Projektbericht - The Agentcy

Eric Echtermeyer[1], *Lasse* Friedrich[2], *Benedikt* Prisett[3], and *David* Schäfer[4]

[1] *6373947*
[2] *9924680*
[3] *5709658*
[4] *7086451*

Abstract. Im Rahmen unseres Projektes “The Agentcy” haben wir ein kollaboratives Multi-Agent-System entwickelt, welches in der Lage ist, selbstständig kleine Webanwendungen, auf Basis von frei wählbaren Benutzeranforderungen, zu entwicklen. Die verschiedenen Agents basieren auf modernen LLMs und interagieren innerhalb eines mehrstufigen Entwicklungsprozesses miteinander. Dieser umfasst eine initiale Anforderungsanalyse, den Entwurf eines Datenbankschemas, die Implementierung des Backends und die Entwicklung eines Frontends. Durch Kommunikation, Spezialisierung und den Einsatz einer Docker-Testumgebung sind die Agents in der Lage, in dem von uns entwickelten System die komplexe Aufgabe der Full-Stack-Webentwicklung zu lösen.

1 Einleitung

Seit der breiten Verfügbarkeit und enormen Leistungssteigerung im Bereich der generativen KI, welche vor allem durch die Veröffentlichung von ChatGPT 3.5 von OpenAI Ende 2022 ausgelöst wurde, finden die neuen Fähigkeiten von LLMs in zunehmend mehr Produkten Anwendung. Eine besondere Stärke von LLMs ist die Fähigkeit zur Generierung von und dem Arbeiten mit Code. Aufbauend auf dieser Stärke haben wir das Projekt "Multi-Agent-Dev" zum einfachen und automatisierten Generieren von kleineren Full-Stack-Webanwendungen ins Leben gerufen. Das Ziel des Projektes ist es, ausgehendend von einer sprachlich im Dialog übermittelten Anforderungsanalyse, mittels mehrerer LLM-Agents eine Website mit Datenbankstruktur, Backend und Frontend generieren zu lassen. Diese Grundidee findet im Markt bereits ersten Anklang, wie in Abschnitt 2 dargestellt, und soll durch diese Arbeit erweitert und evaluiert werden.

2 Verwandte Arbeit

Die Idee, generative KI für die Entwicklung von Software zu benutzen, ist nicht neu und findet zum einen bereits als Hilfestellung im Alltag eines Softwareentwicklers Anwendung, wird nun aber auch in Projekten verwendet, bei denen der vollständige Entwicklungsprozess von LLMs übernommen wird. Ein bekanntes Framework hierfür ist das Open-Scource-Projekt ChatDev [1], in welchem LLM-Agents gemeinsam eine virtuelle Softwarefirma bilden und so selbständig Softwareprojekte entwickeln können. Es gibt hier eine Reihe an verschiedenen Agents, welche Rollen zur Planung, Design, Entwicklung und dem Testen abbilden und während des Entwicklungsprozesses miteinander kommunizieren, um verschiedene Aufgaben zu lösen.

Neben Projekten wie ChatDev gibt es weitere Frameworks wie AutoGen von Microsoft, die für Muti-Agent-Konversationssysteme entwickelt wurden [2]. Im Gegensatz zum vorherigen Beispiel ist AutoGen allgemeiner für verschiedene Mulit-Agent-Systeme verwendbar und nicht auf die Softwareentwicklung beschränkt. Nutzer können individuell anpassbare Agents erstellen und diese miteinander kommunizieren lassen, um Aufgaben zu lösen.

Unser Projekt ist in dessen Grundzügen von den bereits existierenden Lösungen inspiriert, allerdings ohne diese Frameworks zu verwenden. Sämtliche Interaktionen wurden grundlegend neu implementiert und einzig auf LangChain zurückgegriffen.

