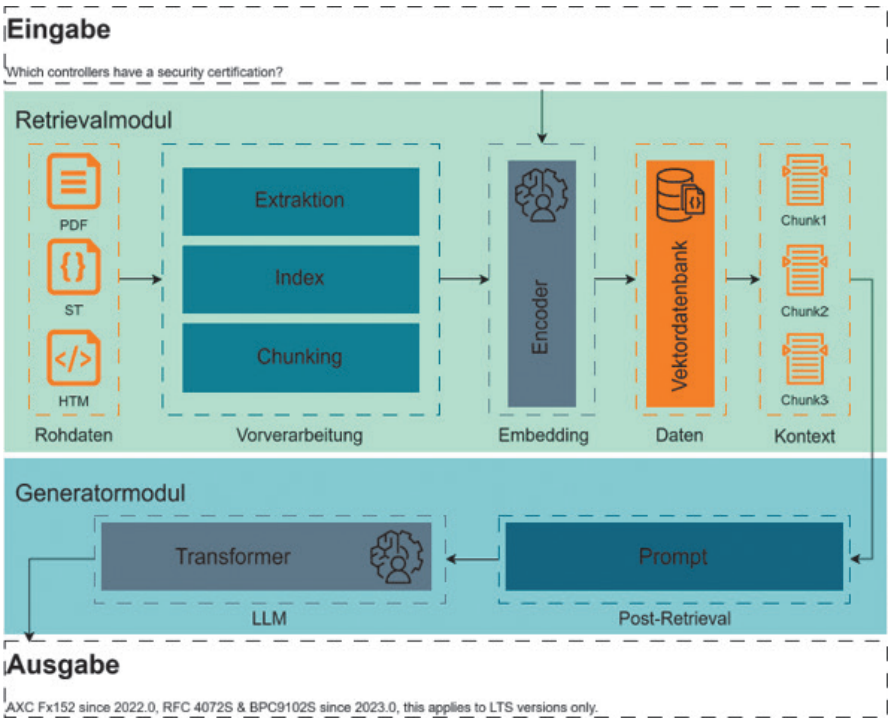

Bild 3. Schematischer Aufbau eines RAG-Systems zur Beantwortung spezifischer Nutzeranfragen im Engineering Kontext. Das Retrieval-Modul enthält eine Vektordatenbank mit Texten aus Foren und Hilfe-Seiten, welche bei einer Anfrage passend ausgewählt und dem Generatormodul als Kontext mitgegeben werden

Effizienter mit Generativer KI

Zunehmender Fachkräftemangel und der Bedarf an profitablen, skalierbaren Lösungen lässt generative KI im Bereich der Automatisierungstechnik in den Fokus rücken. Wir präsentieren Ansätze, um SPS-Programmierer:innen entlang des gesamten Softwareentwicklungszyklus durch den Einsatz von großen Sprachmodellen zu unterstützen. Durch automatisierte SPS- Code-Generierung und kontextsensitive Beantwortung spezifischer Nutzeranfragen bei der Bedienung von Engineering Software werden Effizienzgewinne und Qualitätssteigerungen ermöglicht.

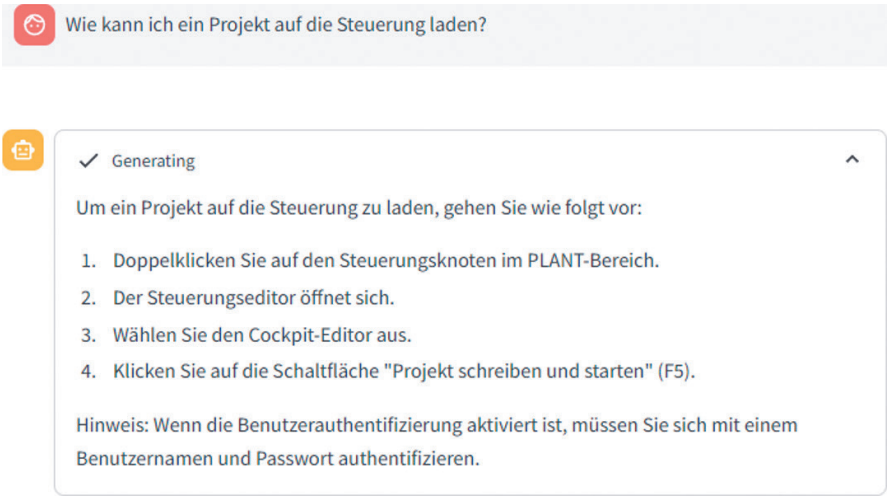

Bild 4. GUI des RAG-Systems mit Beispielfrage. Eine mögliche Anfrage, für deren Beantwortung spezifischer Kontext aus der Vektordatenbank des Retrieval-Moduls herangezogen wird

