Studerendes anvendelse af generativ kunstig intelligens i programmeringsundervisning

Anders Kalsgaard Møller, Aalborg Universitet https://orcid.org/0000-0003-4581-4094
Kristine Bundgaard, Aalborg Universitet https://orcid.org/0000-0002-3085-5398

Abstract

Flere studerende søger ind på videregående uddannelser inden for it- og STEM-området og skal introduceres til programmering. Mange studerende oplever læringskurven i introducerende programmeringskurser som meget stejl og er særligt udfordrede i forståelsen af, hvordan de forskellige programmeringsbegreber skal anvendes i praksis. Undervisningen er ofte meget øvelsesbaseret, og underviseren er ofte udfordret med at hjælpe alle studerende på holdet. Her har chatbots baseret på generativ kunstig intelligens (GenAI) vist sig som en mulig støtte. I dette studie undersøges de programmeringsaktiviteter, som studerende anvender GenAI til, og teknologiens potentielle indvirkning på de studerendes læringsudbytte. Undersøgelsens empiri er baseret på observationer fra et kursus i programmering og en spørgeskemaundersøgelse, hvor (N=30) studerende har deltaget. Resultaterne fra undersøgelsen peger på, at AI-baserede chatbots som ChatGPT kan spille en værdifuld rolle i programmeringsundervisning ved at støtte studerende i at udvikle deres computationelle praksisser og problemløsningsevner, når teknologien indgår i et samskabende system med andre aktører. Samtidig kan AI-baserede chatbots være med til at aflaste underviseren ved at svare på studerendes spørgsmål, så underviseren har mulighed for at indgå i andre læringsaktiviteter.

ISSN: 1903-248X

Engelsk abstract

More students are applying for admission to IT- and STEM-related higher education programmes and need to be introduced to programming. Many students experience the learning curve as steep and are particularly challenged in understanding how different programming concepts can be applied in practice. Programming courses often focus on programming exercises, making it very demanding for the teacher to assist all students in a timely manner. Here, chatbots based on generative artificial intelligence (GenAI) can be a potential source of support. In this study, we explore the programming activities that students use GenAI for as well as the technology's potential impact on the students' learning outcome. The empirical material includes observations from an introductory course in programming and a survey where (N=30) students participated. The results from the study indicate that AI-based chatbots such as ChatGPT can play a valuable role in the teaching of programming by scaffolding students in developing their computational practices and problem-solving skills when they are included as an element in a coactive person-environment system. At the same time, AI-based chatbots can contribute by aiding the teacher in answering student questions, enabling the teacher to contribute to other learning activities.

Introduktion

Med den stigende efterspørgsel efter digitale kompetencer er der et voksende antal studerende, som søger ind på videregående uddannelser indenfor it- og STEM-området (Danmarks Statistik, 2024). Dette medfører, at flere studerende skal introduceres til programmering, ikke kun inden for områder som ingeniørvidenskab, datalogi og naturvidenskab, men også inden for humaniora og samfundsvidenskab.

Mange studerende oplever imidlertid læringskurven i introducerende programmeringskurser som meget stejl, da de både skal forstå den konceptuelle ramme, arbejde med logiske ræsonnementer, computationelle praksisser og syntaksen af et tekstbaseret programmeringssprog (Butler & Morgan, 2007; Lahtinen et al, 2005).

En mulig støtte er den nye generation af chatbots baseret på generativ kunstig intelligens (GenAI), såsom ChatGPT, Gemini og Copilot, som ved hjælp af store sprogmodeller og maskinlæring er i stand til at producere indhold som tekst, billeder eller kode baseret på forespørgsler eller prompts (Lim et al., 2023). Studier har blandt andet vist, at ChatGPT kan bruges til at hjælpe programmører med at generere kode og fejlfinde i forbindelse med softwareudvikling (Nazir & Wang, 2023). Sprogmodellen kan forholde sig til kodesyntaksen og dermed hjælpe med at identificere fejl, ligesom den kan bruges til at optimere eksisterende kode og komme med forbedringsforslag. Den kan ligeledes forklare fejl, kommentere på kode og skrive dokumentation. På grund af teknologiens evner til at generere og beskrive kode er man også begyndt at undersøge, hvorvidt og hvordan den kan anvendes i programmeringsundervisning til at understøtte studerende og dette særligt på introducerende programmeringskurser (Becker et al., 2023; Hellas et al., 2023; Lau et al., 2023). AI-baserede chatbots kan fx give studerende eksempler på, hvordan opgaver kan løses, understøtte undervisere i at lave opgaver til studerende, give uddybende forklaringer af forskellige dele af kode eller programmeringsbegreber, hjælpe mod ”kode-skrive-blokade” eller oversætte kompiler-fejlbeskeder (Becker et al., 2023; Leinonen, 2023).

Denne artikel tager udgangspunkt i en teoretisk forståelse af denne støtte som en samskabende stilladsering (Mascolo, 2005), hvor den studerendes udvikling ses som et produkt af interaktioner mellem individet og dets miljø, hvor både individet og miljøet bidrager til læringsprocessen. I forbindelse med opgaveløsningen sker stilladseringen traditionelt igennem forskellige typer af feedback, nemlig feedback fra underviser, medstuderende, softwareudviklingsmiljøet og kompileren (computerprogram, som oversætter programmer, der er skrevet i et bestemt programmeringssprog, til maskinsprog) (Butler & Morgan, 2007).

Mens underviseren kan hjælpe med at svare på spørgsmål og give gode råd i udviklings- og kodeprocessen, kan softwareudviklingsmiljøet give mere specifik feedback på programmeringssyntaks og kodekonstruktion. Softwareudviklingsmiljøet kan ligeledes give feedback på loops, betingelser og andre programmeringsbegreber ift. implementering. Endelig kan kompileren fortælle, hvis der er syntaksproblemer, og i visse tilfælde hvordan problemet løses (Butler & Morgan, 2007). Udfordringen er her, at den feedback, man får fra softwareudviklingsmiljøet og kompileren, begrænser sig til at være relateret til syntaksen og simpel implementering, mens de ikke kan give feedback på design af programmet, og fx hvordan begreberne kan anvendes til at løse specifikke problemer. Dermed overlades disse spørgsmål til underviseren, som ofte skal dække store hold og dermed risikerer at blive flaskehals for de studerendes læring.