3 Grundlage Agents

Unter einem LLM-Agent im Allgemeinen versteht man ein KI-System, welches auf einem LLM basiert, jedoch Fähigkeiten jenseits der einfachen Textgenerierung aufzeigt. Hierzu gehören bspw. das bis zu einem gewissen Grad autonome Führen von Unterhaltungen, Lösen von Aufgaben und "Denken". Ein Agent hat hierbei eine vorher festgelegte Persona, welche dessen Eigenschaften und Fähigkeiten definiert. Diese Persona wird anschließend um weitere konkrete Anweisungen mit notwendigen Informationen ergänzt, um gegebene Aufgaben zu erfüllen. Systeme, welche aus mehreren Agents bestehen, erlangen durch das Arbeiten miteinander die Fähigkeit, Fehler und Missverständnisse erkennen und korrigieren zu können. Diese Arbeitsteilung ermöglicht

somit das Abarbeiten deutlich komplexerer Aufgabenstellungen, im Vergleich zu einem allein agierenden LLM.

4 Projektimplementation

4.1 Aufbau und Architektur

Der von den Agents ausgeführte Entwicklungsprozess lässt sich zunächst in vier abgegrenzte Phasen unterteilen: Anforderungsanalyse, Entwurf des Datenbankschemas, Implementation der Backendfunktionalitäten und die Entwicklung des Frontends. Innerhalb dieser Phasen sind verschiede Agents tätig, die jeweils individuell angepasste Personas haben und daher darauf spezialisiert sind, bestimmte Arten von Aufgaben auszuführen. Durch Konversationen miteinander und die Weitergabe von Informationen arbeiten sie gemeinsam an der Implementierung der Webapplikation, korrigieren Fehler und stellen somit die funktionierende Integration der verschiedenen Softwareschichten sicher. Eine Übersicht dieses Prozesses ist Abbildung 1 zu entnehmen.

Als den Agents zugrunde liegendes LLM wurde sich in unserem Projekt, für das von OpenAI veröffentlichte Modell GPT 3.5 Turbo entschieden. Es handelt sich hierbei um ein komplexes Sprachmodell aus der Generative Pretrained Transformer (GPT) [3] Serie, dessen Funktionalität von OpenAI über eine API für sehr geringe Kosten zur Verfügung gestellt wird. Da es schnelle Texterstellungen ermöglicht und sehr gut für die Generierung von Code sowie natürlicher Sprache geeignet ist, bietet es eine zuverlässige Grundlage für unser Projekt. Es wurde zudem auch der Einsatz des Modells GPT 4 evaluiert, um den Nutzen eines noch leistungsstärkeren, aber auch teureren Modells zu vergleichen. Von der Nutzung eines anderen bspw. Open-Source-Modells wurde zunächst abgesehen, da viele andere Modelle nicht in der Lage sind, derartig komplexe Aufgaben mit vielschichtigen Anweisungen zu erfüllen.

Einige der ersten und wichtigsten Aufgaben übernimmt der Orchestrator-Agent, welcher als Product Owner agiert. Der Orchestrator ist dafür zuständig, die Anforderungen des Useres zu erfassen und aus diesen anschließend konkrete Aufgaben für die drei folgenden Entwicklungsphasen abzuleiten.