unabhängig von ihrem Schwierigkeitsgrad (von befragten Experten eingestuft), zufriedenstellend beantwortet werden. Ein geringer Anteil der Fragen (< 10 %) konnte durch das System nicht zufriedenstellend beantwortet werden, was jedoch auf eine zweideutige Fragestellung zurückzuführen ist. Eine beispielhafte Frage und die zugehörige Antwort in der entwickelten GUI sind in Bild 4 dargestellt. Um weiterführende Unterstützung innerhalb der Softwareentwicklungsumgebung zu realisieren, können der Projektkontext oder die Nutzerhistorie zur Beantwortung der Fragestellung hinzugezogen werden. Eine solche Lösung müsste dann zur Wahrung der Vertraulichkeit mittels lokaler Modelle oder innerhalb einer abgesicherten Umgebung implementiert werden.

Zusammenfassung und Ausblick

In diesem Beitrag wurden Potenziale für den Einsatz generativer KI entlang des gesamten Softwareentwicklungszyklus in der Automatisierungstechnik dargestellt. Es zeichnet sich ab, dass die Möglichkeiten vielfältig sind, durch generative KI derzeit mühsame und langwierige Tätigkeiten zu unterstützen. Das Spektrum reicht von der automatisierten Erstellung von Pflichtenheften über die Generierung von E-Plänen und HMI-Layouts bis hin zur automatisierten SPS-Code-Erstellung, Test und Dokumentation. Dabei ist der Reifegrad derzeit verfügbarer Lösungen sehr unterschiedlich und hängt von der Verfügbarkeit großer kuratierter Datenmengen und deren Verwertbarkeit in großen kommerziellen Sprachmodellen ab. Insbesondere stellt die Berücksichtigung von Datenschutzaspekten bei der Verwendung sensibler Daten mit großen kommerziellen Sprachmodellen noch eine Hürde dar. Es ist daher zu erwarten, dass LLM-basierte Assistenzsysteme für die Softwareentwicklung in der Automatisierungstechnik zukünftig auf hybriden Architekturen aufsetzen, welche im Hintergrund unterschiedliche Sprachmodelle verwenden, je nach Art und Klassifizierung der Fragestellung. Weiterhin besteht derzeit noch großer Bedarf an der Feinjustierung existierender Verfahren, z. B. Embedding-Modellen in RAG-Systemen und multimodalen Modellen für Bildverarbeitung speziell für automatisierungstechnische Bedarfe.

Literatur

1. Achiam, J.; Adler, S.; Agarwal, S. et al.: GPT-4 Technical Report. OpenAI, 2023 DOI:10.48550/arXiv.2303.08774
2. Devlin, J.; Chang, M.-W.; Lee, K. et al.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2019), S. 4171–4186 DOI:10.48550/arXiv.1810.04805
3. Touvron, H.; Martin, L.; Stone, K. et al.: Llama 2: Open Foundation and Fine-Tuned Chat Models. 2023 DOI:10.48550/arXiv.2307.09288
4. Touvron, H.; Lavril, T.; Izacard, G. et al.: LLaMA: Open and Efficient Foundation Language Models. 2023 DOI:10.48550/arXiv.2302.13971
5. Anil, R. et al.: Gemini: A Family of Highly Capable Multimodal Models. 2023 DOI:10.48550/arXiv.2312.11805
6. Zhao, W. X.; Zhou, K.; Li, J. et al.: A Survey of Large Language Models. 2023 DOI:10.48550/arXiv.2303.18223
7. Dukino, C.; Friedrich, M.; Ganz, W. et al.: Künstliche Intelligenz in der Unternehmenspraxis. Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO, Fraunhofer-Verlag, Stuttgart 2019 DOI:10.24406/publica-fhg-300040
8. Weyrich, M.: Industrielle Automatisierungs- und Informationstechnik. Springer, Berlin und Heidelberg 2023 DOI:10.1007/978-3-662-56355-7
9. Holm, T.: Aufwandsbewertung im Engineering modularer Prozessanlagen. Dissertation. Helmut-Schmidt-Universität, Hamburg 2016 DOI:10.51202/9783186465207
10. Tufano, M.; Agarwal, A.; Jang, J. et al.: AutoDev: Automated AI-Driven Development. 2024 DOI:10.48550/arXiv.2403.08299
11. Fan, A.; Gokkaya, B.; Harman, M. et al.: Large Language Models for Software Engineering: Survey and Open Problems. In: International Conference on Software Engineering: Future of Software Engineering. IEEE/ACM 2023, S. 31–53 DOI:10.1109/ICSE-FoSE59343.2023.00008
12. Abualhaija, S.; Arora, C.; Dell'Anna, D. et al.: Preface: 7th Workshop on Natural Language Processing for Requirements Engineering. In: CEUR Workshop Proceedings 3672 (2024)
13. Dalpiaz, F.; Ferrari, A.; Franch, X. et al.: Natural Language Processing for Requirements Engineering: The Best Is Yet to Come. IEEE Software 35 (2018) 5, S. 115–119 DOI:10.1109/MS.2018.3571242
14. Ronanki, K.; Berger, C.; Horkoff, J.: Investigating ChatGPT's Potential to Assist in Requirements Elicitation Processes. In: 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE 2023, S. 354–361 DOI:10.48550/arXiv.2307.07381
15. Ruan, K.; Chen, X.; Jin, Z.: Requirements Modeling Aided by ChatGPT: An Experience in Embedded Systems. In: 31st International Requirements Engineering Conference Workshops (REW). IEEE 2023, S. 170–177 DOI:10.1109/REW57809.2023.00035
16. Drath, R.; Luder, A.; Peschke, J. et al.: AutomationML – the Glue for Seamless Automation Engineering. In: International Conference on Emerging Technologies and