I følgende studie anvender vi en tilgang, hvor GenAI anvendes aktivt i et introduktionskursus i programmering for kandidatstuderende på en cand.it.-uddannelse inden for humaniora og samfundsvidenskab. Dette sker med henblik på at besvare følgende forskningsspørgsmål:

- Hvilke specifikke programmeringsaktiviteter anvender studerende GenAI til i det konkrete forløb?
- Hvordan oplever de deltagende studerende GenAI's påvirkning af læringsudbyttet i programmeringsundervisningen?

I følgende afsnit vil vi præsentere den teoretiske ramme, som ligger til grund for vores videre analyse. Dernæst følger en beskrivelse af vores undersøgelsesdesign, undersøgelsens resultater samt en diskussion og konklusion.

Teoretisk ramme

Programmeringsundervisning og anvendelse af GenAI

Til at beskrive de forskellige aspekter, som studerende skal lære i forbindelse med programmeringsundervisningen, benyttes begrebet computational thinking (CT). Begrebet blev først introduceret af Papert (1980) og blev mere alment kendt med Cuny et al. (2010), som definerede begrebet som:

> "The thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Cuny et al., 2010, s.1).

Mange forskellige forskere har efterfølgende beskæftiget sig med at definere, hvilke kompetencer og processer begrebet skal indeholde (Barr & Stephenson, 2011; Grover & Pea, 2013; Selby & Woollard, 2013). I denne artikel tager vi udgangspunkt i Brennan og Resnicks (2012) framework, som inddeler CT i computationelle begreber, praksisser og perspektiver. Computationelle begreber er de begreber, som anvendes, når man skriver programkode, nemlig sekvenser, løkker, parallelisme, begivenheder, betingelser, operatører og data. Når man arbejder med programmering, er det fx vigtigt at forstå, at et program er opbygget af en sekvens af steps, som computeren udfører, og at man, når man skal gemme data i ens kode, typisk gør det ved hjælp af forskellige variabeltyper. Ligeledes kan man benytte sig af kodestrukturer såsom løkker, hvor programmet gennemløber samme kodesekvens flere gange.

De computationelle praksisser er de praksisser, som anvendes, når man designer og udvikler programmer. Når man designer og udvikler, er det bl.a. vigtigt at kunne arbejde inkrementelt og iterativt, dvs. at kunne bryde projektet ned i mindre dele og arbejde på dem én ad gangen, hvor man løbende forbedrer dem gennem iterationer. Man skal ligeledes kunne arbejde med testning og fejlfinding/debugging, som består i at afprøve kode for at se, om denne fungerer som forventet, og finde og rette eventuelle fejl (bugs). Man skal desuden kunne genbruge og remixe, dvs. tage eksisterende projekter, kode eller idéer og tilpasse eller udvide dem til nye formål. Endelig skal man kunne arbejde med abstraktion og modularisering, som handler om at forenkle kompleksitet ved at bryde et stort problem ned i mindre, håndterbare dele og genbruge disse dele. Det involverer at skabe generaliserede løsninger eller blokke af kode.

De computationelle perspektiver er de holdninger, forståelser og tilgange, som udvikles, når man programmerer, og består af at udtrykke sig gennem teknologien, at stille spørgsmål til teknologien og føle sig forbundet til andre gennem teknologien, fx ved at skabe for eller med andre. Der er altså her tale om mere overordnede holdninger og forståelser af teknologien og af andre gennem teknologien.

Når vi i introduktionen beskrev, at de studerende i mindre grad er udfordrede ift. at forstå begreberne, men snarere ift. at anvende dem, handler det om, hvordan begreberne anvendes i de forskellige praksisser. Brennan og Resnick (2012) beskriver dette som:

> "The intersection between computational thinking concepts and computational thinking practices" (Brennan og Resnick, 2012, s.23).

I denne sammenhæng argumenterer Brennan og Resnick for, at det ikke er nok, at man som udvikler ved, hvordan en løkke fungerer; man skal også være i stand til at anvende denne viden

i ens kode på en meningsfyldt måde, at kunne forstå andres brug af løkker og kunne fejlfinde kode, hvor løkker indgår. I relation til de computationelle perspektiver handler det om at kunne forstå, hvordan tingene virker, og således hvilke begreber der indgår, og hvordan de kan anvendes, når man skal udtrykke sig eller skabe med andre.

I artiklen bidrager dette framework til en analytisk forståelse af de udfordringer, som de studerende har, og hvordan de lærer programmering. Det vil desuden blive brugt til en kategorisering af de typer af processer, de studerende anvender GenAI til.

I et forsøg på at imødekomme de udfordringer, studerende oplever i introduktionen til programmering, har man gennem mange år arbejdet med forskellige undervisningstilgange (Fincher, 1999; Hu, 2014; Robins et al., 2003) samt udviklet og anvendt forskellige programmeringssprog og metoder målrettet undervisning (Schulte & Bennedsen, 2006; Weintrop & Wilensky, 2017). Man diskuterer ikke blot *hvordan*, men også *hvad* de studerende skal lære i programmeringsundervisningen (Lu & Fletcher, 2009; Lye & Koh, 2014).

Som vi har været inde på ovenfor, handler de udfordringer, studerende møder, typisk ikke så meget om at forstå de forskellige programmeringsbegreber såsom variable, løkker og betingelser, men snarere hvordan disse anvendes i praksis (Lahtinen et. al., 2005). Her er der fokus på, at studerende anvender programmeringsbegreberne sammen med logiske ræsonnementer for at kunne løse virkelighedsnære problemer. Dette gør udenadslære uegnet til programmering, da ethvert problem kræver en unik løsning ved hjælp af de programmeringsværktøjer, de studerende har lært (Butler & Morgan, 2007). Af samme grund er programmeringsundervisningen ofte bygget op omkring øvelser eller problemløsning, hvor de studerende selv skal forsøge at løse forskellige problemstillinger ved hjælp af de programmeringsbegreber, som de har fået undervisning i (Romero et al., 2017). I denne forbindelse er det vigtigt at sikre, at problemernes eller opgavernes sværhedsgrad ligger inden for de studerendes nærmeste udviklingszone (Vygotsky, 1978), og at de får den støtte, der skal til for at løse opgaverne. Dette kan AI-baserede chatbots potentielt hjælpe med.

