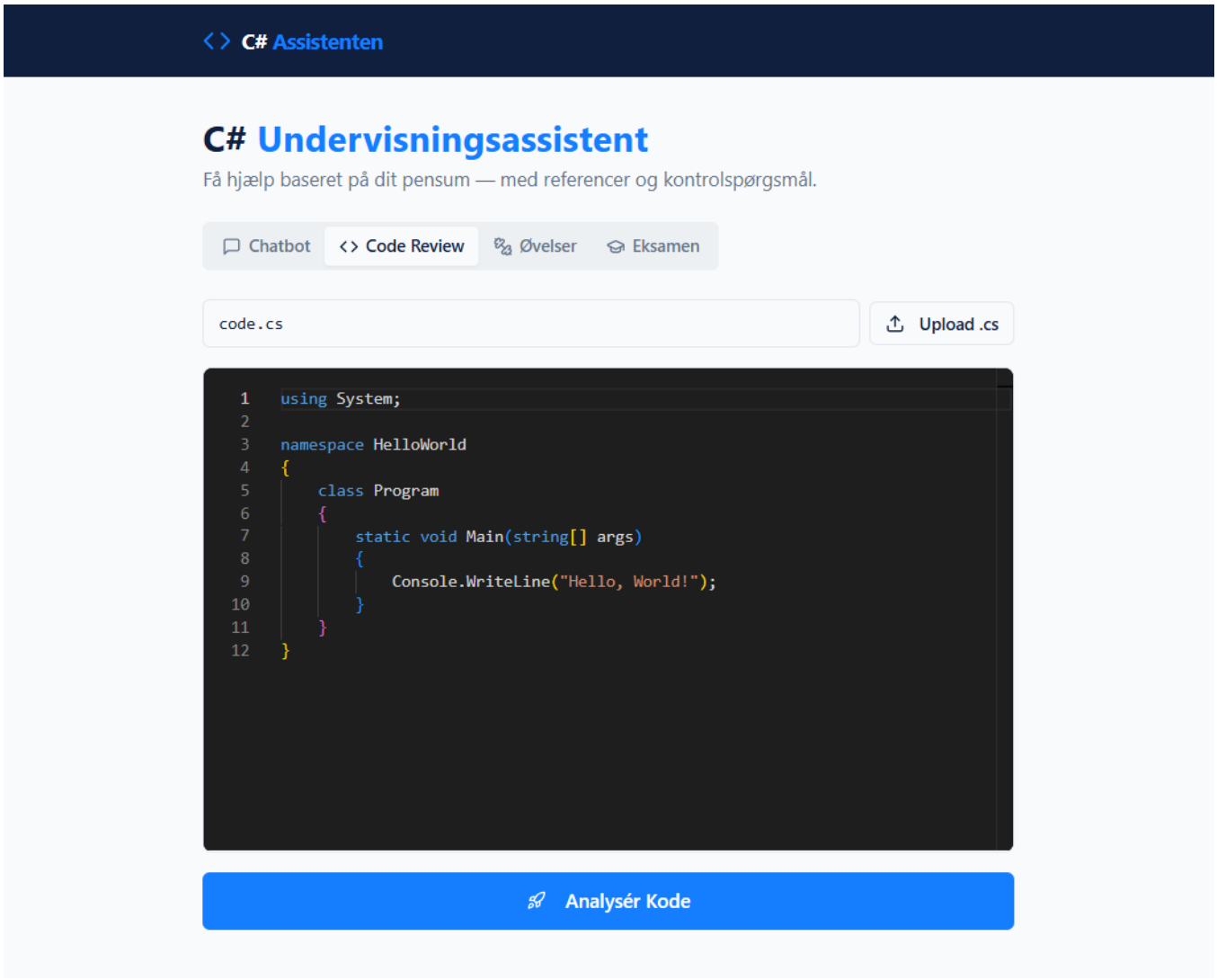
Sokratisk AI-vejledning som didaktisk ressource

En designbaseret undersøgelse af generativ AI i introducerende programmeringsundervisning



Masteruddannelsen i IKT og Læring (MIL), Aalborg Universitet
Masterprojekt, 15 ECTS, 4. semester
Studerende: Tue Hellstern (100261-0317)
Vejleder: Andreas Lindenskov Tamborg
Antal tegn: 107.451

Abstract

This project investigates how generative artificial intelligence can be designed as a didactic resource in introductory programming education. Grounded in a Design-Based Research (DBR) approach, the study develops and evaluates a Socratic AI agent aimed at supporting students' reflection and problem-solving processes rather than providing complete solutions.

The study is conducted as an exploratory DBR-inspired first iteration, in which a web-based AI agent based on large language models is developed and implemented in a first-semester programming course on Danish AP degree program in computer science. The target group is students with limited or no prior programming experience. The agent is designed based on principles of Socratic dialogue, curriculum alignment through retrieval-augmented generation, and a modular learning structure. The empirical data consist of questionnaire responses from 12 students and three semi-structured interviews.

The findings indicate that students generally perceive the agent as a meaningful learning tool, particularly in supporting reflection and guided problem-solving. However, the analysis shows that the effectiveness of Socratic AI guidance is contingent on a set of conditions. Three key mechanisms are identified: (1) sequencing between students' own cognitive production and dialogic guidance, (2) verification burden among novices, where limited prior knowledge makes it difficult to assess AI-generated responses, and (3) the agent's role within a broader AI ecosystem, in which students use multiple parallel tools.

Based on these findings, the project derives a set of conditional design principles for Socratic AI-based guidance: sequencing of student production and dialogue, differentiated scaffolding, consideration of verification burden, curriculum alignment, and awareness of the broader AI ecosystem. The project contributes both a concrete didactic design and an empirically grounded specification of the conditions under which Socratic AI guidance supports learning in programming education.

Indholdsfortegnelse

1 Introduktion 5

2 Problemformulering 6

3 Baggrund og motivation 6

4 Litteraturreview 8

4.1 Generativ AI i programmeringsundervisning 9

4.2 Designbaseret forskning og didaktiske interventioner 10

4.3 Syntese 10

5 Teoretisk ramme 12

5.1 Den sokratiske metode i AI-baseret vejledning 12

5.2 Kognitiv belastningsteori og programmeringslæring 13

6 Design og udvikling af AI-agent 14

6.1 Vidensdatabase og pensumafgrænsning 14

6.1.1 Dokumentstruktur: PDF og tilhørende JSON-metadata 15

6.1.2 Pensumafgrænsning som didaktisk og teknisk princip 16

6.2 Systemarkitektur 16

6.2.1 Frontend 16

6.2.2 Backend 17

6.3 To-model AI-pipeline med retrieval-augmented generation 17

6.3.1 Trin 1: RAG-søgning 17

6.3.2 Trin 2: Strukturering via OpenAI GPT-4o mini 17

6.3.3 Trin 3: Pædagogisk formattering via Claude Sonnet 18

6.4 Agentens fire moduler 18

6.4.1 Chatbot-modulet 18

6.4.2 Kodereview-modulet 19

6.4.3 Øvelsesmodulet 19

6.4.4 Eksamenstræningsmodulet 19

6.5 Sikkerhed og adgangskontrol 19

6.6 Centrale designvalg og didaktisk begrundelse 20
6.7 Agentens organisatoriske integration som åbent designvalg 21
6.8 Design sammenfatning 21
7 Metode 22
7.1 Eksplorativ DBR-inspireret første iteration 22
7.2 Validitet og forskerposition 23
7.3 Reproducerbarhed og dataadgang 24
7.4 Datagrundlag 24
7.5 Syntese af spørgeskemadata 29
7.6 Interview 30
7.6.1 Udvælgelse af interviewdeltagere 31
7.6.2 Interviewramme 32
7.6.3 Tematisk analyse 34
7.6.4 Interviewanalyse 34
8 Diskussion 41
8.1 Projektets bidrag i forhold til den eksisterende litteratur 41
8.2 Perspektivering ud over datamatikerkonteksten 43
8.3 Kritisk stillingtagen til metode og datagrundlag 44
8.4 Agentens samspil med undervisningskonteksten 44
8.4.1 Underviserens rolle 45
8.4.2 Studerendes parallelle AI-praksis 45
8.4.3 Integration i undervisningsforløbet 46
8.5 Implikationer for praksis 46
8.6 Implikationer for videre forskning 47
9 Konklusion 48
10 Litteraturliste 51
11 Bilagsoversigt 54

1 Introduktion

Dette projekt undersøger, hvordan generativ AI kan designes som en didaktisk ressource i introducerende programmeringsundervisning. Den empiriske kontekst er datamatikeruddannelsens 1. semester, hvor agenten er udviklet og afprøvet, mens projektets analytiske fokus ligger på udviklingen af designprincipper for sokratisk AI-vejledning som didaktisk artefakt.

Introducerende programmeringskurser udgør et afgørende fundament for Datamatiker studerendes videre faglige specialisering. Læringskurven er imidlertid stejl: studerende uden forudgående programmeringserfaring skal på én gang tilegne sig præcise syntaktiske notationer, logisk problemstrukturering og abstrakt tænkning, kompetencer der ikke lader sig erhverve uden vedvarende kognitiv indsats. Dette stiller store krav til undervisningens kapacitet til at levere individuel og differentieret støtte, som i praksis er ressourcekrævende og svær at skalere.

Fremkomsten af generative AI-systemer som ChatGPT, Claude og GitHub Copilot tilbyder en mulighed for at imødekomme dette behov, idet disse systemer er i stand til at generere funktionel kode og give umiddelbar feedback. Denne tilgængelighed indebærer imidlertid en betydelig didaktisk risiko: studerende kan anvende AI til at producere løsninger snarere end til at udvikle forståelse, hvilket potentielt underminerer de læreprocesser, som introduktionskurset er sat i verden for at understøtte. Spørgsmålet er derfor ikke, om generativ AI skal indgå i undervisningen, men hvordan den kan designes til at fremme kognitiv bearbejdning og begrebsforståelse frem for blot at levere færdige svar. Dette projekt tager udgangspunkt i netop dette spørgsmål gennem udvikling og afprøvning af en AI-baseret tutor, designet til at støtte studerendes læring via sokratisk dialog og refleksionsbaseret vejledning.

Projektet gennemføres som et designbaseret studie, hvor en konkret AI-agent udvikles og afprøves i undervisningskonteksten på datamatikeruddannelsen. Formålet er både at undersøge, hvordan en sådan teknologi påvirker studerendes læringsprocesser, og at udvikle didaktiske designprincipper for ansvarlig integration af AI i programmeringsundervisning.

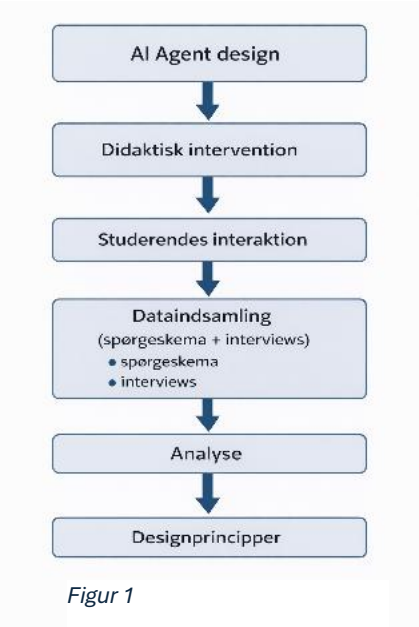
2 Problemformulering

På baggrund af den aktuelle udvikling inden for generativ AI og de didaktiske udfordringer i introducerende programmeringsundervisning undersøger dette projekt følgende problemstilling:

Hvordan kan en sokratisk AI-agent designes til at understøtte studerendes refleksion og problemløsning, og hvordan oplever studerende uden programmeringserfaring interaktionen med en sådan agent i introducerende programmeringsundervisning på datamatikeruddannelsen?

For at besvare denne problemformulering arbejder projektet med følgende underspørgsmål:

- Hvordan oplever studerende anvendelsen af en sokratisk AI-agent som støtte i arbejdet med programmeringsopgaver?
- Hvilke designprincipper for sokratisk AI-baseret vejledning kan udledes på baggrund af projektets design og empiriske erfaringer?



Figur 1

Projektet følger en iterativ designbaseret tilgang, der tager udgangspunkt i udviklingen af selve AI-agenten. Som illustreret i Figur 1 udgør AI Agent design det første trin, hvor agentens sokratiske vejledningsprincipper fastlægges og omsættes til konkrete prompt-strukturer og interaktionslogikker.

Det færdige design afprøves herefter i en didaktisk intervention, hvor agenten integreres i et konkret undervisningsforløb på 1. semester af Datamatikeruddannelsen. I denne fase interagerer de studerende med agenten som led i deres arbejde med programmeringsopgaver, hvilket udgør projektets primære empiriske grundlag.

3 Baggrund og motivation

Gennem mere end 15 år har jeg undervist på datamatikeruddannelsen på KEA, Cphbusiness og senest på online datamatikeruddannelsen på UCL. I denne periode har jeg oplevet en kontinuerlig teknologisk udvikling, men fremkomsten af generativ kunstig intelligens adskiller sig ved sin hastighed og sin gennemgribende betydning for programmeringspraksis.

AI er ikke blot et nyt værktøj. Det udfordrer selve forudsætningerne for, hvordan man lærer at programmere. Når studerende kan få genereret funktionel kode på få sekunder, ændres relationen mellem proces og produkt og mellem forståelse og output.

Denne udvikling rejser for mig et centralt didaktisk anliggende: at integrere AI i undervisningen på en måde, der understøtter læring frem for at erstatte den aktive erkendelsesproces, som faglig forståelse bygger på.

Det er denne didaktiske udfordring, der ligger til grund for projektets problemformulering og den konkrete designtilgang, som udfoldes i det følgende.

Yan et al. (2024) identificerer i et systematisk scoping review 53 forskellige anvendelsesformer for store sprogmodeller i uddannelse, fra automatiseret feedback til indholdsgenerering og peger på, at feltet udvikler sig med en hastighed, der langt overstiger de tilgængelige didaktiske modeller.

Datamatikeruddannelsen er professionsrettet og forpligtet til at afspejle den praksis, de studerende møder i erhvervslivet. AI-kompetencer er allerede en del af softwareudviklerens arbejdsfelt, både som teknologisk område og som udviklingsværktøj. Som uddannelsesinstitution kan vi derfor ikke vælge at ignorere AI. Studerende anvender teknologien allerede, uanset om vi didaktisk rammesætter den eller ej, en tendens der er dokumenteret bredt i litteraturen om generativ AI i uddannelse (Noroozi et al., 2024). **Spørgsmålet er ikke, om AI skal indgå i undervisningen, men hvordan det skal gøre det.**

Samtidig er det veldokumenteret i forskningen, at programmeringsundervisning for begyndere er kognitivt krævende (Paas et al., 2003). Studerende kæmper med abstraktion, algoritmisk tænkning og fejlfinding, og i onlinekonteksten forstærkes disse udfordringer af en høj grad af selvstændighed og begrænset umiddelbar feedback. Læring i programmering forudsætter, at den studerende gradvist opbygger mentale modeller og udvikler evnen til selvstændig problemløsning.

AI kan her fungere som en potentielt stærk støtte, en tilgængelig forklaringspartner og sparringsressource, der kan bidrage med stilladsering i situationer, hvor den studerende befinder sig på grænsen af sin aktuelle forståelse (jf. Wood, Bruner & Ross, 1976; Vygotsky, 1978). Samtidig kan AI reducere oplevet kompleksitet og dermed aflaste den kognitive belastning (Paas et al., 2003). Men AI kan også skabe en illusion af mestring, hvor koden fungerer, uden at forståelsen nødvendigvis er udviklet en problematik der er dokumenteret i nyere forskning, som viser, at studerende i programmeringsundervisning oplevede AI-genereret feedback som mindre meningsfuld end instruktørfeedback (Er et al., 2024). Hvis den studerende springer den

analytiske proces over, risikerer teknologien at svække netop den begrebsdannelse, som undervisningen skal understøtte.

Denne dobbelte erfaring af AI som både mulighed og risiko udgør projektets centrale motivation. Jeg ønsker ikke blot at analysere fænomenet, men at arbejde systematisk og undersøgende med, hvordan AI kan integreres didaktisk ansvarligt i online programmeringsundervisning for begyndere.

Valget af en Design Based Research (DBR) tilgang hænger sammen med min position som praktiserende underviser. Problemstillingen udspringer direkte af min egen undervisningskontekst og de konkrete udfordringer, jeg møder i praksis. DBR giver mulighed for at udvikle, afprøve og justere et konkret AI-didaktisk design i undervisningen og samtidig generere principper, der kan bidrage til en teoretisk informeret forståelse af AI som læringsressource (Design-Based Research Collective, 2003, Educational Researcher). Givet at der allerede foreligger et voksende felt af empirisk forskning på området, understøtter DBR-tilgangen desuden ambitionen om at udvikle det, Barab og Squire (2004) betegner som en "ydmyg teori" (humble theory) – en teoretisk forståelse der er lokalt forankret i den specifikke læringsøkologi på datamatikeruddannelsen, frem for at stræbe efter universelle generaliseringer (Barab & Squire, 2004, Journal of the Learning Sciences).

