GRAND CHALLENGES 2025

Vollautomatisierte Software-Entwicklung?

GESELLSCHAFT FÜR INFORMATIK

Software vollautomatisch entwickeln – diese Vision scheint mit den kürzlichen Fortschritten bei Large Language Models (LLMs) nicht mehr unmöglich. Die tatsächliche Umsetzung könnte allerdings nicht nur wünschenswerte Konsequenzen für die Gesellschaft haben. Diese Grand Challenge nimmt mögliche Hürden und Potentiale unter die Lupe.

Software Engineering (SE) befasst sich mit der Transformation von implizitem und explizitem Wissen von Sachverhalten und Anforderungen in immaterielle Artefakte. Die produzierten Artefakte sind dabei hoch formalisiert: Insbesondere Programme sind formale Beschreibungen dessen, was Computer tun sollen. Aufgrund der hohen Formalisierung bei gleichzeitiger Zugänglichkeit der Artefakte für die eigenen Methoden ist das Software Engineering seit jeher der Gegenstand von Automatisierung: Eine Vielzahl von Werkzeugen hilft beim Anforderungsmanagement, beim Entwurf der Software, bei der Erstellung des Endprodukts in komplexen Build-Pipelines sowie bei der Testerzeugung und -ausführung.

Mit der generativen KI hat nun ein neuer Player die Bühne des Software Engineering betreten: Insbesondere große Sprachmodelle (Large Language Models, LLMs) enthalten einen Teil unseres Allgemeinwissens und können nun ebenfalls auf Basis informell beschriebener Eingaben formalisierte Ausgaben wie Programmcode erzeugen. Gerade die Tatsache, dass es sich bei der Softwareentwicklung im Wesentlichen um einen sprachbasierten Transformationsprozess handelt, macht sie zu einem idealen Anwendungsfall für sprachbasierte generative Transformer. Vor diesem Hintergrund scheint die Vision, Software vollautomatisch zu entwickeln, nicht mehr unmöglich.

Ein vollautomatisches Software Engineering hätte erhebliche gesellschaftliche Auswirkungen: Zum einen kann es dem bestehenden Fachkräftemangel entgegenwirken und zu einer höheren Wertschöpfung führen. Zum anderen kann es zur Folge haben, dass jeder Mensch für seinen Anwendungsfall Software erzeugen kann, wodurch die digitale Teilhabe drastisch erhöht würde. Neben diesen wünschenswerten Auswirkungen sind allerdings auch dystopische vorstellbar: Automatisch erzeugte Software kann zu Kontrollverlusten führen, die in ihrem Schaden den von heute bekannten Softwareausfällen bei weitem übertreffen. Ein Teil der Herausforderung ist deshalb, zu klären und zu reflektieren, wie weit die Automatisierung der Software-Entwicklung gehen kann und soll. Die Grand Challenge insgesamt ist, die technische Entwicklung so zu kanalisieren, dass die Gesellschaft nur die gewünschten Auswirkungen erfährt.

Das ist die Challenge

Die KI-basierte Automatisierung des Software Engineerings (SE) ist in gewisser Weise ein Idealfall für die KI-basierte Automatisierung insgesamt, da

1. das Gebiet einen hohen Grad an Regelmäßigkeit aufweist (Verwendung normierter Sprachen),
2. eine umfangreiche kuratierte Trainingsbasis zur Verfügung steht (Software-Repositories, Tutoring-Systeme wie StackOverflow etc.),

3. das Produkt vollständig immateriell ist und damit unbeschränkt durch KI hergestellt und konsumiert werden kann und
4. die Qualität von Zwischenergebnissen gut objektivierbar ist (ein Programm compiliert oder nicht, die Tests laufen durch oder nicht), was automatische Rückkopplungsschleifen erlaubt.