Factory Automation. IEEE 2008, S. 616–623 DOI:10.1109/ETFA.2008.4638461
17. Vogel, G.; Schulze Balhorn, L.; Schweidtmann, A. M.: Learning from Flowsheets: A Generative Transformer Model for Autocompletion of Flowsheets. Computers & Chemical Engineering 171 (2023) 1 DOI:10.1016/j.compchemeng.2023.108162
18. Mattmüller, J.; Benndorf, A. G.; Preintner, P.: Automatisierte Generierung von digitalen Anlagenschemata. 9th BauSim Conference (2022), IBPSA-Germany and Austria, 2022 DOI:10.26868/29761662.2022.57

Die Autor:innen dieses Beitrags

Dennis Reinhardt erlangte im Jahr 2024 den akademischen Grad Bachelor of Science. Er studierte zwischen 2021 und 2024 Data Science an der TH OWL und war zeitgleich dualer Student am Fraunhofer IOSB-INA. Seit 2024 ist er dualer Masterstudent an der Universität Bielefeld im Bereich Intelligente Interaktive Systeme.

Jörg Jeschin studierte Elektrotechnik an der Technischen Universität Braunschweig und begann 1996 seine Laufbahn bei Phoenix Contact als Softwareentwickler. Ab 2003 übernahm er Führungspositionen im Bereich Research & Development in der Automatisierungssparte des Unternehmens. Seit 2011 führt er als Director die Produktlinie Software and Safety in der Business Area Industriemanagement und Automation (IMA).

Prof. Dr. Jürgen Jasperneite studierte Elektrotechnik und Informationstechnik und promovierte 2002 an der Otto-von-Guericke-Universität Magdeburg. Von 1988 bis 1990 war er als Entwicklungsingenieur bei der Robert Bosch GmbH beschäftigt. Von 1990 bis 2005 war er in unterschiedlichen Funktionen im Entwicklungsbereich der Phoenix Contact GmbH tätig, zuletzt als Entwicklungsleiter des Geschäftsbereiches Automation Systems. Seit 2005 ist Jürgen Jasperneite Professor für Computernetzwerke der TH OWL in Lemgo. 2009 gründete er das Fraunhofer IOSB-INA, dessen Institutsleitung er seitdem innehat.

Dr. Gesa Benndorf studierte Physik an der Friedrich-Schiller-Universität Jena, promovierte 2013 an der TU Dresden und dem Max-Planck-Institut für Physik komplexer Systeme und war anschließend 10 Jahre am Fraunhofer ISE in Freiburg tätig bevor sie 2023 die Gruppenleitung für Maschinelles Lernen am Fraunhofer IOSB-INA in Lemgo übernahm.

Abstract

ChatPLC – Potential of Generative AI for PLC Engineering. In this article we discuss approaches to support automation engineering through generative artificial intelligence (AI) across the entire software development cycle. In particular, the implementation of a local language model for structured text generation according to the IEC 61131-3 norm and the development of a Retrieval Augmented Generation (RAG) system to assist the user with the engineering software is described here. These examples highlight the great potential of employing Large Language Models (LLM) in the area of automation engineering which are already within reach.

Schlüsselwörter

Generative KI, Steuerungsentwicklung, SPS, Sprachmodelle, Softwareentwicklungszyklus

Keywords

Gen AI, Software Development, PLC, Large Language Models, Engineering Automation, Retrieval Augmented Generation

Bibliography

DOI:10.1515/zwf-2024-0121
ZWF 120 (2025) Special Issue; page 196 – 201

ISSN 0947-0085 · e-ISSN 2511-0896