Projektet udspringer således af et ønske om at tage professionelt ansvar i en tid, hvor teknologien udvikler sig hurtigere end de didaktiske modeller. Målet er at bidrage til en kvalificeret integration af AI i programmeringsundervisningen med fokus på begynderens læring, forståelse og udvikling af selvstændig problemløsning.

4 Litteraturreview

Dette litteraturreview giver et indblik i hovedstrømninger og centrale indsigter inden for to forskningsområder, der på hver deres måde er relevante for projektets problemfelt: generativ kunstig intelligens i programmeringsundervisning samt designbaseret forskning som metodisk tilgang til udvikling af didaktiske interventioner. To yderligere perspektiver, den sokratiske metode som dialogisk læringsprincip og kognitiv belastningsteori som forklaringsramme for læring i programmering, introduceres i den efterfølgende teoretiske ramme, hvor de anvendes til at begrunde designet af den udviklede AI-agent. Afslutningsvis sammenfattes de centrale spændinger og forskningsmæssige åbninger, som litteraturen peger på.

Mens forskning i programmeringsundervisning og sokratisk vejledning har en længere tradition, er anvendelsen af generativ AI og LLM-baserede tutorer i uddannelseskontekster et relativt nyt og hurtigt voksende felt. En litteratursøgning i ACM Digital Library viser eksempelvis en markant

stigning i publikationer om Large Language Models (LLM) i datalogisk uddannelse siden 2022 (Becker et al., 2023). Samtidig er den empiriske forskning fortsat begrænset, særligt når det gælder didaktiske design af AI-baserede læringsværktøjer. Dette gælder i endnu højere grad studier, der undersøger anvendelsen af sokratisk designede AI-tutorer i programmeringsundervisning.

4.1 Generativ AI i programmeringsundervisning

Muligheder og potentiale

Generative AI-værktøjer baseret på Large Language Models, eksempelvis ChatGPT og GitHub Copilot, har vist sig i stand til at generere syntaktisk korrekt og funktionel kode på baggrund af natursproglige beskrivelser (Becker et al., 2023). Denne teknologiske udvikling har betydelige implikationer for introducerende programmeringskurser, hvor studerende ofte har behov for løbende vejledning og feedback i arbejdet med programmeringsopgaver.

Forskningen peger på flere potentielle fordele ved sådanne værktøjer. AI-baserede systemer kan levere øjeblikkelig og kontekstafhængig feedback på studerendes kode og dermed støtte læringsprocessen netop i de situationer, hvor den studerende møder vanskeligheder. I store introducerende kurser kan dette fungere som et supplement til den undervisning og vejledning, der traditionelt leveres af undervisere og instruktorer.

Empiriske studier indikerer desuden at samarbejde med LLM-baserede værktøjer kan have en positiv effekt på noviceprogrammørers præstationer i programmeringsopgaver (Lyu et al., 2024). Flere undersøgelser fremhæver også, at AI-værktøjer kan reducere frustration i forbindelse med fejlfinding og samtidig styrke studerendes selvtillid i arbejdet med programmering (Hashmi, Bal, & Rahman, 2024).

Samtidig understreger nyere forskning, at effekten i høj grad afhænger af den pædagogiske udformning af AI-interaktionen. Forskningen peger på, at generativ AI har størst positiv effekt på læring, når den hjælper studerende med selv at nå frem til løsninger frem for at give dem svarene direkte (Badger et al., 2026). Dette peger på betydningen af at designe AI-systemer, der understøtter aktiv refleksion frem for passiv modtagelse af svar.

Risici og begrænsninger

På trods af de potentielle fordele fremhæver litteraturen også en række væsentlige udfordringer ved anvendelsen af generativ AI i programmeringsundervisning. En ofte fremhævet bekymring er risikoen for overafhængighed af AI-værktøjer. Når studerende får adgang til systemer, der kan generere færdig kode, kan der opstå en tendens til at acceptere løsninger uden fuldt ud at forstå den underliggende logik.

Flere studier peger desuden på, at forbedringer i kortsigtede præstationer ikke nødvendigvis fører til bedre transfer (Berger et al., 2026) til nye problemstillinger. Studerende kan f.eks. have vanskeligt ved at anvende begreber og teknikker i nye kontekster uden fortsat AI-assistance.

En yderligere udfordring er de såkaldte hallucinationer (OpenAI, 2025), hvor AI-systemer genererer forklaringer eller kode, der fremstår plausible, men indeholder fejl eller misvisende information. For begyndere kan dette være særligt problematisk, da de ofte mangler den faglige erfaring, der er nødvendig for at identificere sådanne fejl.

Disse udfordringer peger på et centralt didaktisk spørgsmål: hvordan AI kan integreres i undervisningen på en måde, der understøtter læring uden at reducere den kognitive bearbejdning, der er nødvendig for udvikling af begrebsforståelse og problemløsningsevner.

4.2 Designbaseret forskning og didaktiske interventioner

Udviklingen af AI-baserede læringsværktøjer kan forstås inden for rammen af designbaseret forskning (Design-Based Research). Denne tilgang kombinerer udvikling af læringsteknologi med empiriske undersøgelser af dens anvendelse i autentiske undervisningskontekster.

Misfeldt (2010) beskriver denne tilgang gennem begrebet forestillet læringsvej, som betegner den antagelse om læring, der ligger indlejret i et didaktisk design. Når undervisere udvikler nye læringsværktøjer, indgår der implicit en hypotese om, hvordan læring forventes at finde sted. Interventionen fungerer derfor både som et forsøg på at forbedre praksis og som et empirisk grundlag for at undersøge læringsprocesser.

Denne tilgang er særligt relevant i udviklingen af AI-baserede læringsværktøjer, hvor der fortsat findes begrænset empirisk viden om deres didaktiske anvendelse. Designbaseret forskning muliggør således en iterativ udvikling af teknologiske artefakter samtidig med, at der genereres viden om, hvordan disse artefakter påvirker læringsprocesser i konkrete undervisningskontekster.

4.3 Syntese

Den gennemgåede litteratur peger samlet set på, at generativ AI har potentiale til at ændre rammerne for programmeringsundervisning markant. AI-baserede værktøjer kan tilbyde løbende støtte og feedback til studerende og dermed fungere som en tilgængelig læringsressource i situationer, hvor traditionel vejledning ikke er tilgængelig.

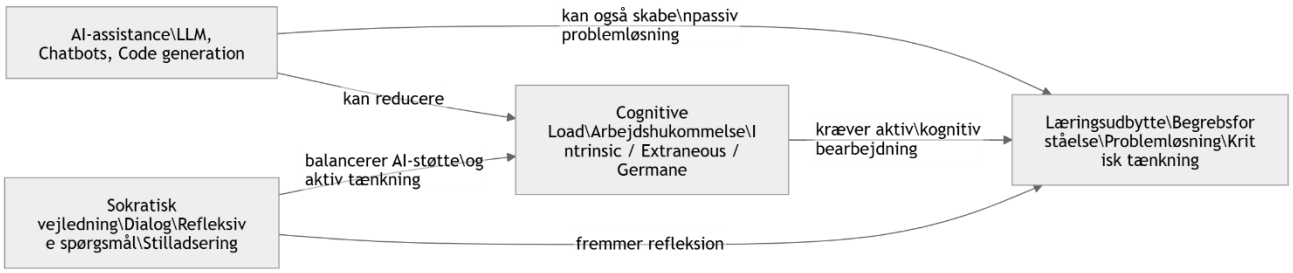
Samtidig fremhæver forskningen en række væsentlige udfordringer, herunder risikoen for overafhængighed af AI-genererede løsninger, fejlbehæftede forklaringer samt en mulig reduktion af den kognitive bearbejdning, der er nødvendig for dybere læring.

I denne sammenhæng fremstår dialogbaserede AI-systemer, inspireret af sokratiske principper, som en potentiel didaktisk strategi til at balancere støtte og selvstændig problemløsning. Ved at strukturere interaktionen som en undersøgende dialog kan AI-systemer potentielt fungere som en form for stilladsering, der både reducerer unødvendig kognitiv belastning og samtidig understøtter refleksion og begrebsdannelse.

På trods af den voksende forskningsinteresse findes der dog fortsat begrænset empirisk viden om, hvordan sokratisk designede AI-tutorer fungerer i praksis i programmeringsundervisning. Dette peger på et behov for praksisnære studier, hvor sådanne værktøjer udvikles og undersøges i autentiske undervisningskontekster.

På denne baggrund undersøger nærværende projekt, hvordan en sokratisk AI-tutor kan integreres i undervisningen på datamatikeruddannelsen, og hvordan en sådan didaktisk intervention påvirker studerendes refleksion, problemløsning og begrebsforståelse i programmering.

På tværs af litteraturen fremstår en central didaktisk spænding mellem den støtte, som AI-baserede systemer kan tilbyde studerende, og den risiko for reduceret kognitiv bearbejdning, som automatiseret problemløsning kan medføre. Figur 2 nedenfor illustrerer denne spænding og peger samtidig på den sokratiske dialog som et muligt designprincip til at balancere AI-assistance og aktiv læring.



Figur 2

Som illustreret i Figur 2, kan sokratisk vejledning forstås som et pædagogisk designprincip, der søger at balancere forholdet mellem AI-assistance og den kognitive indsats, der er nødvendig for udvikling af begrebsforståelse i programmering.

På baggrund af denne syntese fremstår særligt to teoretiske perspektiver som centrale for projektets videre analyse: den sokratiske metode som dialogisk læringsprincip og kognitiv belastningsteori som forklaringsramme for læring i komplekse domæner som programmering. Disse perspektiver danner projektets teoretiske ramme og anvendes i det følgende til at analysere og begrunde designet af den udviklede AI-agent.

5 Teoretisk ramme

5.1 Den sokratiske metode i AI-baseret vejledning

Den sokratiske metode er en dialogisk undervisningsform, hvor læreren gennem spørgsmål guider den lærende til selv at formulere og undersøge sine antagelser frem for at modtage færdige svar (Delić & Bećirović, 2016). Metoden er forankret i en pædagogisk grundforståelse, hvor læring betragtes som en aktiv erkendelsesproces, og hvor spørgsmålet anvendes som didaktisk redskab til at afdække den lærendes forforståelse, udfordre antagelser og fremme refleksion. I uddannelseslitteraturen knyttes metoden ofte til scaffolding-traditionen (Wood, Bruner & Ross, 1976) og til Vygotskys begreb om zonen for nærmeste udvikling (Vygotsky, 1978), idet sokratisk spørgen kan forstås som en form for guidet støtte, der hjælper den lærende videre fra det aktuelle forståelsesniveau (Favero et al., 2024).

Med fremkomsten af LLM-baserede chatbots er det blevet muligt at implementere denne dialogiske tilgang i AI-baserede læringssystemer. Tidlige forsøg har anvendt metoden i forbindelse med matematikundervisning (Shridhar et al., 2022), kodefejlsøgning (Al-Hossami et al., 2024) og generel kritisk tænkning (Favero et al., 2024). Fælles for disse tilgange er, at AI-systemet i stedet for at levere direkte svar stiller spørgsmål, der udfordrer den studerendes forståelse og fremmer refleksion. Empiriske undersøgelser indikerer, at sokratisk designede AI-tutorer kan stimulere kritisk tænkning og selvstændig problemløsning sammenlignet med direkte-svar-orienterede chatbots (Favero et al., 2024), og at studerende oplever sokratisk AI-feedback som understøttende for akademisk integritet, netop fordi agenten ikke udleverer færdige løsninger (Nahar, 2025).

Begrebet anvendes imidlertid relativt løst i den voksende AI-litteratur. Hvor klassisk sokratisk dialog rummer en bred vifte af spørgsmålstyper, herunder afklaring, prøvelse af antagelser, undersøgelse af begrundelser, implikationer og alternative perspektiver (Ang et al., 2023), reduceres metoden i praksis ofte til en generel instruktion om, at AI-systemet skal "*stille spørgsmål frem for at give svar*". Denne forenkling indebærer en risiko for, at det, der i designdokumentationen omtales som sokratisk vejledning, i realiteten dækker over meget forskelligartede interaktionsformer. Favero et al. (2024) beskriver f.eks., hvordan deres sokratiske tutor under prompt-tuning gentagne gange brød med den dialogiske form og selv begyndte at levere svar, eller indledte med formuleringer som "*If I were a Socratic tutor, I would ask…*". Pointen er, at sokratisk dialog i en AI-kontekst ikke er en egenskab ved modellen, men et resultat af et vedvarende designarbejde, der i praksis ofte er ufuldstændigt realiseret.

Hertil kommer en mere grundlæggende empirisk indvending. Goldin, Pedroncini og Sigman (2017) viser i en eksperimentel undersøgelse, at sokratisk dialog kun er effektiv for en delmængde af lærende, idet metodens udbytte afhænger af den enkeltes evne til at abstrahere og generalisere. I forlængelse heraf fremhæver Favero et al. (2024), at sokratisk vejledning bør tilpasses individuelle forskelle i kognitiv bearbejdning, forforståelse og læringsstil. Dette peger på en spænding mellem metodens almene appel og dens betingede virkning: sokratisk dialog er ikke en universelt fungerende didaktisk strategi, men en strategi, hvis effekt afhænger af læringskontekst og lærendes forudsætninger.

Resultaterne fra eksisterende studier er da heller ikke entydige. Selvom flere undersøgelser dokumenterer øget engagement og refleksion, er det ikke i alle tilfælde dokumenteret, at dette omsættes til målbare forbedringer i læringsudbytte. Dette peger på en central didaktisk spænding mellem direkte vejledning og selvstændig problemløsning, og det understreger behovet for at undersøge, under hvilke betingelser sokratisk AI-vejledning fungerer i konkrete undervisningskontekster, frem for at antage en generel virkning. Denne pointe har direkte betydning for nærværende projekts designvalg og analytiske ramme og uddybes i den efterfølgende analyse.

5.2 Kognitiv belastningsteori og programmeringslæring

Kognitiv belastningsteori (Cognitive Load Theory) giver et relevant perspektiv på læring i programmering. Teorien bygger på antagelsen om, at menneskets arbejdshukommelse har begrænset kapacitet (Sweller, 1988).

Teorien skelner mellem:

- **Intrinsic load** – materialets iboende kompleksitet
- **Extraneous load** – belastning skabt af uhensigtsmæssigt design
- **Germane load** – kognitiv indsats der bidrager til læring

Programmering er et domæne med høj intrinsic load, da den lærende samtidig skal håndtere syntaks, logik og problemløsning (Berssanette & De Francisco, 2022).

AI-baserede værktøjer kan potentielt reducere extraneous load ved at tilbyde forklaringer og støtte til fejlfinding. Samtidig indebærer en høj grad af automatiseret hjælp en risiko for at reducere den kognitive indsats, der er nødvendig for opbygning af forståelse.

6 Design og udvikling af AI-agent

På baggrund af den teoretiske ramme er der udviklet en webbaseret AI-agent, som er designet til at understøtte studerendes læring i introduktionsfaget C#. Formålet med artefaktet er at operationalisere projektets centrale designprincipper, særligt sokratisk dialog og reduktion af unødvendig kognitiv belastning, i et konkret læringsværktøj, der kan indgå i programmeringsundervisningen på datamatikeruddannelsen. Den er tilgængelig på adressen dmo.f1-support.dk.

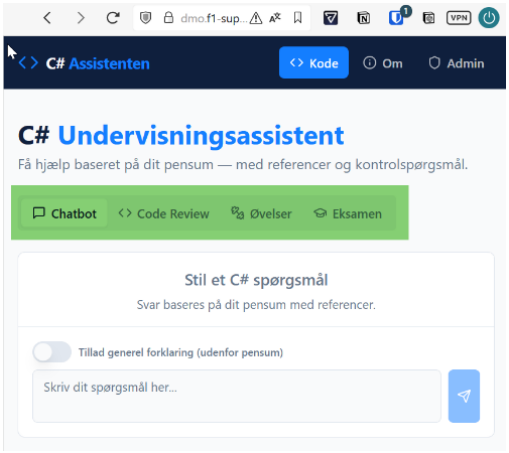
Agentens design tager udgangspunkt i projektets teoretiske ramme, hvor den sokratiske metode anvendes som didaktisk princip for dialogbaseret vejledning, samtidig med at systemet søger at understøtte læring i et kognitivt perspektiv.

I overensstemmelse med kognitiv belastningsteori er agenten designet til at strukturere problemløsningsprocesser uden at overtage den studerendes tænkning. Samtidig forstås udviklingen af systemet som en designbaseret intervention i undervisningspraksis, hvor artefaktet fungerer som en konkret realisering af en forestillet læringsvej (Misfeldt, 2010).