I en undersøgelse af programmeringsunderviseres holdninger til brugen af AI-baserede chatbots var holdningerne delte i relation til, hvorvidt man på den lange bane bør omfavne eller tage afstand fra dem (Lau et al., 2023). Modstandere mod brugen af AI-baserede chatbots baserer bl.a. deres argumentation på problemer med integritet i forhold til udprøvning (Becker et al., 2023; Daun & Brings, 2023; Lau et al., 2023), ophavsret og kreditering, når sprogmodellerne fodres med indhold, samt udfordringer med bias og dårlige kodevaner (Becker et al., 2023). Eksempler på bias i forbindelse med kodegenerering kan fx være mindre effektive eller forældede måder at skrive kode på, hvilket opstår som resultat af kvaliteten af

træningsdata. Endelig frygter man, at studerende bliver for afhængige af AI-baserede chatbots til selvstændigt at kunne producere kode (Becker et al., 2023). Blandt dem, som går ind for at omfavne AI-baserede chatbots i programmeringsundervisning, fokuseres der på den individuelle hjælp og feedback, studerende potentielt kan få, og at de kan hjælpe undervisere med tidskrævende opgaver (Lau et al., 2023).

Anvendelsen af AI-baserede chatbots kan potentielt ændre, hvordan man underviser, og hvad man underviser i. Fremadrettet kan fokus på selvstændigt at kunne skrive kode fx aftage, da AI-baserede chatbots kan gøre det for én, så man i stedet har fokus på at læse og efterredigere AI-genererede kodestykker (Daun & Brings, 2023; Lau et al., 2023). Desuden kan der blive større fokus på at kunne udarbejde kravspecifikationer samt designe og teste med henblik på at kunne anvende automatisk genereret kode (Daun & Brings, 2023). Det kunne fx betyde, at man i undervisningen har fokus på opgaver, hvor de studerende samarbejder med chatbots, hvilket samtidig giver bedre muligheder for at lave mere åbne opgaver, hvor studerende selv skal designe systemer med hjælp fra AI (Lau et al., 2023).

Mens der har været talrige akademiske diskussioner og teoretiske artikler om de potentielle implikationer af GenAI for uddannelse generelt og for programmeringsundervisning specifikt, er antallet af empiriske studier fortsat begrænset. Dette studie bidrager med empiri om, hvordan studerende anvender GenAI i forbindelse med programmeringsundervisning, som vil danne udgangspunkt for en analyse og diskussion af GenAI's rolle i undervisningen, og hvordan teknologien potentielt påvirker de studerendes læringsudbytte.

Samskabende stilladsering

I artiklen tages der udgangspunkt i Mascolos (2005) teori om samskabende stilladsering. I traditionelle beskrivelser af stilladsering (Wood et al., 1976) fokuserer man på, hvordan en mere vidende person kan støtte et lærende individ i at udføre opgaver, som vedkommende ikke ville kunne klare uden hjælp. Ved samskabende stilladsering ser man derimod bredere på, hvordan elementer i person–miljø-systemet, som ligger uden for en individuel aktørs direkte kontrol, kan lede eller kanalisere opbygningen af handlinger på nye og uforudsete måder (Mascolo, 2005). Inden for samskabende stilladsering arbejder man med tre typer stilladsering: social stilladsering, økologisk stilladsering og selv-stilladsering.

Social stilladsering handler om, hvordan individet udvikler sig i samspil med andre personers instruktioner. Instruktioner kan variere fra simple opmuntringer til udførlige instruktioner og imitationer. I programmeringsundervisningen kan både underviseren og medstuderende bidrage til social stilladsering af den studerende i tilegnelsen af ny viden. Dette kan ske direkte

gennem målrettede undervisningsinstruktioner og hjælp til opgaverne eller mere utilsigtet gennem uformelle interaktioner.

Økologisk stilladsering handler om, hvordan individets position eller relation til det fysiske eller sociale miljø påvirker individet. Der kan være tale om en stilladsering, der minder om Gibsons (1977) *affordance*-begreb, hvor fysiske egenskaber ved miljøet kan tilbyde forskellige tjenesteligheder, som understøtter, at individet opnår ny viden, ligesom det kan være selve opgaven eller objektet, der interageres med, som kan strukturere og lede udviklingen af nye måder at handle og tænke på. I programmeringsundervisningen vil dette fx omfatte nogle af de egenskaber, som findes i softwareudviklingsmiljøet, som på forskellig vis tilbyder muligheder og begrænsninger, ligesom softwareudviklingsmiljøets farvekoder for fx variable og andre begreber sammen med diverse fejlkoder kan være med til at stilladsere de studerendes forståelse af programmering. Også selve de opgaver, som stilles, kan i form af deres iboende egenskaber stilladsere den studerende til nye erkendelser.

Selv-stilladsering handler om, hvordan et individs egne handlinger i interaktion med omgivelserne understøtter nye handlinger og viden. Det sker, når en persons handlinger direkte eller indirekte ændrer miljøet på en måde, som foreslår nye betydninger eller mentale processer. Denne ændring kommer dermed til at fungere som et stillads for personens udvikling. Gennem deltagelse kan individer udvikle en større forståelse af en aktivitet og dermed blive udstyret til at håndtere lignende eller beslægtede aktiviteter og underaktiviteter. Her skelnes der mellem tre forskellige typer, nemlig kognitiv stilladsering, brobyggende stilladsering og analogisk stilladsering.

Kognitiv stilladsering omfatter situationer, hvor individets egne handlinger medfører en ændring af miljøet, som så skaber ny viden. I programmeringsundervisning kan det være den studerende, som ændrer en smule på koden for at se, hvilke ændringer det medfører til programmet, hvorigennem den studerende tilegner sig nye forståelser.

Man taler om brobyggende stilladsering, når individet bygger videre på tidligere viden eller forståelse for på den måde at konstruere ny viden. I programmeringsundervisningen kan den studerende fx anvende viden om, hvordan en for-løkke virker, til at hjælpe med at forstå, hvordan en while-løkke virker (to forskellige typer af løkker, hvor den ene løber x antal gange, mens den anden løber, så længe en bestemt betingelse er opfyldt).

Analog stilladsering minder om brobyggende stilladsering, men hvor man laver en mere direkte sammenligning mellem fx to problemer, hvor man bruger erfaringer fra et tidligere problem til at løse et nyt problem. I programmeringsundervisning kan de studerende fx blive stillet en

opgave, hvor de skal lave et program, hvor de ved hjælp af en løkke skal generere talrækken 1,2,3,4,5,6 og efterfølgende blive stillet en opgave om at generere talrækken 2,4,6,8,10,12. Ved hjælp af analogisk stilladsering skulle de studerende gerne indse, at de kan anvende samme tilgang som i første opgave og på den måde stilladseres de til at indse, at de blot skal gange outputtet med 2. Indser de det ikke i første omgang, kan der være brug for ekstra stilladsering fra underviser, før det bliver tydeligt for dem. Med endnu et eksempel vil de så typisk uden problemer finde frem til løsningen på, hvordan de laver talrækken 3,6,9,12,15,18. Således har de enten alene gennem selv-stilladsering eller med støtte gennem social stilladsering tilegnet sig ny viden eller forståelse.