Diese günstige Ausgangslage wird bereits massiv genutzt. Auch erhalten aktuelle subsymbolische KI-Systeme (wie GPT) Schnittstellen, über die sie sich mit traditionellen Systemen (wie beispielsweise einer Websuche) integrieren lassen. Auch können die KI-Systeme bereits auf nicht öffentlich verfügbare Daten angepasst oder auf einzelne Anwendungsdomänen spezialisiert werden. Nimmt man all dies zusammen, so lässt sich erwarten, dass das automatische SE in den nächsten Jahren bislang nie dagewesene Entwicklungssprünge macht.

Vor diesem Hintergrund besteht die Grand Challenge darin, die Entwicklung der Automatisierung im SE so zu kanalisieren, dass ein gesellschaftlich wünschenswertes Ziel erreicht wird. Dazu gehören neben Anstrengungen, große, komplexe Software verlässlich mit vorhersagbaren Qualitätseigenschaften zu erzeugen, auch die ethische Reflektion darüber, was nicht automatisiert werden darf. Auf eine allgemeine Ebene gehoben, gehört dies zur Frage, was das richtige Maß der Zusammenarbeit zwischen Mensch und KI ist. SE kann hier — und sollte vermutlich — die Rolle eines Piloten übernehmen.

Die Grand Challenge kann auf eine Reihe von Einzelfragen heruntergebrochen werden:

1. Ethische Reflexion: Die Verantwortung für das gesetzeskonforme und sozial akzeptable Verhalten von Software liegt derzeit bei Menschen. Dies gilt insbesondere auch für noch nicht (ausreichend) regulierte Einsatzkontexte, in denen über die Software und ihren Einsatz ethisch reflektiert werden soll (vgl. ethischen Leitlinien der GI). Überträgt sich diese Verantwortung auf die Initiierenden oder muss sie im Automatisierungsprozess selbst verankert werden?
2. Qualitätssicherung: Wie können wir sicherstellen, dass die generierten Produkte verlässlich sind? Wie können bisherige Werkzeuge wie Beweisverfahren auch für KI nutzbar werden und eingebunden werden, um Verlässlichkeit zu erhöhen und Garantien zu geben?
3. Zukünftige Rollen: Wie verschieben sich die Rollen und Verantwortlichkeiten der diversen Stakeholder bei der zukünftig KI-gestützten (Weiter-)Entwicklung und Betrieb von „softwareintensiven“ Systemen? Was sind die Konsequenzen für die Ausbildung dieser Stakeholder, die Wirtschaft (insbesondere IT), die Gesellschaft und die Regulierung?
4. Umgang mit dem Automation Bias: Mit der Qualität der automatisch erzielten Ergebnisse steigt auch die Neigung der Menschen, diesen blind zu glauben, selbst wenn sie nicht plausibel sind. Dies wird sich spätestens dann als fatal erweisen, wenn auch die Qualitätssicherung der Automation überlassen wird. Welches sind also die absoluten Mechanismen der Qualitätssicherung, die zum Einsatz kommen müssen?
5. Zuschreibung und Ownership: Bei Fragen der Vergütung oder Haftung geht es wesentlich darum, wer ein Produkt erstellt hat. Wie lässt sich die Herkunft von Softwareprodukten und die Grundlage der darin verkörperten Entscheidungen mit der nötigen Sicherheit nachweisen beziehungsweise rückverfolgen? Wie kann

sichergestellt werden, dass sich menschliche Akteure für die generierten Produkte verantwortlich fühlen?
6. Expertise bewahren: Wenn im Regelfall KI die Software generiert, wie können wir sicherstellen, dass Menschen die Expertise bewahren, die Aufgaben auch manuell ausführen zu können? Wie müssen Curricula zukünftig aussehen?
7. Erklärbarkeit: Im SE muss Stakeholdern erklärt werden, dass das erzeugte System tatsächlich den angegebenen Anforderungen, aber auch weiteren Befindlichkeiten und Rahmenbedingungen, entspricht. Erklärbarkeit von KI ist ein bisher nicht zufriedenstellend adressiertes Problem. Wenn dies im stark formalisierten Gebiet des SE nicht gelingt, kann es dann überhaupt gelingen?
8. Usability: Nicht zuletzt muss auch die Benutzbarkeit von KI-Systemen im SE verbessert werden, weil sich sonst die gewünschten „Demokratisierungseffekte“ nicht einstellen.