Die Anforderungsanalyse beginnt mit einer Abfrage, was für eine Webapplikation entwickelt werden soll. Der Benutzer antwortet mit einer kurzen Beschreibung seiner Vorstellungen und ausgehend hiervon stellt der Orchestrator Fragen, um die genaueren Anforderungen weiter zu erfassen. Sobald er zu dem Entschluss kommt, alle Anforderungen verstanden zu haben, schreibt er eine prägnante Zusammenfassung und geht in den nächsten Arbeitsschritt über. Anhand dieser ersten Aufgaben lässt sich bereits die allgemeine Funktionsweise der Agents verdeutlichen. Der Orchestrator hat zunächst eine über einen ersten Prompt definierte Persönlichkeit, welche beschreibt, dass er ein Entwicklungsprojekt leitet und als Product Owner agiert. Um die Anforderungsanalyse auszuführen, erhält er über einen weiteren Prompt im Folgenden die konkrete Aufgabe, die Anforderungen des Benutzers zu verstehen und explizit Verständnisfragen an diesen zu stellen. Auch das Ausgabeformat im JSON-Format wird ihm in diesem Prompt vorgegeben, um sicherzustellen, dass die Antworten von unserem Framework weiterverarbeitet werden können. Das Erstellen von Aufgaben für die drei Entwicklungsbereiche stellt eine weitere separate Aufgabe dar, die der Orchestrator ausführen kann, und wird ihm entsprechend über einen Prompt aufgetragen, sobald er die Entscheidung getroffen hat, alle Anforderungen verstanden zu haben. Dieses Prinzip der Personas und konkreten Aufgaben für verschiedene Arbeitsschritte lässt sich auch bei den folgenden Agents finden.

Nachdem die initiale Anforderungsanalyse abgeschlossen ist, stößt der Orchestrator die Entwicklung an, indem er die entsprechende Aufgabe an den Datenbankentwickler überträgt. Ausgehend von diesen übermittelten Anforderungen schreibt der Entwickler einen ersten Entwurf für das Datenbankschema und gibt dieses an den Tester der Datenbank weiter. Der Tester ist nun dafür verantwortlich, zu überprüfen, ob Fehler im Code zu finden sind und eine Rückmeldung an den Entwickler zu geben. Diese beiden Agents treten somit in eine Kommunikationsschleife und der Code wird so lange optimiert, bis er vom Tester akzeptiert wird. Den Abschluss der Datenbankentwicklung übernimmt der Dokumentator, welcher ausgehend von dem finalen Code eine detaillierte Dokumentation schreibt. Die Dokumentation enthält einen grundlegenden Überblick über den erstellten Code und ist wichtig, damit der nächste Agent implementierte Strukturen verwenden kann.

Es folgt im weiteren Verlauf die Entwicklungsphase der Backendfunktionalitäten. Der Entwickler für das Backend erhält vom Orchestrator eine auf den Anforderungen basierende Aufgabe sowie die Dokumentation des Datenbankschemas. Nun muss dieser ein Python-Script schreiben, welches die Datenbanktabellen erzeugt und Lese- und Schreibzugriffe für das Frontend ermöglicht. Dieser Code wird anschließend dem Backend-Tester übermittelt. Neben der einfachen Weitergabe des Codes wird dieser zudem in einem Docker-Container ausgeführt, um seine Funktionalität sicherzustellen, indem die Logs geprüft werden. Eine tiefere Erklärung dieses Features ist in Kapitel 4.2 zu finden.

Der Tester erhält somit den Code des vorherigen Entwicklers sowie die Logs aus dem Docker-Container. Er hat nun die Aufgabe, eine Rückmeldung zu geben, ob ein Fehler gefunden wurde bzw. der Code nicht ausführbar ist. Anhand dieser Rückmeldung folgt wie in der vorherigen Entwicklungsphase eine Optimierungsphase, um iterativ alle Fehler des Codes zu beseitigen. Abschließend schreibt der Backenddokumentator eine finale Dokumentation.

Die letzte Entwicklungsphase beginnt erneut mit einem, nun für das Frontend zuständigen Entwickler, welcher auf Grundlage der Aufgabe des Orchestrators sowie der vorherigen Dokumentation der Backendfunktionalitäten den Code für das Frontend entwirft. Es wurde sicher hierbei dazu entschieden, den CSS- und Javascript-Code direkt im HTML zu integrieren, um eine einfachere Ausführung zu ermöglichen. Das Testen des Codes