I denne artikel vil samskabende stilladsering blive brugt som en teoretisk ramme for at forstå, hvordan studerende opbygger forståelse for begreber, praksisser og perspektiver. Ligeledes vil begrebet blive brugt til at belyse, hvordan AI-baserede chatbots som ChatGPT kan være med til at understøtte læring eller omvendt udgøre en hindring for læring.

I denne forbindelse vil vi i analysen kategorisere interaktionen mellem den studerende og AI-baserede chatbots som en form for social relation og dermed som en mulighed for social stilladsering. Dette gør vi ud fra en betragtning om, at interaktionen med GenAI har størst lighed med de forskellige underkategorier, som er beskrevet under social stilladsering hos Mascolos (2005) og ud fra Sharples (2023) pointe om, at interaktionen med AI-baserede chatbots bør opfattes mere som en social samtale og forklaringsproces end en sekvens af prompts. Vi underkender dog ikke, at menneskelig interaktion er anderledes, og at den studerende givetvis forholder sig til, at vedkommende kommunikerer ved at skrive på et tastatur og gennem et interface, som også giver nogle miljømæssige påvirkninger af interaktionen. Dermed skal denne kategorisering blot ses som et analytisk greb og ikke som et udtryk for en overbevisning om menneskelige kvaliteter ved AI-baserede chatbots på trods af dens evne (eller menneskers tilbøjelighed) til at antromorfisere.

Undersøgelsesdesign

Undersøgelsen er baseret på spørgeskemadata og observationer indsamlet i foråret 2024 i forbindelse med et introduktionskursus i programmering. Kurset udbydes på en humanistisk cand.it.-uddannelse på Aalborg Universitet og er et 10 ECTS-kursus, hvor de studerende bliver undervist i grundlæggende programmering, software engineering og prototyping. Kurset består af en blanding af forelæsninger og programmeringsøvelser.

Som forberedelse til forelæsningerne skulle de studerende se forberedelsesvideoer, læse udvalgte artikler og gennemføre programmeringsøvelser. Som led i kurset skulle de udvikle

deres eget programmeringsprojekt i grupper på 2-3 studerende. Projektarbejdet var inspireret af en designbaseret læringstilgang (Gómez Puente et al., 2013) og lignede forslagene i Romero et al. (2017), hvor der benyttes en åben læringsproces, der følger en designproces. I begyndelsen af kurset blev de studerende undervist i grundlæggende programmeringsbegreber, såsom variabler, betingelser, løkker, begivenheder og funktioner ved hjælp af micro:bit Makecode-miljøet (Microsoft, 2024). Derefter blev de introduceret til tekstbaseret programmering gennem programmeringsmiljøet Processing (2024). Efter de første fire forelæsninger, hvor hensigten var, at de studerende opnåede en grundlæggende forståelse af programmeringssproget og begreberne, blev de kort introduceret til GenAI, mere specifikt ChatGPT, som et værktøj, de havde mulighed for at bruge som en støtte til deres programmeringsprojekt. Der blev givet et eksempel på, hvordan man kunne bruge ChatGPT til at forklare en kodesekvens og samtidig pointeret, at såfremt de valgte at bruge GenAI, skulle de være opmærksomme på de kendte udfordringer såsom bias og hallucinationer (Hellas et al., 2023). Desuden skulle de dokumentere deres brug. Det blev også italesat, at de skulle være påpasselige med blot at kopiere længere kodesekvenser direkte fra ChatGPT eller kodesekvenser, de ikke forstod, men i stedet bruge den som en sparringspartner til at forstå kode og til at komme med løsningsforslag.

Kurset var opbygget, så de studerende i starten af modulet blev introduceret til de forskellige programmeringsbegreber, hvor underviser skiftevis gennemgik de enkelte begreber og efterfølgende stillede opgaver med begreberne, som de studerende så skulle løse. Hensigten var, at de studerende først skulle forstå begreberne, og hvordan man anvender dem i simple programmer. Derefter blev de studerende løbende introduceret til mere åbne problemer, hvor de selv skulle forholde sig til, hvilke programmeringsbegreber der skulle anvendes og hvordan. På denne måde blev de studerende løbende mere og mere udfordrede i at træne praksisser i takt med, at de fik en bedre forståelse af de forskellige begreber. Efter nogle sessioner med fokus på træning af praksisser blev de studerende introduceret til det programmeringsprojekt, som de skulle arbejde på i den resterende del af kurset. Efter introduktionen blev der løbende afsat tid til at arbejde på projektet i undervisningen, så der blev varieret mellem øvelser og projektarbejde. Det vil sige, at de studerende — ud over at skulle lave programmeringsøvelser med henblik på at træne forskellige praksisser — skulle begynde at forholde sig til, hvordan deres programmeringsprojekt skulle designes og med hvilket formål. Dermed lagde undervisningen op til en progression, hvor aktiviteterne løbende skiftede fra at have fokus på forståelse af begreber til senere at fokusere mere på praksisser og perspektiver.

Ved kursets afslutning besvarede de studerende (N=30) en online spørgeskemaundersøgelse, som var designet i SurveyXact. De studerende havde en gennemsnitsalder på 32. Ud af de 30

studerende identificerede 21 sig som kvinde og 8 som mand. En respondent ønskede ikke at oplyse køn. Spørgeskemaet indeholdt foruden demografiske spørgsmål åbne spørgsmål om deres brug af ChatGPT i programmeringskurset. De studerende blev stillet følgende spørgsmål:

- Hvilke konkrete opgaver/processer har du brugt ChatGPT til i forbindelse med Programmering?
- Er der forskel på, hvad du bruger din underviser, medstuderende og ChatGPT til?
- Hvad har du fået ud af at bruge ChatGPT?
- Hvilke udfordringer har du haft i forbindelse med brugen af ChatGPT?
- Tror du, din brug af ChatGPT har påvirket din indlæring i kurset?