Darum ist es wichtig, das Problem jetzt anzugehen

Der Übergang von einer ungenauen, informellen und unvollständigen Vorstellung auf ein zufriedenstellendes Programm ist das zentrale und harte Problem der Softwareentwicklung. Auch wenn statistische Modelle wie die aktuellen LLM und darauf aufbauende generative KI nun sehr vielversprechend scheinen, bleibt noch unklar, ob und wie gut sie diese Lücke schließen können. Die Verfahren, die auf tiefen neuronalen Netzen basieren, sind black-box und liefern Stand heute keinerlei Garantien. Daher ist eine zentrale Schwierigkeit, diese erstaunlich leistungsfähigen aber potenziell nicht verlässlichen Ansätze mit bestehenden und noch zu entwickelnden Lösungen zum Nachweis von Qualität zusammenzubringen.

Wichtig für die Einschätzung der Schwierigkeit des Problems ist es außerdem, sich vor Augen zu führen, wie vielfältig die heutigen Aufgaben im Software Engineering sind. Diese reichen von der initialen Produktidee, der Klärung der Anforderungen, der Entwicklung und dem Verständnis von Fachterminologien, über Entwurf, Implementierung und Qualitätssicherung bis hin zur Lösungsdokumentation und gegebenenfalls zur Systemzertifizierung. Auch in Zukunft werden sich Anforderungen über die Zeit immer ändern. Daher muss nicht nur zu einem Zeitpunkt Software generiert werden, sondern es besteht die Notwendigkeit, alles Wissen und alle Produkte trotz kontinuierlicher Änderungen konsistent zu halten. Die Hoffnung ist, dass generative KIs zukünftig die verallgemeinerten Übersetzungs- und Mustererkennungsfähigkeiten, die notwendig sind, um diese Aktivitäten (voll) zu automatisieren, besitzen oder mit angemessenem Aufwand erlernen können. Es bleibt zu analysieren, ob allerdings kreative Aspekte des Software Engineerings, wie beispielsweise innovative Produktentwicklung, auch sinnvoll (teil)automatisiert werden können und ob dies wünschenswert ist. Wie diese kreativen Ideen dann wiederum bewertet werden können, auch in Hinblick auf ihre Wirkung auf Systeme und Gesellschaft, ist ebenfalls offen.

Dass Software kontinuierlich entwickelt wird und daher nicht „auf der grünen Wiese“ entsteht, sondern meist vor den Hintergrund von einem oder mehreren bestehenden Systemen, bedeutet auch, dass zukünftige generative KIs Lösungen generieren müssen, die in den Kontext des Bestands passen. Hier erscheint es vielversprechend, bestehende Tools beispielsweise zum Verwalten von Abhängigkeiten auch für KIs nutzbar zu machen — noch gibt es aber keine konkreten Lösungen in diese Richtung.

Eine weitere Schwierigkeit wird es sein, dass KI für die Lösung eines konkreten Problems auch über das passende Domänenwissen verfügen muss, etwa für eine Domäne im Allgemeinen oder auch ein einzelnes Unternehmen. Die nötige Domänenexpertise aufzubauen, ist heute bereits ein wichtiger Schritt in der Software-Entwicklung. Diese Expertise in einen hochautomatisierten Prozess zu übertragen, wird eine Herausforderung sein, insbesondere da der Wissensschatz von einzelnen Unternehmen nicht in den generischen LLMs enthalten ist. Fine-Tuning und Retrieval-Augmented LLMs könnten eine Lösungsrichtung sein. Hier stellt sich aber wieder die Herausforderung, dass beispielsweise Fine-Tuning für kleine und mittelständige Unternehmen mangels Fachkräfte, Ressourcen und Expertise nicht durchgeführt werden kann. Andererseits ist die Preisgabe von Unternehmenswissen in fremde Silos ebenfalls wenig erstrebenswert. Wie hier eine Konzentration von technischer Expertise auf wenige, womöglich außereuropäische Akteure vermieden werden kann, ist eine Herausforderung im Bereich KI, die über die Software-Entwicklung hinausgeht.