Agenten er struktureret i fire funktionelle moduler:

1. En dialogbaseret chatbot
2. Et kodereview modul
3. Et øvelsesmodul
4. Et eksamenstræningsmodul

Hvert modul er designet ud fra pædagogiske principper med særligt fokus på den sokratiske metode, der prioriterer vejledende spørgsmål frem for direkte svar, med henblik på at fremme aktiv og dybdeorienteret læring (Favero et al., 2024; Nahar, 2025).



Figur 3. dmo.f1-support.dk

Et centralt designprincip er, at agentens svar til enhver tid er forankret i det konkrete pensumindhold, som de studerende har adgang til. Agenten er instrueret i primært at basere sine svar på dette materiale gennem en retrieval-augmented generation-pipeline, hvor svar genereres på baggrund af relevante tekstpassager fra pensum.

6.1 Vidensdatabase og pensumafgrænsning

Agentens faktuelle vidensgrundlag er opbygget på basis af i alt 57 PDF-filer, der repræsenterer det undervisningsmateriale, som er udleveret til de studerende på 1. semester, suppleret med yderligere materiale inden for fagets pensum. Samtlige dokumenter er indlæst, opdelt og

indekseret i systemets vektordatabase, hvorfra de søges frem dynamisk ved hver brugerinteraktion.

6.1.1 Dokumentstruktur: PDF og tilhørende JSON-metadata

Hvert dokument i vidensdatabasen består af to sammenhængende filer: en **PDF-fil** med det faglige indhold samt en tilhørende **JSON-metadatafil**, der giver AI-agenten et struktureret og overskueligt overblik over dokumentets indhold og kontekst.

JSON-metadatafilerne indeholder blandt andet følgende felter:

- **id og curriculum:** Unik identifikator og tilknytning til det relevante semester-curriculum (f.eks. csharp-1sem).
- **source:** Titel, sprog, version og filnavn på den tilhørende PDF.
- **scope og level:** Angivelse af, hvilket semester dokumentet tilhører, samt hvilke begreber der er tilladt (allowedConcepts) og eksplicit ekskluderet (disallowedConcepts), f.eks. avancerede emner som LINQ, async/await og dependency injection, der ligger uden for 1. semesterpensums rammer.
- **topics:** En liste over faglige emner dækket af dokumentet, med tilhørende nøgleord og sideangivelser.
- **exerciseSeeds:** Forslag til øvelsesopgaver i tre sværhedsgrader (beginner, intermediate, advanced) med konkrete prompt-hints, der kan anvendes til at generere øvelser direkte fra dokumentets indhold.
- **ui.visibility:** Styrer, i hvilke moduler (chat, kodereview, øvelser, eksamen) dokumentet er aktivt og synligt for de studerende.

Som eksempel viser JSON-metadatafilen for dokumentet *C# Identifiers* (version 2026-02-17), at dokumentet dækker emnet Variabler med nøgleord som int, string, bool, var og scope, og at avancerede emner som LINQ og async/await eksplicit er markeret som ikke-tilladte for dette dokument. Tilsvarende indeholder filen tre øvelsesforslag, spændende fra deklaration af basistyper til demonstration af værdityper versus referencetyper.

Denne todelte dokumentstruktur, tjener et dobbelt formål. For det første øger JSON-metadataen præcisionen i den semantiske søgning, idet agenten ikke alene søger i rå tekst, men ligeledes kan anvende strukturerede metadata til at vurdere et dokuments relevans og faglige rækkevidde. For det andet muliggør strukturen en mere målrettet og kontrolleret brug af dokumentindholdet i øvelses- og eksamensmodulet, hvor øvelsesforslag kan genereres direkte på baggrund af de prædefinerede exerciseSeeds.

6.1.2 Pensumafgrænsning som didaktisk og teknisk princip

Et bevidst og centralt designvalg er, at AI-agenten er konfigureret til udelukkende at besvare spørgsmål med udgangspunkt i det indlæste pensumindhold. Denne afgrænsning er implementeret på to niveauer.

På det tekniske niveau kontrollerer RAG-pipelinen, om der eksisterer tilstrækkeligt relevante chunks i vektordatabasen til at besvare et givent spørgsmål. Hvis matchgrænsen ikke nås (cosine similarity < 0,3), og den studerende ikke har aktiveret tilvalget 'Tillad generel forklaring', returnerer systemet en eksplicit besked om, at spørgsmålet falder uden for pensums rammer.

På det pædagogiske niveau understøtter denne afgrænsning projektets overordnede forskningsformål: at undersøge, hvordan AI-assisteret vejledning påvirker studerendes forståelse af netop de begreber, der indgår i det konkrete undervisningsforløb. En agent, der besvarer vilkårlige programmeringsspørgsmål uden forankring i pensum, ville vanskeliggøre en valid vurdering af det fagspecifikke læringsudbytte og potentielt øge risikoen for, at de studerende anvender agenten som en generel kodegenerator frem for som et refleksivt læringsværktøj (Yan et al., 2024).

6.2 Systemarkitektur

Agenten er opbygget som en trelagsarkitektur bestående af et klientlag (frontend), et serverlag (backend via Supabase Edge Functions) og et AI-lag (to-model pipeline).

6.2.1 Frontend

Brugergrænsefladen er implementeret i React med TypeScript, bundlet via Vite og stilet med Tailwind CSS samt shadcn/ui komponentbiblioteket. Routing håndteres af React Router v6, mens datahentning og caching administreres via TanStack Query. AI-svar leveres som en kontinuerlig datastrøm til klienten ved hjælp af Server-Sent Events (SSE), hvilket giver en responsiv brugeroplevelse, da den studerende modtager svar løbende frem for at afvente et komplet svar. Markdown-formattering af AI-output rendereres via react-markdown, og der er integreret en Monaco Editor-komponent, der muliggør syntaksfremhævet koderedigering direkte i browseren.

6.2.2 Backend

Serverlogikken er implementeret som Supabase Edge Functions, der afvikles i et Deno-runtime-miljø. Databaselaget består af tre centrale tabeller:

Tabel	Formål
source_documents	Metadata om de 57 pensum-PDFer, herunder titel, emne, fagligt omfang og publiceringsstatus
document_chunks	Chunked PDF-indhold tilknyttet pgvector-embeddings til semantisk søgning
user_roles	Adminroller, adskilt fra brugertabellen med henblik på at forhindre privilege escalation

Semantisk søgning i pensumindhold realiseres via pgvector-udvidelsen til PostgreSQL. Embeddings genereres med OpenAI-modellen text-embedding-3-small, og dokumentsøgning foretages via en RPC-funktion (match_documents) baseret på cosine similarity med en matchgrænse på 0,3 og maksimalt fem returnerede chunks pr. forespørgsel. PDF-filer samt tilhørende JSON-metadatafiler lagres i Supabase Storage, hvortil kun administratorer har skriveadgang via row-level security (RLS)-politikker.

6.3 To-model AI-pipeline med retrieval-augmented generation

Et centralt designvalg i systemet er brugen af en to-model pipeline, der kombinerer retrieval-augmented generation (RAG) med sekventiel behandling via to sprogmodeller med komplementære styrker. Denne arkitektur adresserer udfordringen ved at sikre både faktuel præcision og pædagogisk kvalitet i AI-genererede svar (Yan et al., 2024).

6.3.1 Trin 1: RAG-søgning

Når den studerende stiller et spørgsmål, omdannes dette først til en vektorrepræsentation (embedding) via OpenAI text-embedding-3-small. Denne embedding anvendes til at foretage en semantisk søgning på tværs af de 57 indekserede pensumsdokumenter via pgvector, hvorved de mest relevante tekstpassager identificeres og returneres med kildehenvisninger. Herved forankres AI-agentens svar i det fagspecifikke curriculum frem for i sprogmodellens generelle træningsdata, hvilket reducerer risikoen for faktuelle fejl (hallucinations) og sikrer overensstemmelse med undervisningskonteksten.

6.3.2 Trin 2: Strukturering via OpenAI GPT-4o mini

I det andet trin sendes brugerens spørgsmål samt de identificerede RAG-chunks til OpenAI-modellen GPT-4o mini med instruktion om at returnere et struktureret JSON-objekt. Dette objekt valideres med AJV (Another JSON Validator) og indeholder blandt andet en responsplan,

nøglebegreber, et kontrolspørgsmål til den studerende samt en markering af, hvorvidt svaret baserer sig på pensumindhold eller på generel viden (feltet uses_outside_pensum). Denne struktureringstilgang adskiller det analytiske og indholdsmæssige lag fra det sproglige og pædagogiske lag og udnytter GPT-4o minis styrke i struktureret outputgenerering.

6.3.3 Trin 3: Pædagogisk formattering via Claude Sonnet

Det strukturerede JSON-objekt videresendes til Anthropic Claude Sonnet (claude-sonnet-4-20250514), der har til opgave at omdanne det til en sokratisk dialog på dansk Markdown. Claude-modellen er valgt til dette trin på baggrund af dens styrker inden for nuanceret, pædagogisk og kontekstbevidst sproglig formattering. Svaret streames til klienten via SSE, og ved afslutningen af svaret fremsendes en metadata-pakke indeholdende kildehenvisninger og et kontrolspørgsmål til den studerende.

Systemet inkluderer en fallback-mekanisme: Hvis Claude-modellen fejler, vises OpenAI JSON-outputtet direkte i brugergrænsefladen. Tilsvarende aktiveres en 'udenfor pensum'-besked, hvis der ikke identificeres relevante RAG-chunks, medmindre den studerende eksplicit aktiverer funktionen 'Tillad generel forklaring'.

6.4 Agentens fire moduler

Agenten er organiseret i fire funktionelle moduler, der tilsammen dækker centrale aspekter af den studerendes læringsproces i introduktionsfaget C#. For hvert dokument i vidensdatabasen er det i JSON-metadataen angivet, hvilke moduler dokumentet er aktivt i (feltet ui.visibility), hvilket muliggør en granuleret styring af, hvilke dele af pensum der er tilgængelige i de enkelte moduler.

6.4.1 Chatbot-modulet

Chatbot-modulet udgør agentens primære dialogflade og implementerer den sokratiske metode som pædagogisk princip. I overensstemmelse med forskning i AI-tutorlæring instrueres modellen til aldrig at levere fulde svar direkte, men i stedet at guide den studerende gennem ledende spørgsmål, delvise kodeeksempler og refleksionsopgaver (Favero et al., 2024). Hvert svar afsluttes med et kontrolspørgsmål, der inviterer den studerende til at demonstrere og uddybe sin forståelse. Denne tilgang er i overensstemmelse med fund fra Nahar (2025), der viser, at studerende oplever sokratisk AI-feedback som understøttende for akademisk integritet, netop fordi agenten ikke udleverer færdige løsninger.

Denne tilgang kan samtidig forstås som en form for stilladsering, hvor AI-agenten strukturerer problemløsningsprocessen uden at overtage den studerendes kognitive arbejde.

6.4.2 Kodereview-modulet

I kodereview-modulet kan den studerende indsende sin C#-kode til analyse. AI-agenten returnerer struktureret feedback med identifikation af fejl og forbedringsmuligheder, forankret i det relevante pensumindhold. Feedbacken er designet til at behandle fejl som læringsmuligheder frem for som negative bedømmelser, hvilket understøtter en formativ vurderingspraksis og er i tråd med forskning, der fremhæver betydningen af forklaringsorienteret AI-feedback frem for blot korrekte svar (Geyer et al., 2024).

6.4.3 Øvelsesmodulet

Øvelsesmodulet genererer C#-øvelsesopgaver afstemt til det aktuelle pensumindhold. Opgaverne tager udgangspunkt i exerciseSeeds fra JSON-metadatafilerne, der for hvert dokument specificerer opgaveforslag i tre sværhedsgrader (beginner, intermediate, advanced). Opgaverne leveres med graduerede hints og modelløsninger, der frigøres progressivt, så den studerende tilskyndes til selvstændig problemløsning, inden der søges hjælp. Denne tilgang afspejler den pædagogiske logik bag 'generation effect'-fænomenet, hvor aktiv selvgenerering af svar er associeret med bedre læringsudbytte end passiv aflæsning af løsninger (Fiorella & Mayer, 2016).

6.4.4 Eksamenstræningsmodulet

Eksamenstræningsmodulet simulerer en eksaminations- og vurderingssituation struktureret i fire faser: 1. præsentation af eksamensspørgsmål, 2. studerendes besvarelse, 3. AI-faciliteret sokratisk uddybning og 4. evaluering baseret på en rubrik. Denne model bidrager til at afdække gap mellem studerendes overfladiske og dybdeorienterede forståelse og er inspireret af forskningsresultater, der viser, at sokratisk-orienterede AI-agenter signifikant forbedrer kritisk tænkning sammenlignet med direkte-svar-modeller (Favero et al., 2024).

6.5 Sikkerhed og adgangskontrol

Systemets sikkerhedsarkitektur er designet med afsæt i princippet om mindste privilegium. Row-level security (RLS)-politikker er aktiveret på alle databasetabeller, og adminroller administreres i en separat tabel (user_roles), adskilt fra brugerprofiler, for at forhindre uautoriseret privilegieeskalering. Rollekontrol foretages via en has_role()-funktion med SECURITY DEFINER-attribut. API-nøgler til OpenAI og Anthropic lagres som krypterede Supabase Secrets og eksponeres aldrig i klientkode.

JWT-verificering er deaktiveret på Edge Functions for at muliggøre offentlig adgang til chatbotfunktionaliteten i undervisningssammenhæng. Adgangen begrænses i stedet gennem row-level security-politikker og en klar adskillelse mellem administrative og almindelige brugerroller.

6.6 Centrale designvalg og didaktisk begrundelse

Systemets overordnede designvalg kan sammenfattes i nedenstående tabel:

Designvalg	Pædagogisk og teknisk begrundelse
57 PDF-filer med tilhørende JSON-metadata	Dækker det fulde 1. semesterpensum; JSON-strukturen øger præcisionen af semantisk søgning og muliggør granuleret styring af øvelser og eksamen
Streng pensumafgrænsning	Forhindrer, at agenten fungerer som generel kodegenerator; understøtter valid måling af fagspecifikt læringsudbytte (Yan et al., 2024)
To-model pipeline (OpenAI → Claude)	OpenAI GPT-4o mini er overlegen til struktureret JSON-output; Claude Sonnet er overlegen til nuanceret, pædagogisk dansk prosa
RAG med pgvector	Forankrer svar i pensumindhold og reducerer risikoen for faktuelle fejl (hallucinations)
Sokratisk metode	Fremmer aktiv og dybdeorienteret læring; modvirker copy/paste-adfærd og over-reliance på AI (Favero et al., 2024; Nahar, 2025)
Streaming via SSE	Forbedrer brugeroplevelsen ved at reducere oplevet ventetid
Dansksproglig grænseflade	Målgruppen er dansksprogede datamatikerstuderende; sprognær formidling reducerer kognitiv belastning
AJV JSON-validering	Sikrer robust og forudsigelige AI-output uanset modelvariationer

På baggrund af de beskrevne designvalg kan systemets didaktiske intentioner sammenfattes i en række designprincipper. Disse principper repræsenterer en abstraktion af de konkrete tekniske implementeringer og tydeliggør sammenhængen mellem projektets teoretiske ramme og det udviklede artefakt.

Designprincip	Implementering i AI-agenten	Didaktisk begrundelse
Sokratisk vejledning	Chatbot stiller spørgsmål frem for at levere fulde løsninger	Fremmer refleksion og kritisk tænkning hos den studerende
Curriculum alignment	RAG-søgning i pensum-PDF-filer	Sikrer at AI-svar er forankret i undervisningens faglige indhold

Stilladsering	Graduerede hints i øvelsesmodulet	Understøtter progression i problemløsning
Formativ feedback	Kodereview-modul med forklarende feedback	Fejl behandles som læringsmuligheder
Progressiv støtte	Modelløsninger frigives gradvist	Reducerer risikoen for copy/paste-adfærd

6.7 Agentens organisatoriske integration som åbent designvalg

Et bevidst valg i designet har været at holde agentens organisatoriske integration åben. Agenten er udviklet som et selvstændigt artefakt, der i princippet kan anvendes af den studerende både som selvstudieværktøj uden for undervisningstid, som supplement til instruktørstøtte i øvelsestimer, og som ressource i forbindelse med specifikke moduler eller opgaver. Designet specificerer ikke, hvordan denne integration skal foregå i praksis, og indeholder ikke funktioner til, at undervisere kan se eller monitorere studerendes interaktion med agenten.