Ud over de kvalitative besvarelser fra spørgeskemaet omfatter studiets empiriske grundlag også undervisningsobservationer. Undervisningsobservationerne var ustrukturerede observationer foretaget af underviseren, som vil blive brugt til at give supplerende forklaringer eller kontekst til besvarelserne i spørgeskemaet. Besvarelserne fra spørgeskemaundersøgelsen er blevet behandlet ved hjælp af tematisk analyse inspireret af Braun & Clarke (2006). Besvarelserne til hvert spørgsmål i spørgeskemaet er blevet kodet og efterfølgende grupperet efter temaer, som er fremkommet induktivt baseret på besvarelserne i spørgeskemaet. Dette foregik ved at besvarelserne til hvert spørgsmål først blev gennemlæst og derefter kodet. Herefter blev de kodede besvarelser gennemgået og grupperet i overordnede temaer efter sammenhæng og mønstre i besvarelserne. Til sidst blev udsagnene under hvert tema sammenfattet til en beskrivelse af temaet. Observationerne blev efterfølgende brugt til at forklare konteksten og som supplerende forklaringer til temaerne.

Resultater

I det følgende afsnit vil vi først præsentere resultaterne fra spørgeskemaundersøgelsen. Derefter følger et sammenfattende analyse- og diskussionsafsnit.

Opgaver og processer med ChatGPT

I spørgeskemaet blev de studerende bedt om at forholde sig til, hvilke konkrete opgaver/processer de havde brugt ChatGPT til i forbindelse med kurset. Langt de fleste besvarelser fordelte sig mellem fire typer af opgaver eller processer: Til at skabe en bedre forståelse af eksisterende kode eller programmeringsbegreber, som hjælp til at implementere specifikke funktioner og kodestykker, til at identificere og rette fejl i koden samt som inspiration til, hvad de kunne tilføje til deres projekter. I nedenstående giver vi eksempler på de fire temaer.

Til at skabe en bedre forståelse af eksisterende kode eller programmeringsbegreber

For at skabe en bedre forståelse for begreberne kunne de studerende bede ChatGPT om at forklare specifikke begreber, og hvordan de anvendes, såsom arrays og forskellige datatyper. Det gjaldt fx den følgende studerende:

> ”Jeg brugte bl.a. ChatGPT til at få uddybet/beskrevet noget jeg er i tvivl om fx. faglige begreber. Det kunne være arrays eller float.”

De kunne også bede den om at give eksempler på, hvordan begreberne kunne anvendes. Det blev observeret i undervisningen, at ChatGPT primært blev brugt til dette, når de studerende skulle genbesøge begreberne med henblik på at anvende dem i deres projekter. Dog blev det i enkelte tilfælde også anvendt i starten af kurset, da begreberne blev introduceret.

Hjælp til at implementere specifikke funktioner og kodestykker

En anden måde, hvorpå de studerende brugte ChatGPT, var som en hjælp til, hvordan de skulle implementere specifikke funktioner og kodestykker. Det giver denne studerende fx udtryk for:

> ”Forslag til kodninger - hvis vi ikke vidste hvordan vi skulle kode en specifik funktion, havde chatGPT altid et forslag. Det kunne ikke altid bruges, men kunne pege os i en retning.”

Selvom værktøjet nogle gange foreslog overkomplicerede løsninger, gav de studerende i deres besvarelser udtryk for, at brugen af GenAI støttede deres læring, især når de havde en oplevelse af at sidde fast. Desuden italesatte flere studerende, at denne hjælp var nyttig grundet deres begrænsede programmeringserfaringer, og at ChatGPT’s forslag ofte blev brugt til at bekræfte de studerende i, at de var på rette vej:

> ”Min viden og mine færdigheder med at programmere er begrænset. Undervejs i arbejdet med miniprojektet, anvendte jeg derfor ChatGPT hovedsageligt som et understøttende værktøj eller som en vejledende støtte, som kunne hjælpe mig, når jeg ikke kunne komme videre med koden.”

Underviseren observerede ligeledes, at de studerende ofte brugte ChatGPT til at komme med input til, hvordan de skulle implementere forskellige dele af koden. De studerende kunne ikke altid direkte omsætte de input, de fik, men det fik dem typisk i gang med at eksperimentere med kodetilpasning fremfor blot at vente på stilladsering fra underviser.

Identificere og rette fejl i koden

Flere studerende gav i deres besvarelser i spørgeskemaet udtryk for, at de brugte ChatGPT til at identificere og rette fejl i koden (debugging) samt at diagnosticere mere generelle problemer og finde løsninger på dem (fejlfinding).

> "Debugging hvis der er en fejl, vi ikke kunne gennemskue ift. syntaksen, så satte vi koden ind og bad den lave fejlfinding."

Dette forekom både i forhold til logiske fejl (fejl, som giver forkerte resultater på grund af en fejl i programmets logik eller algoritme) og syntaksfejl (fejl i koden, der forhindrer programmet i at køre, fordi kodestrukturen ikke følger de grammatiske regler for programmeringssproget). Ligeledes blev GenAI ifølge spørgeskemabesvarelserne brugt til at optimere koden.

Underviseren observerede, at de studerende ofte havde meget svært ved at afkode de fejlmeddelelser, som udviklingsmiljøet kom med, og få rettet fejlene, hvorfor de havde stor glæde af den mere dialogiske og uddybende forklaring, som ChatGPT kunne bidrage med.

Inspiration til projekter

Flere af de studerende gav i spørgeskemaet udtryk for, at de havde brugt ChatGPT til at få inspiration til, hvad de kunne tilføje til deres projekter og til at komme med forslag til, hvad deres program kunne indeholde. Det gjaldt fx denne studerende:

> "Til at udvide min horisont og dermed mulighederne for at lave et spændende spil i Processing."

Forinden undervisers introduktion af GenAI i undervisningen var der ganske få observationer af, at teknologien blev anvendt. Generelt blev der kun observeret en meget sporadisk brug af GenAI i forbindelse med kursets øvelser, mens langt de fleste studerende anvendte teknologien i forbindelse med projektarbejdet, dog i varierende omfang. Dette kan skyldes, at øvelserne var meget stilladserede, og at de studerende havde adgang til løsningerne, hvilket formentlig har mindsket behovet for at anvende ChatGPT. Selvom de studerende anvendte GenAI til både at forstå computationelle begreber, praksisser og perspektiver, så handlede størstedelen af besvarelserne i spørgeskemaet om computationelle praksisser, såsom debugging og processer og med at få implementeret kodestykker og funktioner.