Falls ein viel höherer Automatisierungsgrad als heute erreicht werden kann, kann sehr schnell Software für neue Probleme entwickelt werden. Eine gesellschaftliche Herausforderung wird es sein, dass Regulierungsbehörden nicht ausreichend Zeit und Expertise haben, um hier verbindliche Regeln zu definieren, bevor die Systeme genutzt werden. Hier müssen Fachleute die Gesetzgebenden auf nationaler und internationaler Ebene dabei unterstützen, gute und wirksame Regularien zu entwickeln und in einem fortlaufenden Prozess weiterzuentwickeln.

Schließlich besteht das Problem der Akzeptanz: Wenn hochgradige Automatisierung möglich ist, wird es gesellschaftlich akzeptiert werden, wenn viele Aufgaben heutiger Entwickler automatisiert werden?

In diesem Zeithorizont ist eine Lösung zu erwarten

In drei bis fünf Jahren könnte es bereits generative KI-basierte Lösungen geben, die hochautomatisiert dabei unterstützen, große Systeme domänenspezifisch zu entwickeln. In zehn Jahren erwarten wir Lösungen, die Initiative übernehmen und hochautomatisiert verlässliche und qualitätsgesicherte Systeme erzeugen, deren Verlässlichkeit auch erklärt werden kann. Die Aufgabe einer Kanalisierung ist es, sich vor die Entwicklung zu setzen.

Diese Anstrengungen wurden bereits in die Lösung investiert

Automatisierung von Aktivitäten und Aufgaben des Software Engineering ist ein Ziel seit Beginn dieser Disziplin. Bisher stellte die Schranke zwischen informell formulierten oder dokumentierten Anforderungen oder Entscheidungen auf der einen und formell beschriebenen Artefakten wie Code auf der anderen Seite aber eine nur schwer zu überwindende Hürde dar. Lösungsansätze sind beispielsweise kontrollierte natürliche Sprache, aus der weitere Artefakte wie Modelle generiert werden können oder die Generierung von Informationen, die dem Menschen bei der Evolution von Software helfen, wie Nachverfolgbarkeitsverbindungen (Tracelinks). Erst durch die kürzlichen Fortschritte bei den Large Language Models kann nun diese Schranke überwunden werden. Bisher werden nur einzelne Code-Abschnitte (z.B. GitHub CoPilot) oder Tests (z.B. CAT-LM [Rao et al., 2023]) generiert.

Welche Disziplinen an dieser Challenge zusammenarbeiten müssen

Um diese Grand Challenge zu adressieren, müssen nicht nur alle Teildisziplinen des SE zusammenarbeiten, sondern auch solche, die sich mit Fragen der Folgenabschätzung beschäftigen. Insbesondere wird für Fragen der Haftung juristische Expertise benötigt und für die Frage, wie viel Automatisierung gesellschaftlich wünschenswert ist, Expertise im Bereich Digital Ethics und Digital Responsibility [Trier et al., 2023].

So könnte das Ziel aussehen

Bei einer Grand Challenge ist beinahe schon definitionsgemäß unklar, ob sie vollständig gelöst werden kann. Um dem Rechnung zu tragen, definieren wir — analog zum autonomen Fahren — fünf Stufen der Automatisierung, von denen die ersten beiden bereits teilweise erreicht sind und die vierte eine wünschenswerte Stufe darstellt und damit unser (Mindest-)Zielszenario ist. Es bleibt unklar, ob Stufe 5 erreicht werden kann und sollte.