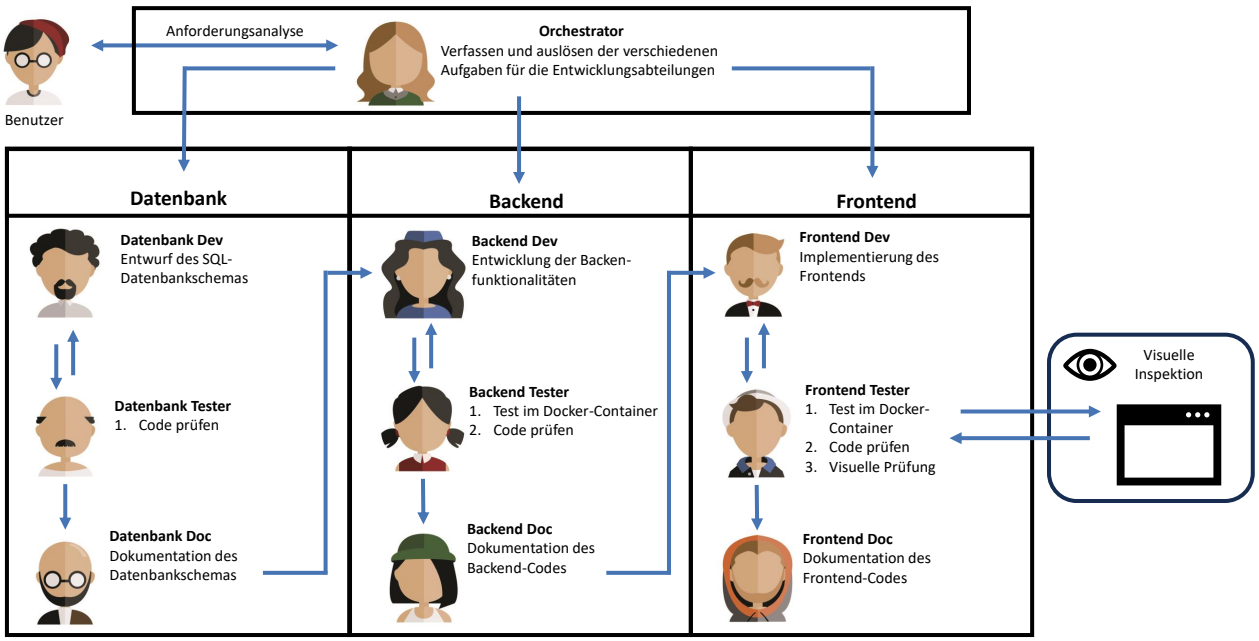

Figure 1. Entwicklungsprozess mit Aufgaben der beteiligten Agents.

erfolgt analog zur vorherigen Entwicklungsphase über einen auf das Frontende spezialisierten Tester, welcher den Code sowie die Logs der Ausführung im Docker-Container überprüft und so die iterative Verbesserung des Frontends ermöglicht. Neben der einfachen Überprüfung des Codes findet hier aber auch eine visuelle Überprüfung des Frontends statt. Sobald der Code ausführbar ist, wird mittels des GPT-4-Vision-Modells geprüft, ob die Seite den Anforderungen entspricht und gegebenenfalls weitere Änderungen vom Entwickler eingefordert. Ist dieser Verbesserungszyklus abgeschlossen, erstellt ein weiterer Dokumentator eine finale Dokumentation dieser Softwareschicht, welche zwar im Entwicklungsprozess der Agents nicht mehr weitergegeben wird, aber wie auch die andere Dokumentation von dem finalen Endbenutzer verwendet werden kann, um selber noch Anpassungen vorzunehmen. Nach dem Abschluss des Entwicklungsprozesses steht die Applikation dem Anwender über einen Docker-Container direkt zum Testen und Nutzen zur Verfügung.

