Wenn aus Requirements Engineering Prompt Engineering wird: Wie bereiten wir uns darauf vor?

Jörn Fahsel, Friedrich-Alexander-Universität Erlangen-Nürnberg, joern.fahsel@fau.de
Dr. Privat-Doz. Andrea Herrmann, Herrmann & Ehrlich, Stuttgart, AH_Science@gmx.de
Prof. Dr. Rüdiger Weißbach, Hochschule für Angewandte Wissenschaften (HAW) Hamburg, ruediger.weissbach@haw-hamburg.de

Abstract: Anforderungen (im weitesten Sinne) dienen als Kommunikationsmittel mit den Stakeholdern und als Grundlage für die Software-Erstellung. Doch welche Form werden sie einnehmen, angesichts aktueller Trends wie Künstlicher Intelligenz, agiler Entwicklung und global verteilter, asynchron arbeitender Teams? Das klassische, aufwändige Upfront Requirements Engineering scheint ausgedient zu haben, auch in den sicherheitskritischen Bereichen. Am Ende könnte aus Requirements Engineeering Prompt Engineering werden. Doch wie bereiten wir in und mit der Lehre diese Zukunft des Requirements Engineering vor?

Der Arbeitskreis

Der Arbeitskreis „Requirements Engineering (RE) und Lehre“ wurde 2013 gegründet [1] und trifft sich regelmäßig in Telefonkonferenzen. Wir sind alle drei in der Hochschullehre, aber auch in kommerziellen Schulungen für das Requirements Engineering tätig.

Drei Trends

Wir haben drei Trends identifiziert, die das Requirements Engineering in der Praxis aktuell prägen und zukünftig prägen werden:

- agile iterative Entwicklungsprozesse,
- global verteilte, asynchron arbeitende Teams und
- Textverarbeitung durch vortrainierte Künstliche Intelligenzen.

Diese sind bereits dabei, das Requirements Engineering zu verändern. Doch wo könnte es hingehen? Und wie bereiten wir unsere Studierenden und Kursteilnehmenden darauf vor?

Andere Autoren wie beispielsweise Dehn et al. [2] gehen noch von dieser Annahme aus: Es werden weiterhin detaillierte textuelle oder modellbasierte Anforderungsspezifikationen erstellt, einschließlich der dafür nötigen Traceability zwischen den Anforderungen. Die Ermittlung, Dokumentation, Analyse und Verifikation werden wie bisher durchgeführt, nur werden die einzelnen Teilschritte („elementary process steps“) durch KI unterstützt. Wir denken, dass die Folgen auf den RE-Prozess weitreichender sein werden.

Folgen der Trends

Als Konsequenz der oben genannten Trends sehen wir folgende Anforderungen und Möglichkeiten im Requirements Engineering Prozess:

- Die iterative Arbeitsweise beinhaltet die regelmäßige Erstellung von Prototypen für die Kommunikation mit den Stakeholdern. Diese können zukünftig auf der Grundlage wohlformulierter Prompts mit Hilfe von KI-Werkzeugen schnell generiert werden. Dies erlaubt auch die Erzeugung von alternativen Varianten, aus denen sich die Stakeholder den passendsten Prototypen auswählen können. Änderungsvorschläge können schnell eingearbeitet werden.
- Die automatische Generierung von Code aus abstrakteren Darstellungen wie Klassendiagrammen oder Prozessmodellen ist bereits Stand der Technik. Wenn diese aus den Prototypen automatisch generiert werden können, dann müssen nur noch die Prompts manuell geschrieben werden; alle anderen Software Engineering Artefakte entstehen automatisiert, mit oder ohne Künstliche Intelligenz. Dies könnte dem in Abbildung 1 dargestellten vereinfachten Software Engineering Prozess folgen. Die Rolle des Menschen beschränkt sich dann auf das Prompt Engineering sowie das Review der Artefakte auf Konformität mit den Benutzerbedürfnissen sowie Qualitätskriterien wie Sicherheit und das Akzeptieren der Lösungsvorschläge.
- Das Prompt Engineering muss systematisiert und standardisiert werden, so wie bisher das Requirements Engineering. Ein Mittel dazu können Prompt Patterns [3] sein.