Brug af underviser, medstuderende og ChatGPT

I spørgeskemaet blev de studerende også bedt om at svare på, om der var forskel på, hvad de brugte henholdsvis underviser, medstuderende og ChatGPT til. Her blev underviser italesat

som en autoritet og en, man kunne stole på, gav de korrekte svar. Omvendt blev det påpeget, at man skulle forholde sig mere kritisk til det, der kom fra medstuderende og ChatGPT. De studerende gav også udtryk for, at de typisk brugte underviser til at få mere konkrete svar, end de kunne få fra ChatGPT, og at det i dialogen med underviser var lettere at stille opfølgende spørgsmål. Desuden havde underviser en bedre forståelse for konteksten og kunne fx udnytte viden om, hvilke udfordringer de studerende typisk havde. Dette er eksemplificeret i følgende citat, hvor den studerende desuden også peger på, at det mundtlige vs. skriftlige medium for kommunikationen med hhv. underviser og ChatGPT spiller en rolle:

> ”Ja, vores underviser er langt nemmere at få hjælp af, fordi vi kan snakke SAMMEN om at finde frem til hvilket problem vi reelt har og hvilken løsning vi søger. Hvorimod ChatGPT er énvejs kommunikation, den forstår os ikke rigtigt, det er os som skal tune os ind på hvad den tror vi beder den om, og det tager simpelthen for lang tid. Alt er på skrift, alt skal tastes før det bliver modtaget af ChatGPT, hvorimod man langt hurtigere kan stille et verbalt spørgsmål til sin underviser eller medstuderende.”

Ud over som en sidebemærkning i dette citat blev medstuderende ikke nævnt så hyppigt i besvarelserne, men blev dog i enkelte tilfælde italesat som nogle, man kunne sparre og brainstorme med.

En fordel ved ChatGPT, som blev fremhævet af de studerende, var, at man kunne stille lige så mange ”dumme spørgsmål”, man ville, uden at skulle være nervøs for at virke uvidende. Nogle studerende nævnte desuden, at ChatGPT blev brugt i situationer, hvor de havde behov for et hurtigt svar for at komme videre, og underviser ikke umiddelbart var tilgængelig.

Rækkefølgen og tilgængeligheden var altså en faktor i, hvem de studerende spurgte om hjælp. Her nævnte flere studerende, at de først forsøgte sig med ChatGPT, og hvis den ikke kunne hjælpe, gik de til underviser. Omvendt påpegede flere studerende også, at de benyttede ChatGPT, hvis underviser ikke var tilgængelig, hvilket kunne indikere, at de prioriterede at spørge underviser før ChatGPT, hvilket også var i overensstemmelse med underviserens opfattelse.

Gennem besvarelserne ses en tydelig prioritering af at søge efter hjælp hos underviser, hvor ChatGPT primært blev brugt som en nødløsning, når underviser ikke var tilgængelig. Her italesætter de studerende, at det er dem, som skal ”tune ind” på chatbotten, hvorimod underviseren formår at tilpasse svaret til de studerende gennem samtale og dermed igennem samskabende stilladsering hjælper de studerende videre. Samtidig italesatte de studerende vigtigheden af at være kritisk over for ChatGPTs output.

Udfordringer med brugen af ChatGPT

Anvendelsen af ChatGPT har ikke været uden udfordringer for de studerende. De studerende var særligt udfordrede af, at de ikke kunne få ChatGPT til at svare på det, de spurgte om, ligesom den løsning, som ChatGPT kom med, ikke altid var korrekt eller virkede efter hensigten. Derfor endte enkelte studerende med ikke at bruge den:

> ”Det er som om den ikke helt kunne finde ud af at rette i noget kode og tilføje ting som vi ønskede det, selvom vi forsøgte at beskrive det grundigt og mere og mere grundigt for hver prompt. Dens løsningsforslag blev for komplicerede til at vi forstod dem, og da de heller ikke virkede når vi indsatte dem i vores program, opgav vi hurtigt at bruge chat GPT. Tror vi forsøgte os i 20-30 minutter uden det gav noget brugbart, derfor opgav vi det.”

Den studerende påpegede her desuden som flere andre, at ChatGPT til tider foreslog for avancerede løsninger, som de ikke forstod eller ikke var i stand til at implementere i deres projekt grundet kompleksiteten. Alt efter kompleksiteten kan studerende altså i visse tilfælde opleve, at ChatGPT giver svar, der befinder sig uden for de studerendes nærmeste udviklingszone. Endelig viste den tematiske analyse, at flere studerende var udfordrede ift. at prompte ChatGPT til at give det korrekte svar, ligesom de oplevede, at den ikke svarede på det, de efterspurgte.

Selvom ChatGPT ikke altid kom med helt korrekte løsninger, gav de studerende samtidig udtryk for, at de ofte var i stand til at tilrette output fra ChatGPT og implementere det i deres kode. I andre tilfælde oplevede de, at selvom de output, ChatGPT kom med, var forkerte, så var de alligevel i stand til at anvende dens output til at finde frem til problemet, formentlig ved hjælp af forskellige selv-stilladseringsprocesser.

De studerendes opfattelse af ChatGPT’s indvirkning på indlæring

Der var generelt blandede tilbagemeldinger blandt besvarelserne i spørgeskemaet ift., hvorvidt ChatGPT har hjulpet de studerende med deres indlæring. Det nævnes bl.a., at ChatGPT kan give et hurtigt svar, så man kan komme videre og ikke sidde fast. ChatGPT kan ifølge de studerende hjælpe med at skabe en forståelse ved at beskrive koden og komme med forslag, som de ikke selv havde tænkt på. På den måde italesættes ChatGPT som en sparringspartner, som kan give input til forklaringer eller anvendelse af begreber, de ikke kendte i forvejen. Flere studerende understreger, at deres indlæring ved brug af ChatGPT afhænger af, om de efterfølgende forstår og kan formidle det output, som værktøjet giver. Samtidig italesættes ChatGPT som et supplement, som ikke kan erstatte undervisere eller medstuderende.

> ”Det har den, ved at se hvordan koder kan løses, får man en forståelse for hvad de forskellige ting gør. Dog skal man selvfølgelig også kunne forstå og formidle det som AI skriver, ellers får man intet ud af det.”

Den studerende italesætter her en vigtig pointe, nemlig at hvis de ikke selv forholder sig til det output, som ChatGPT kommer med, så opnår de ingen læring. Når ChatGPT kommer med et output, som kan stilladsere de studerende, er det ligeledes vigtigt, at de forholder sig til det for på den måde, gennem selv-stilladsering, at tilegne sig ny viden.