Denne åbenhed er ikke alene en pragmatisk afgrænsning, men afspejler en analytisk position: hvordan en AI-agent integreres i en konkret undervisningsorganisation, er et selvstændigt didaktisk spørgsmål, der ikke kan reduceres til agentens tekniske egenskaber. Underviserens rolle, forholdet mellem agentens og underviserens vejledning, og studerendes parallelle brug af andre AI-værktøjer udgør organisatoriske forhold, hvis betydning først kan vurderes empirisk i den konkrete anvendelseskontekst. Disse forhold uddybes i diskussionen på baggrund af de empiriske fund.

6.8 Design sammenfatning

Den udviklede AI-agent repræsenterer et pædagogisk funderet IKT-artefakt, der integrerer aktuelle metoder inden for generativ AI, semantisk søgning og didaktisk design. Vidensdatabasen med 57 PDF-filer og tilhørende JSON-metadatafiler udgør et curriculumspecifikt vidensgrundlag, der sikrer, at agentens svar altid er forankret i det konkrete undervisningsmateriale. Den strenge pensumafgrænsning er ikke alene et teknisk valg, men ligeledes en didaktisk prioritering: den hindrer, at agenten reduceres til et generelt kodegeneringsværktøj, og understøtter i stedet et læringsdesign, hvor de studerende aktivt arbejder med det faglige indhold.

Den to-model pipeline kombinerer faktuel præcision i vidensformidlingen med høj kvalitet i den pædagogiske sproglige interaktion, mens den sokratiske tilgang er valgt på baggrund af empirisk forskning, der dokumenterer dens effektivitet til at fremme kritisk tænkning og dybdelæring sammenlignet med direkte-svar-orienterede AI-systemer (Favero et al., 2024). Agentens fire

moduler adresserer komplementære aspekter af den studerendes læringsproces og er designet til at indgå som en integreret del af undervisningen på 1. semester datamatikeruddannelsen.

I projektets forskningsdesign fungerer AI-agenten således både som et læringsværktøj og som et empirisk undersøgelsesobjekt, hvor interaktioner mellem studerende og systemet kan analyseres med henblik på at forstå, hvordan sokratisk designede AI-systemer påvirker læringsprocesser i introducerende programmeringsundervisning.

For at undersøge, hvordan dette design fungerer i praksis, beskrives i det følgende den metodiske tilgang til afprøvning og analyse.

7 Metode

Dette projekt anvender Design-Based Research (DBR) som metodisk tilgang. DBR er en forskningsstrategi, der kombinerer udvikling af læringsteknologiske artefakter med systematisk empirisk undersøgelse af deres anvendelse i autentiske undervisningskontekster (Design-Based Research Collective, 2003; Barab & Squire, 2004).

I sin fulde form er DBR karakteriseret ved en iterativ logik, hvor design, afprøvning og analyse gensidigt informerer hinanden gennem flere på hinanden følgende cyklusser. Hver iteration tester et didaktisk artefakt empirisk, identificerer styrker og begrænsninger, og fører til et kvalificeret redesign, der afprøves på ny. Formålet med denne iterative bevægelse er ikke alene at evaluere et konkret system, men at generere designprincipper, der gradvist konsolideres gennem gentagne empiriske afprøvninger (Anderson & Shattuck, 2012).

7.1 Eksplorativ DBR-inspireret første iteration

Nærværende studie kan forstås som en eksplorativ DBR-inspireret første iteration. Inden for projektets tidsmæssige og praktiske rammer er der gennemført én designcyklus: udvikling af AI-agenten, afprøvning blandt studerende på 1. semester af datamatikeruddannelsen, og analyse af de empiriske data. Projektet rummer dermed kerneelementerne i DBR, et didaktisk funderet artefakt udviklet og afprøvet i en autentisk undervisningskontekst, men ikke den iterative bevægelse, hvor erfaringerne omsættes til et nyt design og afprøves på ny.

Denne afgrænsning har konsekvenser for fundenes epistemologiske status. De designprincipper, der udledes af analysen, herunder sekventering af produktion og dialog, differentieret scaffolding og verifikationsbyrden hos nybegyndere, er empirisk genererede, men ikke valideret gennem efterfølgende afprøvning. De fremstår derfor som hypoteser om designforbedringer, snarere end som principper, der er konsolideret gennem flere DBR-cyklusser. En naturlig forlængelse af

projektet ville være en anden iteration, hvor disse principper omsættes til konkrete redesign-elementer, eksempelvis en revideret prompt-struktur i chat-modulet, og afprøves empirisk på ny.

Denne afgrænsning er forenelig med Barab og Squires (2004) begreb om humble theory, hvor ambitionen ikke er at levere universelle påstande, men en lokalt forankret forståelse af, hvordan et bestemt design fungerer i en specifik læringsøkologi. Et eksplorativt første forløb opfylder denne ambition, så længe rækkevidden af fundene formuleres tilsvarende beskedent.

7.2 Validitet og forskerposition

Et yderligere metodisk forhold, der kræver eksplicit refleksion, er min dobbeltrolle. Jeg er på samme tid designer af AI-agenten, underviser for de studerende, der har deltaget i evalueringen, og forsker, der analyserer det empiriske materiale. Denne dobbeltrolle er karakteristisk for DBR, hvor forskeren typisk er praktisk involveret i den intervention, der undersøges (Anderson & Shattuck, 2012), men den indebærer samtidig en risiko for, at fund afspejler forskerens forforståelse snarere end deltagernes faktiske erfaringer.

For at imødegå denne risiko er der i undersøgelsen anvendt flere konkrete tiltag:

For det første blev interviewene struktureret narrativt, så de studerende indledningsvis blev opfordret til at beskrive konkrete situationer med egne ord, inden samtalen bevæger sig mod mere fokuserede spørgsmål (Kvale & Brinkmann, 2015). Den sokratiske tilgang blev ikke introduceret som begreb i interviewet, og deltagerne opfordres ikke til at vurdere agenten ud fra projektets analytiske kategorier. Dette mindsker risikoen for, at interviewene reproducerer mine egne designintentioner.

For det andet er spørgeskema og interviews behandlet som komplementære datakilder, hvor interviewene anvendes til at nuancere og potentielt udfordre de mønstre, spørgeskemaet identificerer. Et konkret eksempel er sekventeringspointen: spørgeskemaet alene ville have ført til den fortolkning, at chat-modulet var det mindst velfungerende modul. Interviewene gjorde det imidlertid muligt at omfortolke dette mønster og identificere en mere præcis didaktisk variabel. Denne form for datatriangulering styrker analysens validitet ved at lade flere typer empiri belyse samme fænomen.

For det tredje er interviewanalysen organiseret omkring temaer, der er udledt af de studerendes egne fortællinger, snarere end omkring projektets a priori kategorier. Flere af de identificerede temaer, særligt verifikationsbyrden hos nybegyndere og betydningen af det parallelle AI-økosystem, var ikke forudset i designet og er fremkommet gennem analysen.

På trods af disse tiltag er det vigtigt at anerkende, at undersøgelsen ikke er uafhængig af min position. De studerende har deltaget i evalueringen som led i et undervisningsforløb hos mig, hvilket potentielt kan have påvirket både deres villighed til at fremføre kritik og deres tolkning af agentens hensigt.

7.3 Reproducerbarhed og dataadgang

I overensstemmelse med akademiske krav om transparens og reproducerbarhed er undersøgelsens datagrundlag dokumenteret som følger:

Spørgeskemaet er udarbejdet i Microsoft Forms og består af seks sektioner, der dækker baggrundsinformation, modulvise vurderinger og oplevet læringsudbytte. Det fulde spørgeskema, herunder samtlige Likert-spørgsmål og åbne refleksionsspørgsmål, er vedlagt som Bilag 1.

Interviewguiden er præsenteret i sin helhed i hovedteksten under afsnittet Interviewramme. Dette valg er truffet fordi guidens semistrukturerede karakter og dens specifikke formuleringer har analytisk betydning for, hvordan interviewdataene skal forstås.

Den udviklede AI-agent, herunder dens prompt-struktur, RAG-pipeline og modulopbygning, er beskrevet teknisk i afsnittet Design og udvikling af AI-agent. Selve agenten er offentligt tilgængelig på dmo.f1-support.dk, hvilket muliggør efterprøvning og videre undersøgelse.

De anonymiserede interviewtransskriptioner er ikke offentliggjort af hensyn til deltagernes anonymitet, men kan stilles til rådighed for vejleder og censor på anmodning.

I dette studie blev undersøgelsen gennemført som et eksplorativt første design- og evalueringsforløb, hvor AI-agenten blev afprøvet blandt studerende på 1. semester på datamatikeruddannelsen. Data blev indsamlet gennem spørgeskemaer og interviews. Formålet var at undersøge, hvordan agentens design påvirker studerendes interaktion, refleksion og oplevede læringsudbytte, samt at generere designprincipper, der kan informere et eventuelt senere iterationsforløb.

I de følgende afsnit beskrives først datagrundlaget og deltagerne i undersøgelsen, hvorefter resultaterne af evalueringen præsenteres og analyseres.

7.4 Datagrundlag

Evalueringen af AI-agenten blev gennemført som et eksplorativt brugerstudie blandt 12 studerende på 1. semester på datamatikeruddannelsen. De studerende interagerede med agenten gennem en række strukturerede opgaver, hvorefter de besvarede et spørgeskema i Microsoft Forms.

Spørgeskemaet bestod af seks sektioner:

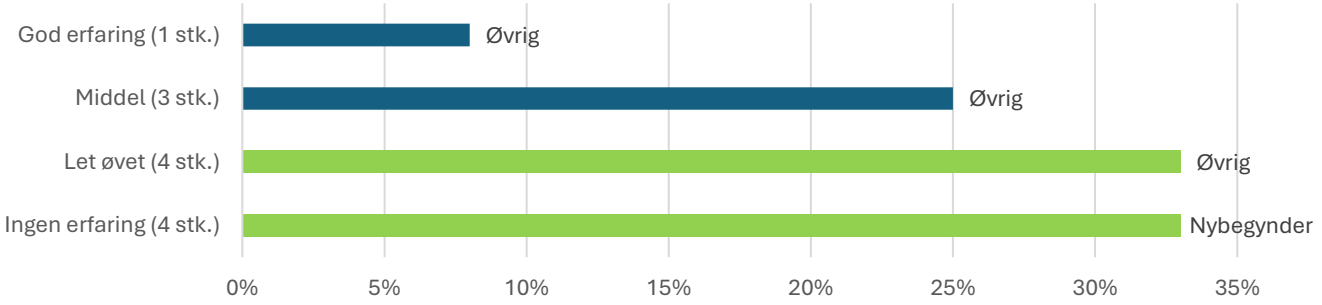
1. Baggrundsinformation om programmeringserfaring
2. Test af chatbot-funktionalitet
3. Test af code review-funktionalitet
4. Test af opgavegenerering
5. Overordnet vurdering
6. Oplevet læringsudbytte

De fleste spørgsmål blev besvaret på en Likert-skala fra 1 til 5 (1 = helt uenig, 5 = helt enig). Spørgeskemaet indeholdt desuden åbne refleksionsspørgsmål om oplevelsen af agentens styrker og svagheder (Se bilag 1 – Spørgeskema)

Deltagerfordeling

Gruppen består af 12 studerende og er præget af en overvægt af nybegyndere. I alt 67 % af deltagerne angiver et erfaringsniveau på 1–2 på en skala fra 1 til 5 (se *Figur 4*).

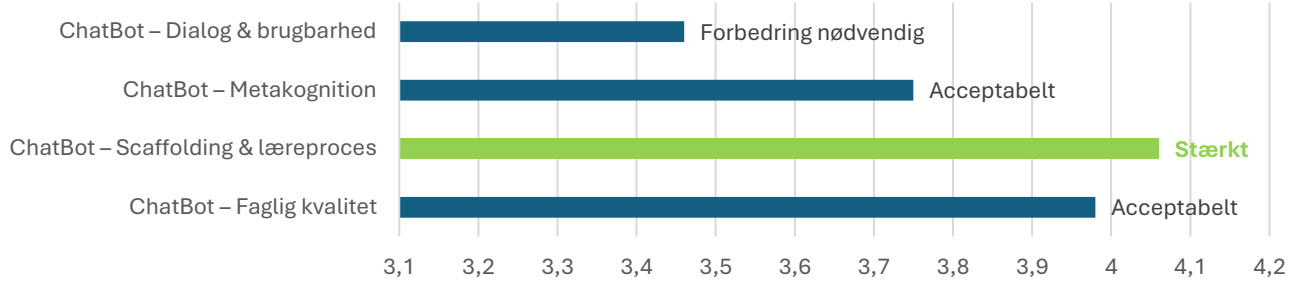
Dette svarer til målgruppen for introduktionsprogrammering på datamatikeruddannelsen.



Figur 4. Fordelingen af deltagernes erfaring med programmering på en skala fra 1 til 5

Resultater pr. funktion

Chatbot-funktionalitet

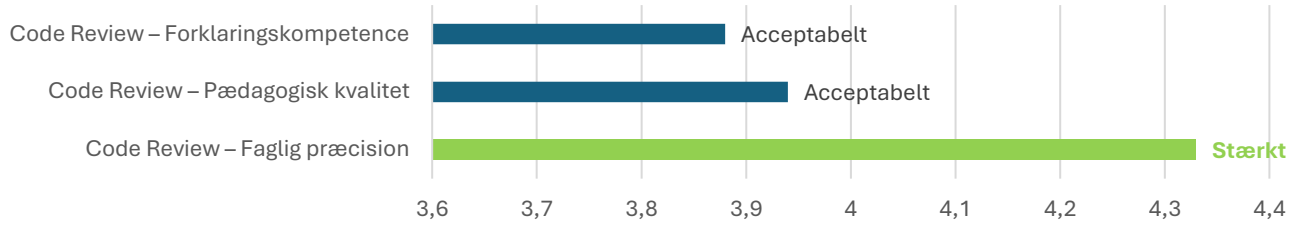


Figur 5. Evaluering af chatbot-funktionalitet fordelt på faglig kvalitet, scaffolding, metakognition og dialogisk brugbarhed.

Evalueringen viser, at chatbotten scorer højt på faglig korrekthed og evnen til at guide studerende mod en løsning frem for blot at levere et færdigt svar. Studerende vurderer i særlig grad agentens evne til at stille spørgsmål, der fremmer refleksion over problemløsningen.

Dette indikerer en vejledningsstil, der i høj grad understøtter refleksiv problemløsning og kan tolkes som en form for sokratisk dialogbaseret vejledning.

Code Review

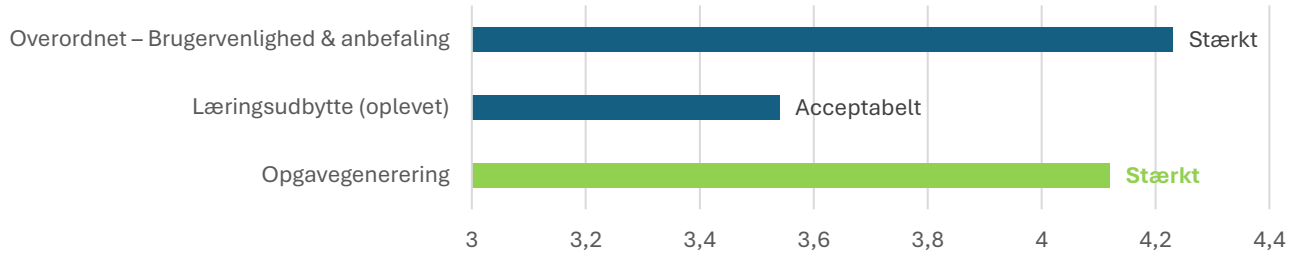


Figur 6. Studerendes vurdering af Code Review-funktionen fordelt på faglig præcision, pædagogisk kvalitet og forklaringskompetence

Code Review-funktionen opnår den højeste samlede vurdering i evalueringen (gennemsnit 4,06). Studerende fremhæver særligt funktionens evne til at identificere konkrete fejl i kode samt give specifikke forslag til forbedringer.

Dette tyder på, at Code Review-funktionen fungerer som et effektivt læringsredskab til at støtte forståelse af kodestil, fejltyper og forbedringsmuligheder.

Opgavegenerering

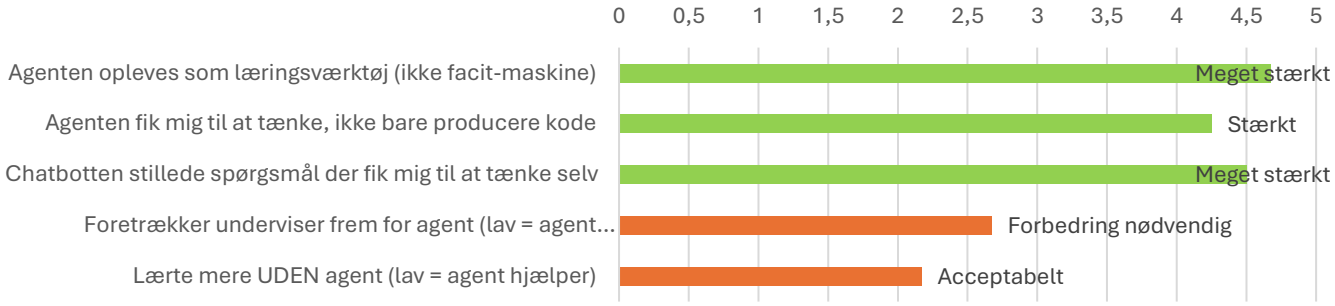


Figur 7. Studerendes vurdering af AI-agentens evne til at generere programmeringsopgaver med passende sværhedsgrad og faglig relevans for introduktionsprogrammering.