- Stufe 1: Die Automatisierung von Routineaufgaben der Transformation erfolgt durch Werkzeuge, die gezielt durch Menschen angestoßen werden. Beispiele sind skriptgesteuerte Build-Pipelines, Anforderungsmanagement (inkl. Traceability) sowie automatische Generierung von Tests. Dabei kommen KI-Systeme wie Recommender zum Einsatz.
- Stufe 2: Die Transformation von informellem Wissen in formale Artefakte wird, angeleitet vom Menschen (z. B. über Prompting), teilweise von KI übernommen. Beispiel: ChatGPT für Spezifikation und Dokumentation, GitHub CoPilot für Programmierung. Grundlage ist hier das in den KI-Systemen verkörperte Allgemeinwissen über die Domäne und die verwendeten Sprachen.
- Stufe 3: Die Transformation erfolgt auf Basis von speziellem Wissen, das sich aus einem bereits (teilweise) bestehenden Software-System als Kontext ergibt. Beispielsweise schlägt ein speziell auf das System trainiertes Modell, das die bestehende Code-Basis kennt, dafür Refaktorisierungen und Ergänzungen zur Umsetzung neuer Anforderungen oder Korrekturen auf Basis von Fehlerberichten vor.
- Stufe 4: Gemischte Initiative: KI schlägt vor, was als nächstes zu tun ist. Etwa analysiert KI Nutzerverhalten und App-Bewertungen und leitet daraus ab, welche Funktionalität als nächstes entwickelt werden sollte.
- Stufe 5: Vollständige Software-Entwicklung durch KI: KI trifft alle oben genannten technischen Entscheidungen selbst und übernimmt Qualitätssicherung. KI interagiert nur noch mit Domänenexperten und/oder Endkunden, um eine Software zu produzieren.

Die Stufen werden nicht gleichzeitig (und eventuell überhaupt) in allen Lebensbereichen erreicht werden: Beispielsweise scheint es, Stand heute, eher denkbar, dass KIs Apps (wie im Bereich Heimautomatisierung) autonom sogar auf Stufe 5 entwickeln, jedoch weniger, dass sicherheitskritische, hochvernetzte Systeme wie im Kontext der Energiesysteme oder andere für die Gesellschaft fundamentale Systeme, wie beispielsweise die elektronische Akte an Gerichten, vollautomatisch im oben genannten Sinne entwickelt werden.

Konkrete Zielszenarien für die Automatisierung und Programmierung durch Endanwender in zwei exemplarischen Lebensbereichen sind die folgenden. Hier verschwimmt die Grenze

zwischen Ersteller der Software und Nutzer der Software sowie zwischen Programmierung im klassischen Sinne und Nutzung von intelligenten Systemen. Daher verdeutlichen diese Beispiele weiter, dass die Schnittstelle der Zusammenarbeit zwischen Mensch und KI neu gestaltet werden muss.

Das erste Szenario wäre das Programmieren eines Haushaltsroboters in (gesprochener) Alltagssprache. Ein Beispieldialog ginge etwa so: „Robo, bring mir nen Kaffee“. Robo: „Wie geht das?” Mensch: „Geh in die Küche, nimm dir eine Tasse, stelle sie unter den Auslauf der Kaffeemaschine, und drücke auf den roten Knopf. Warte, bis die Tasse voll ist, dann bring sie mir”. Eine weitere Aufgabe: „Mach’ mir ein Omelett“ – „Wie?“ – „Schau im Internet nach einem Rezept und mach’s.“ Oder: „Leere die Spülmaschine.“ Dabei ist ein spannendes Teilproblem enthalten, denn der Roboter muss wissen, wo alles hingehört. Instruktion dazu: „Robo, öffne alle Türen und Schubladen in der Küche und merk‘ dir, was drin ist. Nachher stell Gleiches zu Gleichem.“ Hier muss das Sprachverständnis des LLM auf die verfügbaren Fähigkeiten und die API des Haushaltsroboters abgebildet werden und diese bei Bedarf durch neue Fähigkeiten erweitert werden. Analoge Aufgaben könnte man ebenfalls für Industrieroboter, die heute aufwändig von Experten programmiert werden, formulieren.

Ein weiteres Szenario dreht sich um das Instruieren autonomer Fahrzeuge: Angenommen, wir könnten Fahrzeugen die abknickende Vorfahrt wie in der Fahrschule erklären. Das wäre in einer Viertelstunde erledigt. Wir bräuchten also keine 10.000 Stunden Beispielfahrten über abknickende Vorfahrten, bis die Statistik rauskriegt, was dieses Schild bedeutet. Alternativ könnten wir uns das explizite Programmieren für solche Stellen ersparen. Und überprüfbar wäre es auch noch: Inspiziere das erzeugte Programm.