Diskussion

Som det blev beskrevet i artiklens indledning, er en af de store udfordringer med programmeringsundervisning, at studerende ikke i tilstrækkelig grad får opøvet de computationelle praksisser (Lahtinen et. al., 2005), hvilket hæmmer dem i deres problemløsning. Ser man overordnet på de studerendes besvarelser om, hvad de anvendte ChatGPT til, så tegner der sig et billede af, at de studerende hovedsageligt har anvendt ChatGPT til netop at støtte dem i deres arbejde med computationelle praksisser. Selvom vi ikke har lavet en kvantitativ analyse af spørgeskemaets besvarelser, så ses det i den tematiske analyse, at langt de fleste udsagn og koder falder inden for denne kategori. Når ChatGPT bruges til at forklare computationelle begreber, ses en tendens til, at de bruger ChatGPT til at genbesøge begreberne frem for at kigge i slides eller slå dem op på en hjemmeside. Dette sker tilsyneladende med henblik på at huske begreberne, og hvordan syntaksen skal se ud. Desuden pegede spørgeskemaundersøgelsen på, at de studerende særligt anvendte ChatGPT til at understøtte deres projektarbejde. De studerende udviste generelt en skepsis over for det output, som ChatGPT kom med, og brugte ofte underviser for at få uddybende forklaringer. På denne måde blev ChatGPT italesat som et alternativ, som kunne bruges, hvis underviser ikke var tilgængelig, eller de kunne spørge underviser, hvis de ikke var tilfredse med de svar, de fik fra ChatGPT. Disse resultater indikerer et potentiale for, at ChatGPT og andre AI-baserede chatbots vil kunne fungere som hjælp til undervisere og støtte de studerende i at opnå en større forståelse for de computationelle praksisser. Selv i det omfang at ChatGPT bliver brugt til andre processer, er der et potentiale i, at noget af underviserens tid frigives til at kunne understøtte studerende med andre opgaver.

Selvom ChatGPT har et potentiale til at kunne understøtte undervisning, er det vigtigt, at teknologien bliver anvendt hensigtsmæssigt. Som det fremgår af spørgeskemaet, oplevede flere studerende, at det output, ChatGPT kom med, til tider var for kompliceret til, at de kunne bruge det. Det blev ligeledes observeret i undervisningen, at enkelte studerende indledningsvist bad ChatGPT skrive lange kodestykker, som de implementerede i deres projekt

uden at forholde sig til indholdet. Når de så efterfølgende skulle lave tilpasninger af koden, blev de så udfordrede, at de ikke kunne komme videre. Det indhold, som ChatGPT genererede, lå antageligvis for langt væk fra det, de studerende kendte til i forvejen og dermed uden for deres nærmeste udviklingszone, hvorved ChatGPT ikke har været i stand til at give den stilladsering, som skulle til for at understøtte deres forståelse. Dette er et eksempel på, hvordan ChatGPT også kan hindre læring frem for at fremme den. Et andet eksempel var, at ChatGPT indimellem genererede en forkert løsning. Selvom outputtet fra ChatGPT ikke var helt korrekt, kunne de studerende i flere tilfælde ved hjælp af selv-stilladsering eller økologisk stilladsering i form af feedback fra softwareudviklingsmiljøet bruge outputtet til at finde frem til den rette løsning. Det kunne fx være tilstrækkeligt, at kompileren gjorde opmærksom på en fejl, som de studerende så kunne rette, eller der kunne være tale om, at de ud fra erfaringer med et lignende problem gennem analog stilladsering kunne genkende den rigtige løsning. De studerende kunne ligeledes ofte omsætte outputtet fra ChatGPT til nye indsigter og forståelser, når de blev suppleret med yderligere informationer fra deres underviser. Dette er eksempler på, hvordan studerende gennem deltagelse i et dynamisk og samskabende system, der inkluderer underviser, ChatGPT, softwareudviklingsmiljø og den studerende selv, kan udvikle en større forståelse af en aktivitet og dermed blive i stand til at håndtere lignende eller beslægtede aktiviteter og underaktiviteter. Den studerende tilegner sig her viden og færdigheder fra det, der gøres med de involverede aktører, som udvikler sig over tid inden for de givne aktiviteter og konteksten (Mascolo, 2009). Et eksempel på dette er, at de studerende ikke blot passivt anvender ChatGPT på samme måde gennem hele forløbet, men at de også udvikler deres tilgang til, hvordan og hvornår de bruger ChatGPT. På den måde indgik ChatGPT overordnet som en positiv faktor i samspil med de øvrige elementer i miljøet og var på den måde med til at stilladsere de studerendes læring. En vigtig pointe er her, at det ikke siger noget om, hvorvidt ChatGPT vil kunne fungere læringsunderstøttende, hvis det stod alene, men at det har haft en positiv effekt i samspillet med andre aktører.

I spørgeskemaet kom det frem, at nogle studerende havde udfordringer med at prompte ChatGPT til at svare på det, de spurgte om. Selvom ChatGPT måske nok kan give output, som fungerer som social stilladsering, så har teknologien ikke en social bevidsthed eller en forståelse for den sociale kontekst, den indgår i, og dermed er den også afhængig af, at de studerende formår at prompte den på en måde, som gør, at den giver et stilladserende svar. ChatGPT synes at fungere bedst til at understøtte de studerendes læring, når de spørger ChatGPT med udgangspunkt i noget, de selv har lavet. Hvad enten det er at tilføje ny kode, ændre koden eller finde fejl, så vil ChatGPT tage udgangspunkt i noget, de forstår i forvejen og bidrage med nye instruktioner hertil, hvorved de studerende er bedre i stand til at kunne opnå nye indsigter igennem selv-stilladseringsprocesser.

For at ChatGPT kan have en positiv effekt på indlæring, er det altså vigtigt, at studerende bruger den hensigtsmæssigt. Det kan derfor være en god ide at afsætte tid til at gennemgå gode procedurer og måske endda opsætte regler for, hvordan ChatGPT kan bruges. Dette er i tråd med Unescos anbefalinger (Fengchun & Kelly, 2024), som fremhæver behovet for, at studerende opbygger en grundlæggende viden og literacy, der sikrer, at de kan anvende GenAI effektivt og meningsfuldt i deres uddannelse. Desuden peger samme rapport på vigtigheden i at understøtte, at de studerende udvikler en kritisk tilgang til brugen af GenAI og forstår de underliggende udfordringer forbundet med værktøjernes anvendelse (Becker et al., 2023; Hellas et al., 2023), så de kan anvende dem på en etisk forsvarlig måde (Fengchun & Kelly, 2024).