Funktionen til generering af programmeringsopgaver vurderes ligeledes positivt (gennemsnit 4,12). Studerende vurderer både opgavernes faglige relevans og sværhedsgrad som passende for introduktionsprogrammering.

Resultatet indikerer et potentiale for anvendelse af AI-genererede øvelser som støtte til selvstændig træning.

Læringsudbytte

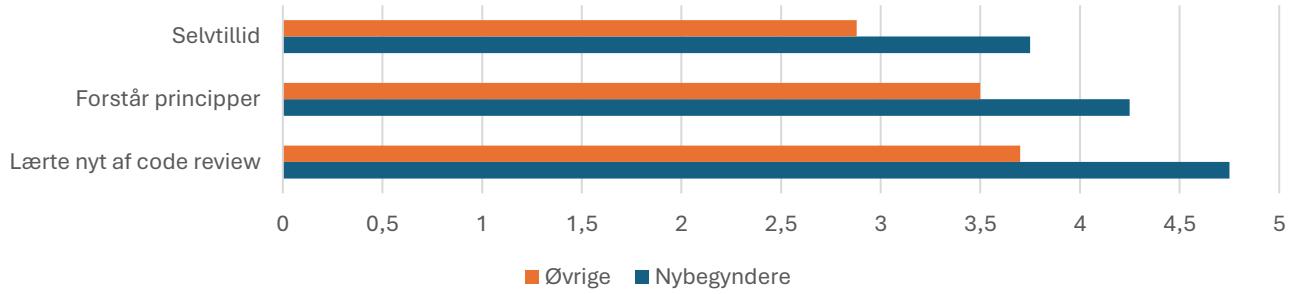


Figur 8. Indikatorer for oplevet læringsudbytte og brug af AI-agenten som læringsværktøj.

Resultaterne viser en interessant forskel mellem funktionel kvalitet og oplevet læring. Mens agenten generelt vurderes positivt som et læringsværktøj, rapporterer studerende kun en moderat forbedring i deres samlede forståelse af programmering efter én enkelt session.

Dette kan indikere, at agentens læringseffekt i højere grad viser sig gennem støtte til problemløsning i konkrete situationer end gennem umiddelbar oplevet kompetenceudvikling.

Sammenligning: Nybegyndere vs. Øvrige



Figur 9. Sammenligning af udvalgte indikatorer for læringsudbytte opdelt efter deltagernes erfaring med programmering

Når resultaterne opdeles efter erfaring, fremgår det, at nybegyndere generelt rapporterer større læringsudbytte end studerende med mere programmeringserfaring.

Dette tyder på, at agenten i særlig grad understøtter studerende med begrænset forforståelse, hvilket er i tråd med projektets mål om at støtte begyndere i introduktionsprogrammering.

De kvalitative svar

De åbne svar viser tre gennemgående temaer.

Guidende læring

Flere studerende fremhæver agentens evne til at guide frem for at levere færdige svar:

"Jeg kan rigtig godt lide at den ikke bare giver svaret."

Refleksion

Studerende beskriver også, at agenten understøtter refleksion:

"Den stiller spørgsmål man selv kan tænke over."

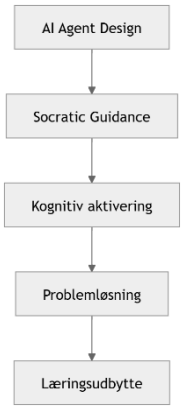
UX-problemer

Der identificeres samtidig nogle tekniske udfordringer:

- Manglende kodeformatering
- Lange svar

Disse observationer fremstår delvis som UX-problemer, men antyder også et dybere mønster, der uddybes i interviewanalysen.

Resultaterne peger samlet på en mekanisme, hvor agentens design påvirker den studerendes læringsproces gennem en række mellemled. Denne relation kan illustreres i denne konceptuelle model.



Figur 10. Foreløbig hypotese konceptuel model

Modellen illustrerer en foreløbig hypotese om, hvordan agentens sokratiske vejledningsstrategi kan føre til kognitiv aktivering hos den studerende og dermed understøtte problemløsning og udvikling af programmeringsforståelse. Modellen er udledt af spørgeskemadataene og bygger på en lineær antagelse om sammenhængen mellem designprincip, kognitiv aktivering og læringsudbytte.

Som det fremgår af den efterfølgende interviewanalyse, peger empirien imidlertid på, at denne sammenhæng er kontingent på en række betingelser, der ikke er repræsenteret i modellen. Modellen tjener derfor som et analytisk udgangspunkt, der efterprøves og kvalificeres i interviewanalysen.

7.5 Syntese af spørgeskemadata

Resultaterne fra evalueringen peger på et mønster, hvor AI-agentens styrker primært ligger i dens evne til at støtte studerendes problemløsningsproces, snarere end i at skabe en umiddelbar oplevelse af forbedret programmeringskompetence.

Evalueringen viser blandt andet, at agenten scorer højt på indikatorer relateret til guidning og refleksion, f.eks. i vurderingen af, at agenten stiller spørgsmål der får studerende til at tænke selv, og at den opleves som et læringsværktøj frem for en "facit-maskine". Dette kan forstås i lyset af sociokulturelle læringsteorier, hvor læring anskues som en proces, der udvikles gennem støtte og dialog.

I et Vygotskij-inspireret perspektiv kan agenten tolkes som en form for digital støtte i den studerendes zone for nærmeste udvikling. Ved at stille spørgsmål og give gradvise forklaringer kan agenten fungere som en midlertidig støtte, der hjælper den studerende med at bevæge sig fra det, vedkommende allerede kan, til det næste niveau af forståelse.

Denne støtte kan også forstås gennem begrebet scaffolding, hvor læringsprocessen understøttes gennem midlertidige strukturer, der gradvist kan fjernes, efterhånden som den studerende opnår større selvstændighed. Resultaterne viser blandt andet, at agenten i høj grad guider studerende mod løsningen frem for at levere den direkte. Det indikerer, at agenten i nogen grad fungerer som en sådan støtte.

Et centralt fund er dog, at denne processtøtte ikke umiddelbart omsættes til en tydelig oplevelse af øget kompetence efter en enkelt session. Flere indikatorer relateret til oplevet læringsudbytte scorer lavere end indikatorer for funktionalitet og processtøtte (se Figur 10).

Dette kan tolkes på flere måder.

For det første kan det afspejle, at læring i programmering typisk er gradvis og kumulativ, hvor forståelsen udvikles over længere tid gennem gentagen problemløsning. En enkelt interaktion med en AI-agent vil derfor ikke nødvendigvis skabe en tydelig oplevelse af forbedret kompetence.

For det andet kan forskellen mellem processtøtte og oplevet læring ses som en indikation af, at agenten primært fungerer som en kognitiv støtte under problemløsning, snarere end som en direkte kilde til konceptuel læring.

Et andet vigtigt resultat er forskellen mellem studerende med forskellig erfaring. Analysen viser, at nybegyndere generelt rapporterer større læringsudbytte end mere erfarne studerende. Dette stemmer overens med forskning i læringsteknologier, hvor støtteværktøjer ofte har størst effekt

for studerende med begrænset forforståelse. For begyndere kan selv små forklaringer, spørgsmål og kodegennemgange have en relativt stor betydning for forståelsen.

Samlet peger resultaterne derfor på, at AI-agenten i højere grad fungerer som en digital tutor, der understøtter refleksion og problemløsning, end som et værktøj der direkte producerer læring i sig selv. Resultaterne tyder derfor på, at læringspotentialet i højere grad ligger i den dialogiske proces, hvor den studerende aktivt engageres i forståelsen af kode og problemer, end i agenten som direkte vidensformidler.

7.6 Interview

Formålet med interviewene var at undersøge, hvordan studerende uden forudgående programmeringserfaring oplever interaktionen med AI-agenten i konkrete situationer, og dermed besvare anden del af projektets problemformulering. Hvor spørgeskemaet kortlægger studerendes vurderinger af agentens forskellige funktioner, giver det semistrukturerede interview adgang til de processer, som ligger bag disse vurderinger: hvordan den studerende rent faktisk arbejder med agenten, hvilke overvejelser der opstår undervejs, og hvordan oplevelsen ændrer sig gennem en konkret problemløsningssituation. Interviewene supplerer dermed spørgeskemaet ved at uddybe og kvalificere de identificerede mønstre gennem studerendes egne beskrivelser af konkrete oplevelser med agenten.

Det semistrukturerede interview er valgt som metode, fordi det giver mulighed for at følge den enkeltes fortælling, mens det samtidig sikrer, at centrale temaer berøres på tværs af interviews (Kvale & Brinkmann, 2015). Hvor spørgeskemaet måler vurderinger på en skala, kan interviewet indfange den processuelle dimension af studerendes læringserfaringer, herunder hvordan agentens vejledning opleves i konkrete arbejdssituationer, og hvilke betingelser der gør dialogen produktiv eller udfordrende. Dette er centralt for at kunne identificere de designprincipper, som projektets problemformulering efterspørger.

Interviewene blev struktureret efter en narrativ tilgang, hvor den studerende indledningsvis blev opfordret til at beskrive en konkret situation med agenten med egne ord (Kvale & Brinkmann, 2015).

Denne åbning blev prioriteret frem for direkte evalueringsspørgsmål, idet situationsbaserede fortællinger giver adgang til erfaringer, som den studerende måske ikke selv ville fremhæve som betydningsfulde, men som alligevel er analytisk relevante. Opfølgende spørgsmål blev anvendt til at uddybe specifikke aspekter, herunder oplevelsen af agentens vejledningsstil, øjeblikke af forståelse eller manglende forståelse og vurderingen af, hvornår agentens tilgang ikke svarede til den studerendes behov.

Den sokratiske tilgang blev ikke introduceret som begreb i interviewet. Studerende blev i stedet opfordret til at beskrive, hvad de lagde mærke til ved agentens måde at svare på, for at undgå at lede svarene i retning af projektets egne analytiske kategorier.

Interviewene blev gennemført online via Microsoft Teams, optaget som lydfiler og efterfølgende transskriberet.

7.6.1 Udvælgelse af interviewdeltagere

Af de 12 studerende, der deltog i spørgeskemaundersøgelsen, blev tre udvalgt til opfølgende semistrukturerede interviews. Udvælgelsen er foretaget som en målrettet stikprøve (purposive sampling), hvor udvælgelseskriterierne er begrundet i projektets problemformulering og det videre analytiske formål med interviewene (Kvale & Brinkmann, 2015).

Det primære udvælgelseskriterium er deltagernes selvrapporterede programmeringserfaring. AI-agenten er designet med henblik på at understøtte studerende uden eller med begrænset forudgående programmeringserfaring, hvilket afspejler målgruppen for introduktionsprogrammering på datamatikeruddannelsen. Det er derfor disse studerendes oplevelser, der har størst analytisk relevans for projektets problemformulering om, hvordan en sokratisk AI-agent kan understøtte refleksion og problemløsning hos studerende uden programmeringserfaring.

Spørgeskemaets data viser, at 8 af de 12 deltagere angiver et erfaringsniveau på 1-2 på en skala fra 1 til 5. Inden for denne gruppe er der udvalgt to studerende, Rasmus og Søren, der repræsenterer forskellige positioner i nybegyndersegmentet. Rasmus har ingen forudgående programmeringserfaring og angiver, at hans eneste tidligere berøring med programmering var blokprogrammering i tre lektioner ti år tidligere. Søren har derimod brugt omkring fem måneder på selvstudier i Unity og C# umiddelbart før studiestart og har dermed en mere udviklet, men stadig uformel, baggrund. Som tredje interviewperson er Meklas udvalgt på baggrund af sin bredere programmeringserfaring fra Java, C++ og C# gennem private projekter. Meklas adskiller sig således fra de to øvrige ved at have en eksisterende faglig referenceramme.

Denne udvælgelsesstrategi er valgt af to grunde. For det første sikrer den, at hovedparten af interviewdataene afspejler agentens primære målgruppe. For det andet giver inklusionen af Meklas mulighed for at undersøge, om og hvordan oplevelsen af agenten varierer med forforståelse, hvilket er relevant for projektets udvikling af betingede designprincipper. Forskellen mellem deltagerne er dermed ikke en svaghed, men en analytisk ressource, der gør det muligt at identificere mønstre, der varierer på tværs af erfaringsniveauer.

Antallet af interviews er begrænset til tre, hvilket reflekterer projektets eksplorative karakter og rammer for et første gangsdesign- og evalueringsforløb. Formålet er ikke at opnå metning i en klassisk forstand, men at supplere spørgeskemadataene med tilstrækkelig kvalitativ dybde til at uddybe og nuancere de identificerede mønstre. Antallet er derfor afstemt efter projektets analytiske ambition om at identificere designprincipper, snarere end at producere generaliserbare bare fund.

Alle tre deltagere har givet informeret samtykke til at deltage i interviewet og til, at samtalen kunne optages og transskriberes med henblik på efterfølgende analyse. Deltagerne er anonymiseret i rapporten ved brug af deres fornavne efter aftale.

7.6.2 Interviewramme

Interviewguiden er opbygget i seks tematiske blokke, der bevæger sig fra det konkrete og oplevelsesnære mod det reflekterende: opstart, en konkret situation, oplevelsen af agentens svar, forståelse og læring, når det ikke virkede, samt afslutning. Hver blok indledes med et åbent narrativt spørgsmål, der inviterer til fortælling, og følges op med uddybende spørgsmål, der kan tages i brug efter behov. Denne progression understøtter den narrative tilgang, der er beskrevet ovenfor, ved at lade deltagernes egne fortællinger styre samtalen, før der spørges til mere overordnede vurderinger.

For hver blok er der formuleret et eksplicit metodisk formål, gengivet i kursiv nedenfor. Formålsbeskrivelserne har til hensigt dels at gøre interviewerens orientering transparent for læseren, dels at sikre konsistens på tværs af de tre interviews. Den indledende formulering nedenfor blev læst op for hver deltager ved interviewets start for at etablere en uformel ramme og signalere, at fokus er på oplevelse frem for korrekt brug.

Opstart

Jeg leder efter din oplevelse, der er ingen rigtige svar. Jeg er interesseret i situationer, følelser og tanker undervejs, ikke om du "brugte det rigtigt".

- Kan du kort fortælle om din baggrund, har du programmeret før?
- Hvad forventede du, inden du gik i gang?

Formål: etablér kontekst og åbn samtalen uformelt. Det første spørgsmål er nødvendigt for at kunne fortolke resten.

En konkret situation

- Kan du fortælle om en konkret situation, hvor du brugte agenten. Hvad arbejdede du på, og hvad skete der?

- *Lad dem fortælle frit. Følg op med:*
  - Hvad tænkte du undervejs?
  - Hvad gjorde du, efter agenten svarede?
  - Var der noget i det øjeblik, der overraskede dig?

Formål: få en rig, narrativ beskrivelse frem for vurderinger. Narrativ tilgang er metodisk velfunderet i kvalitativ interviewforskning (Kvale & Brinkmann, 2015).

Oplevelsen af agentens måde at svare på

- Hvad lagde du mærke til ved den måde, agenten svarede dig på?
- Var der tidspunkter, hvor du reagerede på det - positivt eller negativt?
- Kan du beskrive et sådant øjeblik nærmere?

Formål: afdække reaktionen på den sokratiske tilgang uden at introducere begrebet "sokratisk" lad dem beskrive det med egne ord.

Følgespørgsmål: Hvad tænkte du, da agenten svarede dig med et spørgsmål?

Forståelse og læring

- Hvis du tænker på din forståelse af det, du arbejdede med — oplevede du nogen forskel undervejs eller bagefter?
- Kan du prøve at beskrive, hvad der skete i den proces?

Formål: åbner for refleksion over læringsudbytte uden at forudsætte, at der var et.
Evt. følgespørgsmål:

- Hvis en medstuderende spurgte dig om det samme problem nu, hvad ville du sige til dem?

Når det ikke virkede

- Var der situationer, hvor agenten ikke hjalp eller hvor du følte dig mere frustreret end hjulpet?
  - Hvad skete der i de situationer?
  - Hvad havde du brug for i stedet?

Formål: kritisk perspektiv og afklaring af begrænsninger.

Afslutning og frit rum

- Er der noget ved din oplevelse med agenten, som vi ikke har talt om, som du synes er vigtigt?