Durch diesen vielschichtigen Entwicklungsprozess mit mehreren auf bestimmte Bereiche spezialisierten Agents ist es nun möglich, kleinere Webanwendungen innerhalb von kürzester Zeit zu entwickeln, ein Prozess, der andernfalls mit deutlich größerem Aufwand und notwendigem Fachwissen verbunden wäre.

4.2 Ausführung mit Docker

Ein wesentliches Merkmal unseres Projekts, welches es von bestehenden Lösungen wie den in Kapitel 2 beschriebenen Frameworks ChatDev und AutoGen unterscheidet, ist der Einsatz von Docker zur Ausführung einzelner Softwareebenen. Dies ermöglicht das Testen des Codes bereits während der Entwicklungsphase, anstelle sich lediglich auf eine erweiterte Syntaxprüfungen durch die Sprachmodelle zu verlassen. Zu diesem Zweck werden verschiedene Code-Segmente in unterschiedlichen Container-Images ausgeführt. Die Struktur dieser Images ist relativ strikt festgelegt, darunter immer ein Python-basierter Backend-Container, ein Nginx-Frontendserver und ein Postgres-Datenbankcontainer. Der in diesen Containern ausgeführte Code ist jedoch flexibel wählbar und wird iterativ im Hauptprogramm generiert. Dies ermöglicht nicht nur den bisherigen sequenziellen Programmablauf, sondern auch die Bereitstellung mehrerer unabhängiger Ressourcen, die dauerhaft verfügbar sind.

5 Diskussion

5.1 Evaluierungsmethode

Zur Evaluierung des entwickelten Multi-Agent-Systems wurden von uns 10 Beispiele für mögliche Anforderungen ausformuliert und für diese jeweils der Entwicklungsprozess gestartet. Es wurden zwei unterschiedliche Szenarien getestet. Hierbei wurden während des ersten Szenarios alle Agents basierend auf GPT-3.5-Turbo initiiert und beim zweiten Szenario GPT-4-Turbo Modelle für das Testen der Codes verwendet. Zudem wurde hierbei auch GPT-4-Vision für das Testen des Frontend benutzt. Im Testdurchlauf wurden verschiedene Metriken erhoben und am Ende Durchschnittswerte gebildet. Zu diesen gehören die Laufzeit des gesamten Entwicklungsprozesses, die Anzahl der Kommunikationsschritte zwischen Entwickler und Tester sowie ein abschließender menschlicher Feedbackscore, welcher das Design und die Funktionalität bewertet. Dieser menschliche Feedbackscore wurde in kollaborativen Analysesitzungen erhoben und hat eine Skala von 1 bis 5, wobei Zweiteres das beste ist.

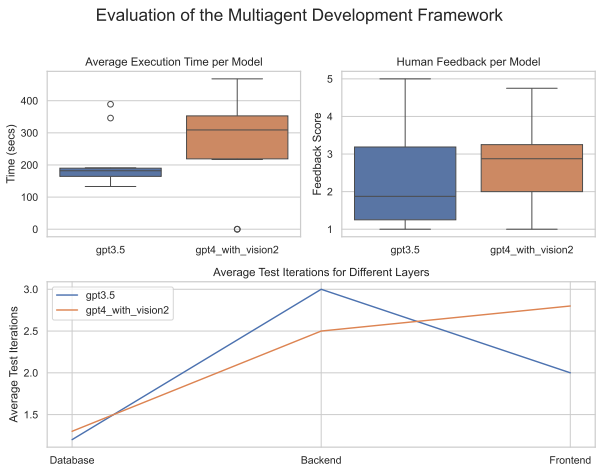

Figure 2. Caption of the Figure 1. Below the figure.