Kriterien für die Zielerreichung der Grand Challenge zu definieren ist zurzeit noch nicht möglich, denn was eine gesellschaftlich wünschenswerte Automatisierung des SE ist, bleibt auszuhandeln.

Welche sozialen, gesellschaftlichen oder ökonomischen Probleme sich mit der Grand Challenge adressieren lassen

1. Wenn sich im Zuge der Bearbeitung der Grand Challenge herausstellt, dass trotz der oben genannten idealen Voraussetzungen das Ziel einer vollständigen Automatisierung nicht erreicht werden kann, hat dies eines hohe Aussagekraft für die Anwendung von KI in anderen Bereichen, in denen die Bedingungen weniger ideal sind. Um nur ein Beispiel zu nennen: Wenn sich die Halluzinationen von LLMs trotz der im SE vorgefundenen, idealen Voraussetzungen nicht abstellen lassen, dann dürfte dies ein gravierendes Problem auch für andere Anwendungskontexte solcher Systeme sein.
2. Die Begleitung des automatisierten SE durch die in dieser Challenge gesetzten Fragestellungen hat das Potential, den Fachkräftemangel bei gleichzeitig erhöhtem Bedarf an IT-Lösungen nicht nur theoretisch, sondern auch tatsächlich zu beseitigen, nämlich weil die Rahmenbedingungen für den Einsatz weitgehend automatisierter Verfahren klar sind.
3. Durch das Absenken der Schwelle, ab der der Einsatz von IT wirtschaftlich ist, führt günstigere SE zu größerer Teilhabe (z.B. App-Entwicklung für NGO, vgl. RE Cares).

4. Erleichterung der Kooperation und Kollaboration verschiedener Stakeholder mit verschiedenen Hintergründen durch eine Übersetzung von Fragen und Antworten in die jeweiligen Fachsprachen, gegebenenfalls sogenannte automatische Moderation (z. B. zur Erkennung und Beseitigung widersprüchlicher Anforderungen)
5. Anheben der Qualitätsstandards von „Schatten-IT“ (z.B. Excel-Sheets, die von Domänenexperten ohne Ausbildung im Bereich IT entwickelt werden, trotzdem aber die Qualitätsmerkmale professioneller IT aufweisen)

Die Entwicklung hin zu einer starken Automatisierung der Software-Entwicklung wird stufenweise verlaufen (vgl. Zielszenarien). Selbst falls nicht alle oben diskutierten Stufen erreicht werden, wird Fortschritt in diese Richtung in jedem Fall die Produktivität in der Software-Entwicklung erhöhen. Hoffentlich (und wenn richtig eingesetzt) kann er auch die Qualität und Verlässlichkeit von Software verbessern und die stärkere Teilhabe an Software-Entwicklung durch Nicht-Expert*innen ermöglichen. Fortschritt entlang dieser Level wird daher der Innovationskraft der Gesellschaft zugutekommen, die Erstellung von Software demokratisieren und die Digitalisierung vorantreiben.

Über die Autor*innen

Prof. Dr.-Ing. Anne Koziolek vom GI-Fachbereich Softwaretechnik (FB SWT) ist Professorin am Karlsruher Institut für Technologie (KIT) und forscht insbesondere zur Verwendung von Large Language Models für Tracelink Recovery und Inkonsistenzerkennung.

Prof. Dr. Kurt Schneider, Sprecher des FB SWT, ist Professor an der Leibniz Universität Hannover und forscht zu Requirements Engineering, Softwarequalität, Apps und Web Engineering sowie Informationsflussanalyse.

Prof. Dr. Friedrich Steimann vom GI-Fachbereich Softwaretechnik und dem GI-Querschnittsfachausschuss Modellierung ist Professor an der Fernuniversität in Hagen und forscht zu objektorientierte Programmierung, Softwaremodellierung und Programmiersystemen.