7.6.3 Tematisk analyse

Interviewmaterialet er analyseret gennem en tematisk analyse, en af de mest etablerede tilgange til kvalitativ dataanalyse, der tilbyder en fleksibel ramme for identifikation, analyse og fortolkning af mønstre i datamaterialet (Braun & Clarke, 2006). Tilgangen er valgt, fordi den muliggør en induktiv tilgang til de studerendes egne fortællinger, samtidig med at analysen kan informeres af projektets teoretiske ramme, hvor det er relevant.

Analysen følger de seks faser beskrevet af Braun og Clarke (2006). I første fase blev interviewene transskriberet og gennemlæst flere gange med henblik på at opnå fortrolighed med materialet samt identificere foreløbige iagttagelser. I anden fase blev datamaterialet kodet systematisk, hvor analytisk relevante passager blev fremhævet og forsynet med beskrivende koder. Kodningen var primært induktiv, men ikke teorifri, idet mønstre identificeret i spørgeskemadataene fungerede som sensitiverende perspektiver.

I tredje og fjerde fase blev koderne grupperet i potentielle temaer, som efterfølgende blev revideret iterativt i relation til både de kodede passager og det samlede datamateriale. Formålet var at sikre temaernes interne sammenhæng og analytiske gyldighed. I femte fase blev de endelige temaer defineret og navngivet med henblik på at indfange deres centrale analytiske bidrag. I sjette fase blev temaerne integreret i den skriftlige fremstilling og forankret i konkrete interviewuddrag.

Analysen kombinerer en induktiv og en teoretisk informeret tilgang. Flere temaer, herunder verifikationsbyrde og agenten i et AI-økosystem, fremkom induktivt gennem de studerendes fortællinger og var ikke forudset i designet. Andre temaer, særligt sekventering af produktion og dialog, opstod i samspil mellem empiri og teori, hvor begreber fra kognitiv belastningsteori og scaffolding-traditionen blev anvendt til at fortolke identificerede mønstre.

Analysens resultat er en række tværgående temaer, der danner grundlag for udviklingen af projektets designprincipper. I en designbaseret forskningsramme fungerer disse temaer således ikke alene som analytiske kategorier, men som empirisk funderede bidrag til videre design.

7.6.4 Interviewanalyse

De fire tværgående temaer, der præsenteres i det følgende, er udledt gennem den tematiske analyse, der er beskrevet i metodeafsnittet (afsnit 7.6.3), suppleret af de mønstre, der allerede

var identificeret i spørgeskemadataene. Temaerne er formuleret med henblik på at trække den analytiske forskel frem, som interviewdataene bidrager med ud over spørgeskemaet.

Differentieret scaffolding som forudsætning for sokratisk vejledning

Spørgeskemaet viser, at nybegyndere generelt rapporterer et større oplevet læringsudbytte end mere erfarne studerende. Interviewene nuancerer dette billede ved at pege på, at behovet for vejledning ikke kun varierer mellem gruppen af nybegyndere og øvrige, men også *internt* hos den enkelte studerende afhængigt af situation og energiniveau.

Rasmus, der ingen forudgående programmeringserfaring har, beskriver et behov for at kunne justere agentens vejledningsniveau eksplicit:

> *Hvis man kunne bryde det op i forskellige niveauer … jeg vil gerne have en lille smule vejledning, eller jeg vil ikke have noget hjælp, eller jeg vil gerne have meget hjælp, fordi jeg er helt ny (Rasmus)*

Han trækker selv en parallel til menneskelig vejledning og fremhæver, at en underviser "*justerer jo også lidt eftertid*" afhængigt af, hvor den studerende befinder sig. Meklas, der har programmeringserfaring fra Java og C++, formulerer behovet fra modsatte ende: gruppen er "*ret blandet i forhold til niveau*", og en initial niveaubestemmelse kunne gøre svarene mere målrettede.

Denne pointe er ny i forhold til spørgeskemaet og har en direkte teoretisk implikation. I et Vygotskij-inspireret perspektiv forudsætter scaffolding, at vejledningen møder den lærende i zonen for nærmeste udvikling, men denne zone er ikke fast, hverken på tværs af studerende eller hen over et arbejdsforløb.

Agentens nuværende design implementerer én fast sokratisk strategi for alle brugere, hvilket i praksis betyder, at scaffoldingen for nogle vil være for tilbageholdende og for andre for hjælpende. Interviewene peger på, at kalibrering af scaffolding-niveauet kunne betragtes som et selvstændigt designprincip.

Verifikationsbyrden som ny kognitiv belastning hos nybegyndere

Et særligt mønster fremtræder i Rasmus' beskrivelse af, hvordan agenten kan opleves, når den studerende endnu ikke har en faglig referenceramme at vurdere svarene op imod:

> *Hvis du ikke forstår det, du laver, og du bruger koden på den måde, ligesom i starten … så kan du faktisk ikke bruge det til noget. Så bliver det faktisk et udueligt værktøj (Rasmus)*

Han beskriver desuden, hvordan han i en tidligere kontekst brugte "*rigtig meget energi på siden og confirmede, at det jeg lavede, var rigtigt*". Dette peger på en mekanisme, som spørgeskemaets

frustrationsmarkører ikke afdækker: agenten reducerer som intenderet extraneous load gennem strukturering af problemløsningsprocessen, men introducerer samtidig en ny form for kognitiv belastning hos nybegynderen, nemlig byrden ved at skulle verificere agentens svar uden tilstrækkelig faglig baggrund.

Designet antager dermed implicit en forudgående faglig forankring hos den studerende, en grundlæggende forståelse af, hvad et begreb som variabel, if-sætning eller metode refererer til, og en intuition for, hvordan kode kan opføre sig. Denne forankring fungerer som et "stillads", agentens spørgsmål kan hænges op på: når agenten spørger "*hvad sker der med variablens værdi her?*", kan den studerende med forankring koble spørgsmålet til en mental model, hun allerede besidder, og bruge dialogen til at præcisere, udfordre eller udvide denne model. Spørgsmålet bliver dermed germane load, kognitiv indsats, der bidrager til begrebsdannelsen.

Hos den helt uerfarne nybegynder er denne forankring imidlertid endnu ikke etableret. Når Rasmus modtager et sokratisk spørgsmål, har han ikke en forhåndsforståelse at koble spørgsmålet til; spørgsmålet bliver i stedet en selvstændig kognitiv opgave, han skal løse oveni den oprindelige problemløsning. Han skal samtidig afkode, hvad agenten spørger om, gætte på, hvad det "rigtige" svar kunne være, og vurdere, om hans gæt overhovedet giver mening — alt sammen uden et fagligt fundament at vurdere det op imod. Dialogen lægger sig dermed ikke som en støtte under den eksisterende forståelse, men som en ekstra opgave ved siden af problemet.

Denne pointe knytter sig direkte til kognitiv belastningsteoris skelnen mellem extraneous og germane load (Paas et al., 2003). Agentens dialogiske form er kun germane, hvis den studerende har et fagligt grundlag at hænge dialogen op på. Uden dette grundlag bliver dialogen extraneous: den optager kognitiv kapacitet uden at bidrage til begrebsdannelsen, og kan i værste fald fortrænge den oprindelige problemløsning, fordi arbejdshukommelsen ikke kan rumme begge opgaver samtidig.

Sekventering af produktion og dialog som didaktisk variabel

Spørgeskemaet viser, at både Code Review-modulet og opgavegenereringsmodulet vurderes højt (gennemsnit ca. 4,1), mens chat-modulet scorer lavere på flere indikatorer for oplevet læringsudbytte. Dette mønster fremstår umiddelbart paradoksalt: chat-modulet er det modul, hvor den sokratiske dialog er mest rendyrket, og det er dette designprincip, projektet teoretisk har prioriteret højest. Hvorfor scorer det modul, der mest direkte realiserer projektets centrale designprincip, lavere end de øvrige?

Interviewene tilbyder en forklaring, der ikke kunne udledes af kvantitative data alene, og som peger på en didaktisk variabel, agenten i sin nuværende form ikke eksplicit håndterer: rækkefølgen mellem den studerendes egen kognitive produktion og den dialogiske vejledning.

Mønsteret i de tre interviews

Meklas formulerer pointen mest direkte i sin beskrivelse af, hvordan Code Review-modulet adskiller sig fra hans øvrige AI-brug:

> *Så sidder jeg med koden, som jeg ligesom har lært at prøve at gøre det hele, så godt jeg nu kan, men så får jeg fra den … det der hvorfor (Meklas)*

Det er ikke vejledningsformen i sig selv, der vurderes positivt, men det forhold at vejledningen indtræffer *efter* den studerendes egen kognitive investering. Meklas beskriver, hvordan han som konsekvens har integreret modulet i sin daglige praksis, også efter selve evalueringen, til feedback på fejlhåndtering og kommentarer, områder hvor han allerede har en hypotese, agenten kan reagere på.

Søren formulerer det samme mønster med en anden begrundelse. For ham er det visuelle element afgørende: han fremhæver gentagne gange, at Code Review-modulets struktur med "*problem, løsning, lidt kode*" gør feedbacken anvendelig som *konstruktiv kritik af det, man har lavet*. Han kontrasterer dette med chat-modulet, hvor han oplever, at agenten stiller for mange spørgsmål, før han har fået lejlighed til at formulere sin egen forståelse: "*der er seks spørgsmål. Du er kommet i gang med det første endnu, nej.*" Hans kritik handler ikke om, at agenten stiller spørgsmål, men om at spørgsmålene kommer på et tidspunkt, hvor han endnu ikke har en position at svare ud fra.

Rasmus beskriver det samme fænomen, men fra nybegynderens perspektiv. Han savner muligheden for at justere agenten op og ned afhængigt af, om han er på vej ind i et nyt emne (*hvor han har brug for forklaring først*) eller på vej ud af et problem (*hvor han har brug for vejledning på sit eget udkast*). Hans pointe er, at den samme studerende har brug for forskellig sekventering på forskellige tidspunkter.

Teoretisk forankring

Mønsteret kan analyseres gennem tre af projektets etablerede teoretiske perspektiver, som tilsammen forklarer, hvorfor sekventeringen har den observerede effekt.

Kognitiv belastningsteori. I projektets teoretiske ramme er programmering identificeret som et domæne med høj intrinsic load, hvor den lærende samtidig skal håndtere syntaks, logik og problemløsning (Berssanette & De Francisco, 2022). Sokratisk dialog er designet til at reducere extraneous load gennem strukturering af problemløsningsprocessen.

Men dialog er i sig selv en kognitiv operation: den studerende skal forstå spørgsmålet, holde sin egen tankegang fast, og koble spørgsmålet til en faglig kontekst.

Når dialogen indledes uden at den studerende har et konkret artefakt at hænge sig op på, må vedkommende holde *både* problemet, *agentens spørgsmål* og *sin egen hypotese* aktivt i arbejdshukommelsen samtidig. Det er præcis den type kognitiv samtidighed, kognitiv belastningsteori advarer imod (Paas et al., 2003). I Code Review-modulet fungerer koden som ekstern repræsentation af problemet; den studerende behøver ikke holde den i hovedet, og dialogen kan dermed bidrage germane load uden at konkurrere om kapacitet.

Generation effect. Projektets teoretiske ramme refererer eksplicit til generation effect (Fiorella & Mayer, 2016, citeret i Liang et al., 2025) som begrundelse for øvelsesmodulets graduerede hint-struktur. Det samme princip er relevant her: aktiv selvgenerering af svar er associeret med bedre læringsudbytte end passiv aflæsning. I Code Review-konteksten har den studerende allerede genereret en løsning, og den efterfølgende dialog konsoliderer og udfordrer denne genererede forståelse. I chat-modulet er rækkefølgen modsat: dialogen kommer før genereringen og risikerer i værste fald at substituere den. Generation effect-perspektivet antyder dermed, at chat-modulets potentielle læringseffekt forudsætter, at den studerende først genererer en hypotese og at agentens prompt-struktur i dag ikke tvinger denne sekvens igennem.

Scaffolding. I scaffolding-traditionen (Wood, Bruner & Ross, 1976) er det centrale princip, at støtten skal være kontingent på den lærendes aktuelle position. Wood og kollegaer beskriver, hvordan en kompetent vejleder først *observerer* den lærendes forsøg og derefter intervenerer på det niveau, hvor støtten er mest produktiv. Code Review-modulet realiserer denne sekvens, fordi den studerendes kode fungerer som det artefakt, agenten kan observere og respondere på. Chat-modulet realiserer den ikke i samme grad, fordi agenten her må intervenere uden et artefakt at observere. Den må stille spørgsmål uden at vide, hvor den studerende befinder sig i sin tænkning.

Designmæssig implikation

Disse tre teoretiske perspektiver konvergerer om en pointe, der lader sig formulere som et betinget designprincip: *sokratisk dialog understøtter refleksion og problemløsning mest effektivt, når den følger efter en hypotese eller et udkast fra den studerende, snarere end når den indleder problemløsningsprocessen*.

For chat-modulet har dette en konkret implementeringsmæssig konsekvens. I sin nuværende form indleder agenten ofte dialogen med rene undersøgende spørgsmål ("hvad ved du allerede om interfaces?"), før den studerende har formuleret en position. En revideret prompt-struktur kunne i stedet opfordre den studerende til *først* at formulere et udkast, et eksempel eller en hypotese f.eks. ved at agenten åbner med en formulering som "*skriv et kort bud på, hvordan du*

forstår det, og så kigger vi på det sammen". Først herefter aktiveres den sokratiske spørgsmålssekvens, nu med et konkret artefakt at hænge dialogen op på.

En sådan revision ville bringe chat-modulet tættere på den sekventering, der i Code Review-modulet allerede fungerer, og samtidig bevare modulets dialogiske karakter. Det ville også adressere den frustration, som flere studerende beskriver i de åbne svar, at chat-modulet stiller for mange spørgsmål, før der er noget at tale om.

Dette fund udfordrer en implicit antagelse i projektets oprindelige design – nemlig at sokratisk dialog i sig selv er en tilstrækkelig didaktisk strategi. Empirien viser, at dialog som princip skal suppleres med eksplicit håndtering af dens sekventering i den studerendes problemløsningsproces.

Sammenhæng med de øvrige fund

Sekventeringspointen står ikke isoleret, men spiller sammen med de øvrige interviewfund. Den hænger direkte sammen med pointen om verifikationsbyrden: når en nybegynder ikke har genereret en hypotese, har vedkommende heller ikke et grundlag for at vurdere agentens spørgsmål, hvilket forstærker den kognitive belastning. Den hænger også sammen med pointen om differentieret scaffolding: forskellige studerende har brug for forskellig sekventering på forskellige tidspunkter, og en fast prompt-struktur kan ikke imødekomme denne variation alene.

Tilsammen peger dette på, at projektets centrale teoretiske princip, sokratisk dialog, ikke skal forkastes, men kvalificeres med en yderligere designvariabel: *hvornår* i den studerendes problemløsningsproces dialogen indsættes. Denne kvalificering er en empirisk genereret tilføjelse til den teoretiske ramme og udgør et af projektets potentielle bidrag til diskussionen af, hvordan sokratisk AI-vejledning kan designes i programmeringsundervisning.

Agenten i et AI-økosystem

Et tværgående mønster i alle tre interviews er, at studerende ikke vurderer agenten som et isoleret værktøj, men som ét blandt flere i et personligt AI-økosystem. Alle tre studerende refererer eksplicit til ChatGPT, Claude, Copilot eller Gemini som sammenligningsgrundlag og beskriver, hvordan de over tid har udviklet egne strategier for, hvornår de bruger hvilket værktøj.

Søren har f**.**eks. udviklet en personlig regel for sin brug af Claude: "*Jeg vil ikke se noget kode. Hvis jeg sidder fast, kan jeg spørge dig om det, men jeg vil ikke se noget kode*". Meklas beskriver en bevidst arbejdsdeling: "*Jeg vælger den til nogen ting, og den her til noget andet*". Rasmus har slået Copilot fra i Visual Studio, fordi den "*ødelagde min læring*".

Denne kontekstuelle indsigt har to implikationer for projektets analyse. For det første betyder det, at studerendes vurderinger af agenten i spørgeskemaet implicit foretages i forhold til alternative AI-værktøjer, som de allerede bruger.

Når en studerende vurderer agentens vejledning positivt, sker det således på baggrund af en sammenligning, som spørgeskemaet ikke gør synlig. For det andet peger det på, at agentens succes i undervisningskonteksten ikke alene afhænger af dens interne kvaliteter, men af, om den udfylder en *funktion*, som studerendes øvrige værktøjer ikke dækker.

For Søren og Meklas er den funktion især pensumforankringen, der skærper feedbacken og gør den fagligt afgrænset. For Rasmus er den specifikke værdi mindre tydelig, fordi han endnu ikke har en faglig referenceramme, der gør pensumforankringen mærkbar.