5.2 Auswertung

Die Ergebnisse der Evaluierung, dargestellt in 2, zeigen die allgemeine Performance des Systems sowie einen Vergleich zwischen dem Einsatz des Modells GPT 3.5 und GPT 4 (mit Vision für das Frontend). Aus den Laufzeiten ist zu erkennen, dass die Ausführung mit GPT 3.5 deutlich schneller ist und eine geringe Varianz aufweist. Die Iterationsanzahl zwischen dem Entwickler und Tester stellt dar, wie viele Verbesserungen im Durchschnitt auf jeder Entwicklungsebene notwendig waren. Für die Datenbankebene ist zu erkennen, dass für beide Modelle lediglich sehr wenige Iterationen notwendig waren. Dies verdeutlicht, dass beide Modelle sehr gut in der Lage sind, SQL-Code zuverlässig zu generieren und entsprechend wenig Fehler machen, welche verbessert werden müssen. In der Backendschicht findet der Tester jedoch deutlich mehr Dinge, die ausgebessert werden müssen. Besonders aufällig ist hierbei, dass die Testläufe mit GPT 3.5 im Vergleich zu GPT 4 eine deutlich höhere Iterationsanzahl aufeisen. Dies lässt sich damit begründen, dass GPT 3.5 älter und leistungsschwächer im Vergleich zu GPT 4 ist und mit der komplexen Aufgabe der Backendentwicklung größere Probleme hat. Bei der Frontendentwicklung hat wieder GPT 4 eine höhere Anzahl an Iterationen. Dies hängt jedoch damit zusammen, dass hier zusätzlich das Vision-Modell eingesetzt wurde, um zusätzlich eine visuelle Bewertung vorzunehmen. Der von uns vergebene menschliche Feedbackscore zeigt, dass die durch GPT 4 erstellten Webapplikationen im Durchschnitt besser sind, als wenn diese mit GPT 3.5 erstellt werden. Es ist jedoch anzumerken, dass letzteres Modell eine höhere Varianz aufzeigt und in der Lage ist, zwar weniger aber dafür auch durchaus gute Seiten zu erstellen.

Allgemein wird aus unserer Evaluation deutlich, dass die meisten Schwachstellen in den Ergebnissen aus der Schnittstelle zwischen dem Frontend und Backend entstehen. In vielen Fällen sind die fertigen Applikationen leider oft nicht in der Lage, Daten korrekt in die Datenbank zu schreiben oder sie aus dieser zu lesen.

6 Abschluss

Mit unserem Projekt “The Agentcy” haben wir erfolgreich ein auf kollaborativen LLM-Agents basierendes Programm entwickelt, welches in der Lage ist, mit einer einfachen Beschreibung der allgemeinen Anforderungen, kleinere Webanwendungen selbständig zu entwicklen und diese dem Benutzer direkt zum Testen bereitzustellen. Durch die Einteilung des Entwicklungsprozesses in mehrere Phasen und den Einsatz von verschiedenen auf bestimmte Bereiche spezialisierten Agents werden so die bereits ausgeprägten Fähigkeiten von LLMs erweitert, um noch komplexere Aufgaben zu lösen. Besonders durch das Testen von einzelnen Komponenten mit Docker hebt sich unser Projekt auch von anderen bereits bestehenden Frameworks ab. Es ist allerdings ebenfalls hervorzuheben, dass die Zuverlässigkeit in den Antworten der Agents noch immer eine große Herausforderung darstellt.

7 Arbeitsaufteilung

Unser Projekt wurde gemeinschaftlich erarbeitet und alle Gruppenmitglieder haben sich eingebracht. Fokusgebiete in der Arbeitsaufteilung können jedoch wie folgt gesetzt werden:

- Eric: Pipeline, Auswertung
- Lasse: Docker, Vision
- Benedikt: Prompting, Parser, Report
- David: User-Interface, Pipeline, Prompting

References

[1] C. Qian, X. Cong, W. Liu, C. Yang, W. Chen, Y. Su, Y. Dang, J. Li, J. Xu, D. Li et al., *Communicative agents for software development* (2023), 2307.07924

[2] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu et al., *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework* (2023), 2308.08155

[3] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., *Language models are few-shot learners* (2020), 2005.14165