Prof. Dr. Walter Tichy, FB SWT, ist Professor an der Kutaisi International University (KIU), Georgien, und Prof. emeritus am Karlsruher Institut für Technologie (KIT) war Professor am Karlsruher Institut und forscht insbesondere zu Sprachverarbeitung für Aufgaben des Software Engineering.

Unterstützende GI-Gliederungen und Mitautor*innen

Diese Grand Challenge wurde formuliert und koordiniert von Mitgliedern des Fachbereichs Softwaretechnik. Beteiligt werden könnten und sollten weiterhin die Fachbereiche KI, Mensch-Computer-Interaktion, Wirtschaftsinformatik sowie Informatik und Gesellschaft.

Mitautor*innen

- Prof. Dr. Gregor Engels (Universität Paderborn)
- Prof. Dr. Sabine Glesner (TU Berlin)
- Prof. Dr. Florian Matthes (TU München)
- Prof. Dr. Ralf Reussner (Karlsruher Institut für Technologie)

- Prof. Dr. Klaus Schmid (Universität Hildesheim)

Unterstützer*innen

- Prof. Dr. Colin Atkinson (Universität Mannheim)
- Prof. Dr. Steffen Becker (Universität Stuttgart)
- Prof. Dr. Thorsten Berger (Ruhr-Universität Bochum)
- Prof. Dr. Dirk Beyer (Ludwig-Maximilians-Universität München)
- Prof. Dr. Eric Bodden (Universität Paderborn)
- Prof. Dr. Bernd Brügge (TU München)
- Prof. Dr. sc. Maria Christakis (TU Wien)
- Prof. Dr. Michael Felderer (Deutsches Zentrum für Luft- und Raumfahrt (DLR) und Universität zu Köln)
- Prof. Dr. Martin Glinz (Universität Zürich)
- Prof. Dr. Falk Howar (TU Dortmund)
- Prof. Dr. Jan Jürjens (Universität Koblenz)
- Jun.-Prof. Dr. Verena Klös (TU Dresden)
- Prof. Dr. Samuel Kounev (Universität Würzburg)
- Dr. Heiko Koziolek (ABB Corporate Research)
- Prof. Dr. Leen Lambers (BTU Cottbus-Senftenberg)
- Prof. Dr. Anna-Lena Lamprecht (Universität Potsdam)
- Prof. Dr. Stefan Leue (Universität Konstanz)
- Prof. Dr. Daniel Mendez (fortiss GmbH)
- Prof. Dr. Mira Mezini (TU Darmstadt)
- Prof. Dr. Barbara Paech (Universität Heidelberg)
- Prof. Dr. sc. Michael Pradel (Universität Stuttgart)
- Prof. Dr. Rick Rabiser (Johannes Kepler Universität Linz)
- Prof. Dr. Albrecht Schmidt (Ludwig-Maximilians-Universität München)
- Jun.-Prof. Maike Schwammberger (Karlsruher Institut für Technologie)
- Prof. Dr. Janet Siegmund (TU Chemnitz)
- Prof. Dr. Norbert Siegmund (Universität Leipzig)
- Prof. Dr. Wolfgang Reif (Universität Augsburg)
- Prof. Dr. Gabriele Taentzer (Universität Magdeburg)
- Prof. Dr. Matthias Tichy (Universität Ulm)
- Prof. Dr. Andreas Vogelsang (Universität Köln)
- Prof. Dr. Stefan Wagner (Universität Stuttgart)
- Prof. Dr. Manuel Wimmer (Johannes Kepler Universität Linz)

Weiterführende Literatur

Rao, N., Jain, K., Alon, U., Le Goues, C., Hellendoorn, V.J. (2023, October). CAT-LM: Training Language Models on Aligned Code And Tests. In Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (to appear, preprint via https://conf.researchr.org/home/ase-2023)

Trier, M., Kundisch, D., Beverungen, D., Müller, O., Schryen, G., Mirbabaie, M., & Trang, S. (2023). Digital Responsibility: A Multilevel Framework for Responsible Digitalization. Business & Information Systems Engineering, 65(4), 463-474.