Denne forskel knytter sig til den foregående pointe om verifikationsbyrden: pensumforankring opleves som en kvalitet for studerende, der allerede ved nok til at vurdere relevansen af et svar, mens den for de helt nye studerende er mindre central end selve forklaringen.

Sammenfatning af interviewfundene

Hvor spørgeskemaet kortlægger, *hvad* studerende mener om agentens forskellige funktioner, viser interviewene, *under hvilke betingelser* disse vurderinger fremkommer. De fire temaer peger samlet på, at agentens didaktiske virkning er kontingent på en række forhold, som agentens nuværende design ikke selv adresserer: den studerendes forforståelse, deres samtidige adgang til alternative AI-værktøjer, om de allerede har produceret kode at hænge dialogen op på, og om de befinder sig i en udforskningsfase eller en frustrationsfase.

Denne kontingens har en metodisk betydning for, hvordan projektets designprincipper formuleres. I overensstemmelse med Barab og Squires (2004) lægger interviewene op til, at projektets designprincipper ikke kan formuleres som universelle påstande om sokratisk AI-vejledning, men må formuleres som *betingede* principper, der specificerer, hvornår en given designstrategi understøtter refleksion og problemløsning, og hvornår den bør modificeres. Denne pointe danner grundlag for den følgende diskussion.

8 Diskussion

Dette kapitel diskuterer projektets fund i lyset af den problemformulering, der har styret undersøgelsen: hvordan en sokratisk AI-agent kan designes til at understøtte studerendes refleksion og problemløsning, og hvordan studerende uden programmeringserfaring oplever interaktionen med en sådan agent i introducerende programmeringsundervisning på datamatikeruddannelsen. Analysen har gennem den kvantitative kortlægning af studerendes vurderinger og den tematiske analyse af tre interviews identificeret en række mønstre, der både bekræfter og kvalificerer eksisterende viden om sokratisk AI-vejledning. Diskussionen samler disse mønstre med henblik på at klargøre, hvilke bidrag de tilfører feltet, og under hvilke betingelser de identificerede designprincipper kan forventes at gælde.

Diskussionen er struktureret i seks dele. Først diskuteres projektets bidrag i forhold til den eksisterende litteratur (8.1). Dernæst overvejes, i hvilket omfang de identificerede principper kan tænkes at have gyldighed ud over datamatikerkonteksten (8.2). Tredje afsnit indeholder en kritisk vurdering af projektets metode og datagrundlag (8.3), og fjerde afsnit diskuterer den organisatoriske dimension af agentens anvendelse i et undervisningsmiljø (8.4). Endelig overvejes implikationerne for praksis (8.5) og for videre forskning (8.6).

8.1 Projektets bidrag i forhold til den eksisterende litteratur

Litteraturreview identificerede en række spændinger og åbninger i forskningen i sokratiske AI-tutorer i programmeringsundervisning. Eksisterende studier dokumenterer, at sokratisk designede AI-agenter kan stimulere kritisk tænkning og refleksion sammenlignet med direkte-svar-modeller (Favero et al., 2024), og at studerende oplever sokratisk feedback som understøttende for akademisk integritet (Nahar, 2025). Samtidig viser Goldin, Pedroncini og Sigman (2017), at sokratisk dialogs effekt er betinget af den lærendes evne til at abstrahere og generalisere. Reviewet pegede dog også på, at feltet er præget af en relativt løs operationalisering af "sokratisk metode" og af begrænset empirisk viden om, under hvilke betingelser sokratisk AI-vejledning understøtter læring i konkrete undervisningskontekster.

Nærværende projekt bidrager til denne diskussion på fire måder, der tilsammen kvalificerer, snarere end bekræfter, det sokratiske princip.

For det første identificerer projektet sekventering som en selvstændig didaktisk variabel. Hvor litteraturen typisk behandler sokratisk dialog som en samlet egenskab ved AI-agenten, viser interviewene, at det afgørende ikke alene er, om agenten stiller spørgsmål, men hvornår i den studerendes problemløsningsproces spørgsmålene indsættes. Sammenligningen mellem chat-modulet og Code Review-modulet, det sidste har den højeste samlede vurdering (gennemsnit

4,06), mens det første scorer lavest på flere indikatorer for læringsudbytte, peger på, at sokratisk dialog er mest produktiv, når den følger efter en hypotese eller et udkast fra den studerende. Denne pointe lader sig forankre i scaffolding-traditionen (Wood, Bruner & Ross, 1976), generation effect (Fiorella & Mayer, 2016) og kognitiv belastningsteori (Paas et al., 2003), men er ikke eksplicit udfoldet i den nuværende AI-tutor-litteratur, der i højere grad fokuserer på dialogformens kvalitet end på dens timing.

For det andet introducerer projektet begrebet verifikationsbyrde som en ny kognitiv belastning hos den helt uerfarne nybegynder. Litteraturen lægger op til, at sokratisk dialog reducerer extraneous load gennem strukturering af problemløsningsprocessen. Interviewet med Rasmus viser imidlertid, at dialogen samtidig kan introducere en kognitiv opgave, nemlig byrden ved at skulle vurdere agentens svar uden tilstrækkelig faglig baggrund. Dette nuancerer Goldin et al. (2017): det er ikke kun abstraktionsevnen, der betinger sokratisk dialogs effekt, men også den studerendes kapacitet til at verificere agentens udsagn. Pointen knytter sig direkte til kognitiv belastningsteoris skelnen mellem extraneous og germane load: dialogen er kun germane, hvis den studerende har et fagligt grundlag at hænge dialogen op på. Denne præcisering er ikke en afvisning af det sokratiske princip, men en specificering af dets gyldighedsbetingelser i netop den målgruppe, projektet undersøger: *studerende uden forudgående programmeringserfaring*.

For det tredje peger projektet på, at den studerendes brug af agenten foregår i et AI-økosystem af parallelle værktøjer; ChatGPT, Claude, Copilot, Gemini, som de studerende allerede har integreret i deres arbejdspraksis. Litteraturen behandler typisk AI-tutorer som isolerede artefakter, der vurderes på egne præmisser. Interviewene viser, at studerendes vurderinger implicit er komparative, og at agentens succes afhænger af, om den udfylder en funktion, som de øvrige værktøjer ikke dækker. For Søren og Meklas er denne funktion særligt pensumforankringen og Code Review-modulets struktur. Pointen har metodisk betydning for AI-tutor-forskningen: studier der måler studerendes vurderinger uden at gøre denne komparative ramme synlig, risikerer at tilskrive agenten egenskaber, der i virkeligheden er differentielle.

For det fjerde formulerer projektet i forlængelse af Barab og Squire (2004) sine designprincipper som betingede frem for universelle. Sekventering, differentieret scaffolding og hensyn til verifikationsbyrden er ikke generelle regler for, hvordan sokratiske AI-agenter bør designes, men principper der specificerer hvornår en given strategi understøtter studerendes refleksion og problemløsning, og hvornår den bør modificeres. Denne tilgang adresserer den løse operationalisering af sokratisk metode, som litteraturreview identificerede som et problem i feltet.

8.2 Perspektivering ud over datamatikerkonteksten

Selvom undersøgelsen er gennemført i en specifik undervisningskontekst, 1. semester på datamatikeruddannelsen, er de identificerede principper ikke logisk afhængige af denne kontekst. Sekventerings-pointen, verifikationsbyrden og den betingede karakter af sokratisk dialog hviler på generelle mekanismer i kognitiv belastningsteori og scaffolding-traditionen, der i princippet er gyldige for ethvert domæne, hvor sokratisk AI-vejledning anvendes med begyndere.

Introducerende programmeringsundervisning er karakteriseret ved en stejl indledende læringskurve, hvor den studerende på kort tid skal tilegne sig et nyt fagligt sprog, og hvor risikoen for ukritisk at acceptere fejlbehæftede svar er tilsvarende høj. Når det faglige sprog endnu ikke er etableret hos den lærende, opstår en specifik sårbarhed: den studerende har hverken det begrebsapparat, der gør sokratiske spørgsmål produktive, eller den faglige referenceramme, der gør det muligt at vurdere agentens svar. Dette dobbelte underskud, fraværet af både hypotese og verifikationsgrundlag, gør den helt uerfarne nybegynder særligt udsat for at acceptere AI-genererede svar uden tilstrækkelig forforståelse. Mønsteret er imidlertid ikke unikt for programmering, men genfindes i en lang række andre uddannelseskontekster, hvor begyndere møder et formaliseret fagsprog under tidspres. Det gælder f.eks. matematik og statistik, juridisk argumentation og medicinsk klinisk ræsonnement, hvor den lærende ligeledes må arbejde med et endnu ikke etableret begrebsapparat. I sådanne kontekster må sokratisk AI-vejledning forventes at støde på den samme grundlæggende udfordring som i nærværende undersøgelse: kombinationen af manglende fagligt sprog og manglende verifikationsgrundlag kan gøre dialogen ineffektiv eller ligefrem kontraproduktiv, hvis den anvendes uden differentiering.

På samme måde er AI-økosystem-perspektivet generelt overførbart. Studerende på tværs af uddannelser anvender i stigende grad parallelle generative AI-værktøjer som ChatGPT, Claude eller Copilot, og enhver fagspecifik AI-tutor vil derfor blive vurderet relativt til disse alternativer. Det betyder, at en didaktisk designet agents legitimitet og oplevede nytte i mindre grad afhænger af dens absolutte kvaliteter og i højere grad af, om den udfylder en funktion, som de almene værktøjer ikke dækker, typisk gennem fagspecifik forankring, pensumafgrænsning eller en differentieret pædagogisk strategi. Denne komparative ramme er ikke en metodisk særegenhed ved nærværende undersøgelse, men en strukturel betingelse, som AI-tutor-forskning bør forholde sig til.

Samtidig skal denne overførbarhed formuleres med forsigtighed. De konkrete empiriske mønstre er identificeret hos tre studerende i en specifik kontekst og udgør, i Barab og Squires (2004) terminologi, en humble theory: en lokalt forankret forståelse, der kan informere designet af sokratiske AI-tutorer i andre kontekster, men som forudsætter ny empirisk afprøvning før den kan

generaliseres. Det er fundenes form, betingede principper, der specificerer kontingens, snarere end deres konkrete indhold, der udgør projektets bidrag på tværs af uddannelseskontekster.

8.3 Kritisk stillingtagen til metode og datagrundlag

Projektets fund må læses i lyset af en række metodiske begrænsninger. Spørgeskemaet omfatter 12 studerende, og interviewene er gennemført med tre udvalgte deltagere. Resultaterne kan derfor ikke generaliseres til den bredere studerendepopulation, og evalueringen er baseret på en relativt kort interaktion med agenten, og en længerevarende anvendelse i et samlet undervisningsforløb kunne tænkes at give andre indsigter om, hvordan studerende oplever interaktionen over tid.

Hertil kommer min dobbeltrolle som designer, underviser og forsker. Selvom interviewene er struktureret narrativt og deltagerne ikke er introduceret til projektets analytiske kategorier, har de deltaget i evalueringen som led i mit eget undervisningsforløb. Dette kan have påvirket deres villighed til at fremføre kritik. De empiriske mønstre, der træder frem i interviewene, særligt verifikationsbyrden og AI-økosystem-perspektivet, er imidlertid ikke forudset i designet, hvilket understøtter, at fundene ikke alene reproducerer mine egne designintentioner.

Endelig baserer flere indikatorer sig på selvrapporterede vurderinger af interaktionen og det oplevede udbytte. Dette er konsistent med problemformuleringens fokus på studerendes oplevelse af interaktion, men indebærer, at fundene afspejler studerendes egne beskrivelser snarere end uafhængigt observerede mønstre. Interviewene supplerer spørgeskemaet med situerede beskrivelser, men udgør et lille empirisk udsnit. Tilsammen forstærker disse forhold pointen om, at projektets bidrag er identifikationen af betingede principper og ikke validering af universelle påstande.

8.4 Agentens samspil med undervisningskonteksten

Agenten er fra starten designet til at indgå i et undervisningsmiljø, ikke som et selvstændigt læringsværktøj løsrevet fra den øvrige undervisning. Et tværgående mønster i interviewene er, at agentens didaktiske virkning ikke alene afhænger af dens design, men også af det samspil med undervisningen, den indgår i. Selvom projektets empiriske grundlag ikke tillader systematiske udsagn om de organisatoriske rammer som sådan, gør interviewene en række forhold ved dette samspil synlige, som det er analytisk relevant at adressere. De følgende tre delafsnit behandler underviserens rolle, studerendes parallelle AI-praksis og spørgsmålet om integration i undervisningsforløbet.

8.4.1 Underviserens rolle

Rasmus' beskrivelse af agenten som mest værdifuld i situationer, hvor underviseren ikke kan nå rundt til alle, peger på, at agenten ikke erstatter underviseren, men udfylder en funktion i en arbejdsdeling, der ikke er eksplicit designet. De lavere scorer på indikatorer som oplevet forståelse og selvtillid peger i samme retning. Rasmus tilføjer, at mundtlig forklaring og diskussion med medstuderende udgør en dimension, agenten ikke kan dække. Tilsammen indikerer dette, at agenten fungerer som ét element i en bredere vejledningsøkologi, hvor underviseren fortsat har en central rolle. Hvilken rolle dette præcis er, og hvordan den bør afstemmes med agentens, er ikke afgjort af designet. Designet kunne i en senere iteration eksplicitere underviserens position, f.eks. ved at give undervisere indsigt i mønstre i studerendes interaktion med agenten, eller ved at strukturere undervisningsforløbet så agenten anvendes på specifikke tidspunkter, hvor dens funktion er didaktisk velbegrundet. Det er imidlertid afgørende at understrege, at sådanne valg om undervisningens tilrettelæggelse ikke følger automatisk af agentens design, men kræver selvstændig didaktisk overvejelse i den konkrete undervisningskontekst.

8.4.2 Studerendes parallelle AI-praksis

Interviewene viser, at studerende ikke møder agenten i et vakuum, men som ét værktøj i en allerede etableret AI-praksis. Søren, Meklas og Rasmus har alle udviklet personlige strategier for, hvilke AI-værktøjer de bruger til hvad, strategier, der er formet uden for undervisningens rammer, og som agenten må forholde sig til. En didaktisk integration af AI-værktøjer kan derfor ikke alene baseres på det konkrete værktøjs egenskaber, men må forholde sig til den samlede AI-praksis, studerende bringer ind i undervisningen. Dette stiller krav om, at undervisere udvikler didaktisk-strategiske kompetencer i forhold til at navigere et AI-økosystem, snarere end alene at vælge et bestemt værktøj.

Konkret indebærer dette, at undervisere må udvikle indsigt i, hvilke værktøjer studerende anvender til hvilke formål, hvilke faglige antagelser disse værktøjer indlejrer i de studerendes arbejdsproces, og hvordan de studerendes individuelle AI-strategier varierer på tværs af et hold. Dette kan ses som en ny dimension af det velkendte didaktiske spørgsmål om undervisningsdifferentiering. Hvor differentiering klassisk har handlet om at tilpasse undervisning til studerendes faglige niveau, læringspræferencer eller forudsætninger, tilføjer AI-praksissen et yderligere lag: studerende møder ikke længere undervisningen alene med en faglig forforståelse, men også med en allerede etableret arbejdsdeling med generative værktøjer, der medierer deres tilgang til stoffet. En differentieret undervisning må derfor forholde sig ikke kun til, hvor de studerende fagligt befinder sig, men også til, hvilke AI-praksisser de bringer med sig, og hvordan disse interagerer med den didaktisk designede agent.

Det rejser samtidig nye problemstillinger. For det første kan studerendes parallelle brug af generelle AI-værktøjer underminere den didaktisk designede agents fagspecifikke forankring, hvis de studerende vælger at sammenligne agentens svar med svar fra mere alment trænede modeller. For det andet kan ulighed i studerendes AI-kompetence og i deres adgang til betalte premium-versioner forstærke eksisterende forskelle i forudsætninger, snarere end udjævne dem. For det tredje vil indsigt i studerendes AI-praksis i sig selv kræve nye former for dialog mellem undervisere og studerende, dialog som ikke nødvendigvis er en del af den eksisterende undervisningspraksis. Disse problemstillinger peger på, at didaktisk-strategiske kompetencer i et AI-økosystem ikke alene er et spørgsmål om at vælge mellem værktøjer, men om at udvikle en pædagogisk praksis, der kan forholde sig analytisk til de studerendes samlede AI-mediede arbejdsmåder.

8.4.3 Integration i undervisningsforløbet

Interviewene peger på, at agentens nytteværdi varierer afhængigt af, hvor i den studerendes problemløsningsproces den anvendes, illustreret af forskellen mellem chat-modulet og Code Review-modulet. Denne pointe har en didaktisk pendant på forløbsniveau: agentens integration i et undervisningsforløb bør tilsvarende tage stilling til, hvornår i forløbet den introduceres, og hvilke didaktiske aktiviteter den supplerer. Designet specificerer ikke dette, og fundet om sekventering peger på, at en sådan specificering ville være didaktisk meningsfuld. En naturlig forlængelse af projektet vil derfor være at undersøge, hvordan agenten kan integreres i et samlet kursusdesign, hvor undervisning, øvelser, instruktørstøtte og AI-vejledning indgår som komplementære elementer.