- Die Leistungsfähigkeit von Prompt Engineering könnte die Relevanz grafischer Modelle in der Modellierung und der Interaktion über die Modellierung reduzieren; der Prototyp ist schnell verfügbar.
- Die Reviews können ebenfalls teilweise automatisiert werden oder zumindest die Checklistenerstellung für die manuellen Reviews kann automatisch erfolgen.
- Auch das Erkennen von Traceability-Beziehungen zwischen Artefakten, die Durchführung einer Impact-Analyse vor einer Änderung, Risikoanalysen in der Architektur sowie das Erstellen von Berichten über die Softwarequalität können automatisiert werden.
- Global verteilte, asynchron arbeitende Teams brauchen die zentrale, für alle zugängliche Ablage aller Artefakte, Zugriff auf die Tools und zusätzlich regelmäßige Kommunikation.

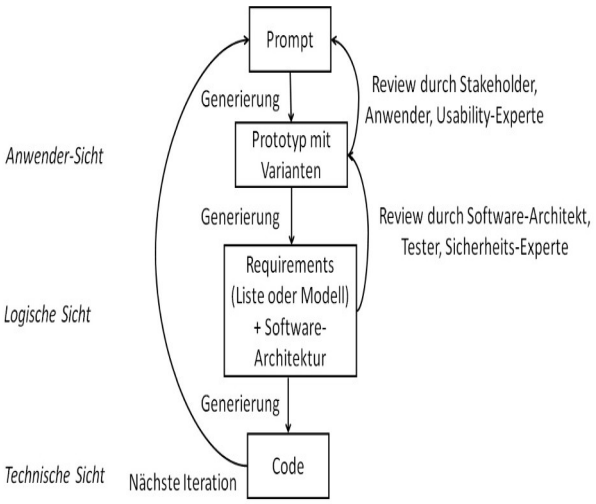

Abbildung 1: Software Engineering Prozess vom Prompt zum Code

Auswirkungen auf die Lehre

Obwohl das Obige wie Zukunftsmusik klingt, wird es beim aktuellen Tempo des technischen Fortschritts nicht lange dauern, bis die Vorstellung – zumindest in Teilen – wahr wird. Wie lehrt man etwas, das sich noch in Entstehung befindet?

Zum einen sind wir der Meinung, dass man nichts automatisieren sollte, was man nicht auch von Hand tun könnte. Das heißt, die Studierenden müssen sehr wohl noch Anforderungen selbst formulieren, UML-Diagramme erstellen und Codieren lernen, um ein Gefühl dafür zu bekommen.

Zum anderen beziehen wir unsere Studierenden in unseren eigenen Lern- und Explorationsprozess mit ein [4]. In Hausarbeiten, Seminaren und Abschlussarbeiten bearbeiten unsere Studierenden Themen wie „Erstellen Sie mit Hilfe von chatGPT User Stories und bewerten Sie deren Qualität“. Damit erlernen sie auch Strategien für das Austesten solcher Werkzeuge für die Arbeit im Requirements Engineering und den Review der erzeugten Ergebnisse. Genau das werden sie zukünftig benötigen.

Referenzen

[1] Webseite des Arbeitskreises „RE und Lehre“: https://rg-stuttgart.gi.de/arbeitskreise/ak-requirements-engineering-und-lehre

[2] Simon Dehn, Georg Jacobs, Thilo Zerwas, Joerg Berroth, Matthis Hötter, Matthias Korten, Marvin Müller, Nico Gossen, Serena Striegel, Dirk Fleischer: On identifying possible artificial intelligence applications in requirements engineering processes. Forschung im Ingenieurwesen (2023) 87:497–506

[3] Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, Douglas C. Schmidt: ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design (2023). arXiv-Preprint unter https://arxiv.org/pdf/2303.07839.pdf

[4] Jörn Fahsel, Vera Kraus, Tabea Kurreck, Kea Rahmann: Agile Lehre - Erfahrungsberichte. Software Engineering (Workshops) 2016:209-219