Tilsammen peger disse forhold på, at den didaktiske integration udgør en selvstændig designvariabel, der i nærværende projekt er holdt åben, men som de empiriske fund gør synlig som genstand for videre didaktisk og organisatorisk udviklingsarbejde.

8.5 Implikationer for praksis

For undervisere, der overvejer at integrere sokratiske AI-værktøjer i programmeringsundervisning, peger projektets fund på tre praktiske opmærksomhedspunkter. Først at sokratisk vejledning ikke er en egenskab ved en model, men resultatet af et vedvarende designarbejde, hvor sekventeringen mellem den studerendes egen produktion og den dialogiske vejledning bør tænkes ind eksplicit. Dernæst at agentens nytteværdi for nybegynderen forudsætter en minimal faglig forankring; helt uerfarne studerende kan have brug for forklarende støtte forud for sokratisk dialog, ikke i stedet for. Endelig at den fagspecifikke forankring, f.eks. pensumafgrænsning, fungerer som en differentierende kvalitet, der adskiller en didaktisk designet agent fra generelle AI-assistenter.

Selvom de praktiske opmærksomhedspunkter er formuleret med udgangspunkt i programmeringsundervisning, hviler de underliggende mekanismer ikke på fagets specifikke karakteristika, men på generelle forhold, der gør sig gældende i mange introducerende undervisningskontekster. Sekventeringspointen, at dialog forudsætter en hypotese eller et udkast at hænge sig op på, er principielt overførbar til discipliner, hvor studerende ligeledes møder en stejl læringskurve og skal lære at tænke inden for et endnu ikke etableret fagligt sprog.

Det gælder f.eks. matematik og statistik, hvor sokratisk AI-vejledning kunne tage udgangspunkt i den studerendes egen løsningsstrategi frem for at formulere problemet på den studerendes vegne. Det gælder tilsvarende juridisk argumentation eller medicinsk klinisk ræsonnement, hvor den dialogiske støtte med fordel kan følge efter den studerendes egen formulering af et argument eller en differentialdiagnose.

Verifikationsbyrden, hvor begynderen mangler grundlag for at vurdere agentens svar, er en lige så generel udfordring i alle fag, hvor AI-værktøjer indføres tidligt i et uddannelsesforløb. Den praktiske implikation, at forklarende støtte må gå forud for sokratisk dialog, er dermed ikke en programmeringsspecifik anbefaling, men en generel didaktisk overvejelse, undervisere på tværs af fag bør forholde sig til. Det samme gælder den fagspecifikke forankring: i ethvert fag, hvor studerende anvender generelle AI-værktøjer parallelt, vil en didaktisk designet agent skulle differentiere sig gennem netop den faglige afgrænsning, som de generelle assistenter ikke kan levere. Principperne fungerer dermed som en form for didaktisk grundskabelon, hvis konkrete operationalisering må tilpasses det enkelte fags genstand og den specifikke undervisningskontekst.

8.6 Implikationer for videre forskning

En naturlig forlængelse af nærværende projekt er en anden DBR-iteration, hvor de identificerede principper omsættes til konkrete redesign-elementer og afprøves empirisk på ny. En revideret prompt-struktur i chat-modulet kunne f.eks. opfordre den studerende til først at formulere et udkast eller en hypotese, før den sokratiske dialog aktiveres, hvilket ville bringe modulet tættere på den sekventering, Code Review-modulet allerede realiserer. Tilsvarende kunne en eksplicit niveaubestemmelse ved sessionens begyndelse adressere det behov for differentieret scaffolding, både Rasmus og Meklas formulerer fra hver sin position.

På et mere overordnet plan peger projektet på et behov for studier, der eksplicit inddrager studerendes parallelle AI-praksis som analytisk variabel, og som kombinerer selvrapporterede vurderinger med uafhængigt observerede mål for refleksion og problemløsning. Dette ville styrke det empiriske grundlag for at vurdere, under hvilke betingelser sokratisk AI-vejledning understøtter refleksion og problemløsning i konkrete undervisningssituationer.

9 Konklusion

Dette projekt har undersøgt, hvordan en sokratisk AI-agent kan designes til at understøtte studerendes refleksion og problemløsning, og hvordan studerende uden programmeringserfaring oplever interaktionen med en sådan agent i introducerende programmeringsundervisning på datamatikeruddannelsen. Projektet er gennemført som et eksplorativt første designforløb inden for en designbaseret forskningstilgang, hvor en konkret AI-agent er udviklet, afprøvet og evalueret blandt 12 studerende på 1. semester, suppleret af tre semistrukturerede interviews.

På spørgsmålet om, hvordan en sokratisk AI-agent kan designes, demonstrerer projektet, at sokratisk vejledning ikke er en egenskab ved sprogmodellen, men resultatet af et systematisk designarbejde. Forskellen kan formuleres præcist: en sprogmodel kan instrueres til at fremstå sokratisk gennem en prompt, men hvorvidt agenten faktisk fastholder den dialogiske form, hvorvidt dens spørgsmål er fagligt forankrede, og hvorvidt dialogen indsættes på didaktisk velbegrundede tidspunkter i den studerendes problemløsningsproces, afgøres ikke af modellen, men af det omkringliggende design. Den udviklede agent realiserer dette gennem en kombination af curriculumforankret retrieval-augmented generation, en to-model pipeline med komplementære styrker i strukturering og pædagogisk formatering, samt fire moduler, der hver realiserer den sokratiske tilgang i forskellige læringssituationer. Designet operationaliserer projektets teoretiske ramme, særligt den sokratiske metode, scaffolding-traditionen og kognitiv belastningsteori, i et konkret IKT-baseret læringsartefakt og udgør et eksempel på, hvordan generativ AI kan rammesættes didaktisk i et fagspecifikt undervisningsforløb.

På spørgsmålet om, hvordan studerende oplever interaktionen, viser de empiriske fund et nuanceret billede. Studerende vurderer agenten højt på dimensioner relateret til guidet læring og refleksion, og særligt nybegyndere rapporterer et meningsfuldt læringsudbytte. Code Review-modulet opnår den højeste samlede vurdering, mens chat-modulet, hvor den sokratiske dialog er mest rendyrket, paradoksalt scorer lavest på flere indikatorer for læringsudbytte. Interviewene afdækker, at dette mønster ikke skyldes en svaghed ved sokratisk dialog som sådan, men at dialogen forudsætter et fagligt artefakt eller en hypotese fra den studerende for at være kognitivt produktiv. For den helt uerfarne nybegynder kan dialogen samtidig introducere en ny kognitiv belastning, en verifikationsbyrde, idet den studerende skal vurdere agentens svar uden tilstrækkelig faglig referenceramme. Studerendes oplevelse af agenten er desuden formet af deres parallelle brug af andre AI-værktøjer, hvilket gør vurderingen implicit komparativ.

På spørgsmålet om designprincipper udleder projektet på baggrund af det empiriske grundlag følgende betingede principper for sokratisk AI-baseret vejledning:

1. *Sekventering*. Sokratisk dialog understøtter refleksion og problemløsning mest effektivt, når den følger efter en hypotese eller et udkast fra den studerende, snarere end når den indleder problemløsningsprocessen.
2. *Differentieret scaffolding*. Vejledningsniveauet bør kunne kalibreres efter den studerendes aktuelle behov, ikke alene efter en fast prompt-struktur.
3. *Hensyn til verifikationsbyrden*. For studerende uden faglig referenceramme bør forklarende støtte gå forud for sokratisk dialog, ikke erstatte den.
4. *Fagspecifik forankring*. Pensumafgrænsning og curriculumforankring fungerer som differentierende kvaliteter, der adskiller en didaktisk designet agent fra generelle AI-assistenter.
5. *Bevidsthed om AI-økosystem*. Agentens didaktiske virkning afhænger af dens position i den samlede AI-praksis, studerende bringer med ind i undervisningen.

I overensstemmelse med Barab og Squires (2004) begreb om humble theory er disse principper formuleret som betingede frem for universelle: de specificerer, hvornår en given designstrategi understøtter refleksion og problemløsning, og hvornår den bør modificeres.

En anden DBR-cyklus kunne afprøve de fem principper systematisk gennem konkrete redesign-elementer. Sekventering kan testes ved at omstrukturere chat-modulet, så agenten først anmoder den studerende om en hypotese eller et kodemæssigt udkast, før den dialogiske vejledning aktiveres, og effekten kan sammenlignes med den nuværende åbne dialogform. Differentieret scaffolding kan operationaliseres som en eksplicit niveaubestemmelse ved sessionens begyndelse, hvor agenten kalibrerer prompt-strukturen efter den studerendes selvvurderede erfaringsniveau. Hensyn til verifikationsbyrden kan adresseres gennem et indledende forklaringsmodul, som tilbyder begrebsmæssig støtte til den helt uerfarne nybegynder forud for sokratisk dialog. Fagspecifik forankring kan styrkes ved at udbygge agentens metadata-struktur, så curriculumforankringen bliver mere granulær og synlig for den studerende.

Endelig kan bevidsthed om AI-økosystem informere designet ved at agenten eksplicit positionerer sin funktion i forhold til andre AI-værktøjer, f.eks. ved at fremhæve, hvad den kan, som generelle AI-assistenter ikke kan. Sammen ville disse redesign-elementer udgøre grundlaget for en empirisk afprøvning, der kan bevæge principperne fra empirisk genererede hypoteser i retning af konsoliderede designprincipper inden for den specifikke læringsøkologi, projektet undersøger.

Projektets samlede bidrag består dermed af to komplementære dele: et fungerende, didaktisk funderet IKT-artefakt, der demonstrerer, hvordan sokratisk AI-vejledning kan operationaliseres i en konkret undervisningskontekst, og et sæt empirisk genererede designprincipper, der

kvalificerer, frem for blot bekræfter, den eksisterende forskning i sokratiske AI-tutorer. Som første iteration i en designbaseret forskningsproces peger projektet samtidig fremad mod en næste iteration, hvor de redesign-elementer, der her er skitseret, kan afprøves empirisk og bidrage til en gradvis konsolidering af principperne inden for den specifikke læringsøkologi, datamatikeruddannelsen udgør.

10 Litteraturliste

Anderson, T., & Shattuck, J. (2012). Design-based research: A decade of progress in education research? Educational Researcher, 41(1), 16–25. https://doi.org/10.3102/0013189X11428813

Ang, B. H., Gollapalli, S. D., & Ng, S.-K. (2023). Socratic question generation: A novel dataset, models, and evaluation. I A. Vlachos & I. Augenstein (Red.), Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (s. 147–165). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.eacl-main.12

Badger, M., López, N., & Nissen, C. F. R. (2026). The Impact of GenAI Chatbots on Student Learning in Higher Education: A Literature Review. International Journal of Technology in Education, 9(1), 43–69. https://doi.org/10.46328/ijte.5111

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. The Journal of the Learning Sciences, 13(1), 1–14. https://doi.org/10.1207/s15327809jls1301_1

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming is hard – or at least it used to be: Educational opportunities and challenges of AI code generation. I Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (s. 500–506). https://doi.org/10.1145/3545945.3569759

Berssanette, J. H., & De Francisco, A. C. (2022). Cognitive load theory in the context of teaching and learning computer programming: A systematic literature review. IEEE Transactions on Education, 65(3), 440–449. https://doi.org/10.1109/TE.2021.3127215

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Delić, H., & Bećirović, S. (2016). Socratic method as an approach to teaching. European Researcher, 111(10), 511–517. https://doi.org/10.13187/er.2016.111.511

Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry. Educational Researcher, 32(1), 5–8. https://doi.org/10.3102/0013189X032001005

Er, E., Akçapınar, G., Bayazıt, A., Noroozi, O., & Banihashem, S. K. (2024). Assessing student perceptions and use of instructor versus AI-generated feedback. British Journal of Educational Technology. https://doi.org/10.1111/bjet.13558

Farrokhnia, M., Banihashem, S. K., Noroozi, O., & Wals, A. (2024). A SWOT analysis of ChatGPT: Implications for educational practice and research. Innovations in Education and Teaching International, 61(3), 460–474. https://doi.org/10.1080/14703297.2023.2195846

Favero, L., Pérez-Ortiz, J. A., Käser, T., & Oliver, N. (2024). Enhancing critical thinking in education by means of a Socratic chatbot. I ECAI'24: International Workshop on AI in Education and Educational Research (AIEER). https://doi.org/10.48550/arXiv.2409.05511

Fiorella, L., & Mayer, R. E. (2016). Eight ways to promote generative learning. Educational Psychology Review, 28(4), 717–741. https://doi.org/10.1007/s10648-015-9348-9

Geyer, B., Stampfl, R., & Hauser, E. (2024). Chatbots & Socrates: Dialogues in learning. ERCIM News, 136. https://ercim-**news**.ercim.eu/en136/special/chatbots-socrates-dialogues-in-learning

Goldin, A. P., Pedroncini, O., & Sigman, M. (2017). Producing or reproducing reasoning? Socratic dialog is very effective, but only for a few. PLoS ONE, 12(3), e0173584. https://doi.org/10.1371/journal.pone.0173584

Hashmi, A., Bal, A. S., & Rahman, S. (2024). Generative artificial intelligence in higher education: Exploring student perspectives on the role of AI in supporting learning. Education and Information Technologies. https://doi.org/10.1007/s10639-024-12876-9

Kvale, S., & Brinkmann, S. (2015). Interview: Det kvalitative forskningsinterview som håndværk (3. udg.). Hans Reitzels Forlag.

Liang, W., Zhao, L., Yan, L., Sha, L., Yu, J., Chen, G., Tsai, Y.-S., Wei, B., & Gašević, D. (2025). Beyond answers: How AI chatbots reasoning shapes student learning. Manuskript indsendt til publicering. https://doi.org/10.48550/arXiv.2502.13789

Lyu, W., Wang, Y., Chung, T. R., Sun, Y., & Zhang, Y. (2024). Evaluating the effectiveness of LLMs in introductory computer science education: A semester-long field study. I Proceedings of the Eleventh ACM Conference on Learning @ Scale (s. 63–74). https://doi.org/10.1145/3657604.3662036

Misfeldt, M. (2010). 'Forestillet læringsvej' i IT-baserede pædagogiske udviklingsprojekter. Dansk Pædagogisk Tidsskrift, 58(4), 42–52.

Nahar, N. (2025). Exploring student perspectives on AI-generated feedback using a Socratic Method chatbot. Journal of Learning Development in Higher Education, Special Issue 37: ALDinHE Conference Proceedings. https://doi.org/10.47408/jldhe.vi37.1338

Noroozi, O., Soleimani, S., Farrokhnia, M., & Banihashem, S. K. (2024). Generative AI in education: Pedagogical, theoretical, and methodological perspectives. International Journal of Technology in Education, 7(3), 373–385. https://doi.org/10.46328/ijte.845

OpenAI. (2025, 4. september). Why language models hallucinate. Hentet 27. april 2026, fra https://openai.com/da-DK/index/why-language-models-hallucinate/

Paas, F., Renkl, A., & Sweller, J. (2003). Cognitive load theory and instructional design: Recent developments. Educational Psychologist, 38(1), 1–4. https://doi.org/10.1207/S15326985EP3801_1

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Vygotsky, L. S. (1978). Mind in society: The development of higher psychological processes. Harvard University Press.

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. Journal of Child Psychology and Psychiatry, 17(2), 89–100. https://doi.org/10.1111/j.1469-7610.1976.tb00381.x

Yan, L., Sha, L., Zhao, L., Li, Y., Martinez-Maldonado, R., Chen, G., Li, X., Jin, Y., & Gašević, D. (2024). Practical and ethical challenges of large language models in education: A systematic scoping review. British Journal of Educational Technology, 55(1), 90–112. https://doi.org/10.1111/bjet.13370

11 Bilagsoversigt

• Bilag 1	Spørgeskema – AI-agent	2 sider
• Bilag 2	Resultater – Excel	17 sider
• Bilag 3	Resultater – analyseret	12 sider
• Bilag 4	Analyserapport – AI-agent	5 sider
• Bilag 5	Test af AI-kodeagent	32 sider
• Bilag 6	Meklas – transskription	8 sider
• Bilag 7	Meklas – interview-oversigt	7 sider
• Bilag 8	Rasmus – transskription	14 sider
• Bilag 9	Rasmus – interview-oversigt	3 sider
• Bilag 10	Søren – transskription	14 sider
• Bilag 11	Class library – C#-format	22 sider
• Bilag 12	Søren – interview-oversigt	2 sider
• Bilag 13	Class library – JSON-format	4 sider