Mens tilgængeligheden af GenAI åbner nye muligheder for, *hvordan* studerende lærer, rejser der sig også spørgsmål om, *hvad* de studerende bør lære (Lu & Fletcher, 2009; Lye & Koh, 2014). Behovet for selvstændigt at kunne producere kode kan fx blive mindre essentielt, mens kompetencer som forståelse og dokumentation ser ud til at få større betydning (Becker et al., 2023). I vores studie observerede vi en udvikling i de studerendes brug af ChatGPT, hvor de gennem deres erfaringer opnåede en dybere forståelse for, hvordan de kunne anvende ChatGPT til at støtte udviklingen af deres programmeringsprojekter. Det er dog usikkert, hvordan introduktionen af ChatGPT har påvirket deres grundlæggende forståelse for programmering. Der kunne fx være kognitive opgaver, som de har udliciteret til ChatGPT og dermed ikke selv forholdt sig til. Uanset hvad peger vores studie på, at en basal forståelse for programmering stadig er nødvendig for at kunne anvende GenAI effektivt og for at kunne omsætte den feedback og de informationer, den bidrager med. Vigtigheden af at have erfaring inden for det domæne, man anvender GenAI, til er tidligere blevet pointeret i forskningslitteraturen (Sabzalieva & Valentini, 2023), da man dermed har mulighed for at verificere outputtet fra det pågældende AI-værktøj.

Med en hensigtsmæssig introduktion ses der på baggrund af vores studie potentiale for, at GenAI kan tilbyde feedback og individuel sparring til de studerende, fx i form af forklaringer af kode, eksempler på løsninger eller fejlfinding (Becker et al., 2023; Hellas et al., 2023; Lau et al., 2023). På denne måde kan GenAI være med til at aflaste underviseren og potentielt forbedre studerendes læringsudbytte, særligt i forhold til de computationelle praksisser (Butler & Morgan, 2007; Lahtinen et al, 2005). Selvom vi ikke entydigt på basis af dette studie kan pege på, at tilgængeligheden af GenAI har forbedret de studerendes læring, så kan de studerendes opfattelse af et øget læringsudbytte, og at de oplever at komme længere i deres projekter, medføre, at de ikke opfatter læringskurven i programmering som så stejl som beskrevet af

Lahtinen et al. (2005) og Butler & Morgan (2007). Dette kan potentielt øge deres motivation for at udvikle sig yderligere inden for faget.

Selvom vi overordnet kan se forskellige interaktioner og stilladseringsprocesser mellem ChatGPT, studerende, undervisere og miljøet, kan vi med det valgte undersøgelsesdesign og baseret på spørgeskemadata ikke komme tættere på de konkrete stilladseringsprocesser, som foregår i det samskabende system. Her kunne det være relevant i fremtidige studier at kigge på studerendes *prompt logs* for på den måde at undersøge de studerendes interaktion med ChatGPT med henblik på at forstå, hvordan GenAI bedst kan anvendes til at understøtte de studerendes læring og udvikling inden for programmering.

Konklusion

Ved at indgå i et samskabende system med andre aktører indikerer nærværende studie, at AI-baserede chatbots som ChatGPT kan spille en værdifuld rolle i programmeringsundervisning ved at støtte studerende i at udvikle deres computationelle praksisser og problemløsningsevner. Samtidig kan AI-baserede chatbots være med til at aflaste underviseren ved indledningsvist at svare på studerendes spørgsmål, så underviseren har mulighed for at indgå i andre læringsaktiviteter. Selvom ChatGPT kan være et nyttigt værktøj, bør der være vejledning eller undervisning i, hvordan den anvendes. Ligeledes skal man være opmærksom på, at den kan lave fejl og give misvisende feedback. Hensigtsmæssig og kritisk reflekteret brug af ChatGPT er således afgørende for at undgå misforståelser og sikre effektiv og stilladseret læring.

Referencer

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48-54. https://doi.org/10.1145/19298 87.1929905

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *In Proceedings of the 2012 annual meeting of the American educational research association*, Vancouver, Canada

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. *In Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 500-506. https://doi.org/10.1145/3545945.3569759

Butler, M., & Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. *Proceedings ascilite Singapore*, 99-107.

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

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. [White paper] http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Danmarks Statistik. (2024). Digitalisering - Temaer. https://www.dst.dk/da/Statistik/temaer/digitalisering

Daun, M., & Brings, J. (2023, June). How ChatGPT will change software engineering education. *In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education* 1, 110-116. https://doi.org/10.1145/3587102.3588815

Fincher, S. (1999). What are we doing when we teach programming? *FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education*, IEEE. DOI: 10.1109/FIE.1999.839268

Gibson, J. J. (1977). *The theory of affordances*. Hilldale, USA, 1(2), 67-82.

Gómez Puente, S. M., Van Eijck, M., & Jochems, W. (2013). A sampled literature review of design-based learning approaches: A search for key characteristics. *International Journal of Technology and Design Education*, 23, 717-732. https://doi.org/10.1007/s10798-012-9212-x

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Hellas, A., Leinonen, J., Sarsa, S., Koutcheme, C., Kujanpää, L., & Sorva, J. (2023). Exploring the responses of large language models to beginner programmers' help requests. *In Proceedings of the 2023 ACM Conference on International Computing Education Research* 1, 93-105. https://doi.org/10.1145/3568813.3600139

Hu, C. (2004). Rethinking of teaching objects-first. *Education and Information technologies*, 9, 209–218. https://doi.org/10.1023/B:EAIT.0000042040.90232.88

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *Acm sigcse bulletin*, *37*(3), 14-18. https://doi.org/10.1145/1151954.106745

Lau, S., & Guo, P. (2023, August). From" Ban it till we understand it" to" Resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. *In Proceedings of the 2023 ACM Conference on International Computing Education Research* 1, 106-121. https://doi.org/10.1145/3568813.3600138

Leinonen, J., Denny, P., MacNeil, S., Sarsa, S., Bernstein, S., Kim, J., ... & Hellas, A. (2023, June). Comparing code explanations created by students and large language models. *In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education 1,124-130.* https://doi.org/10.1145/3587102.3588785

Lim, W. M., Gunasekara, A., Pallant, J. L., Pallant, J. I., & Pechenkina, E. (2023). Generative AI and the future of education: Ragnarök or reformation? A paradoxical perspective from

management educators. *The International Journal of Management Education, 21*(2), 100790. https://doi.org/10.1016/j.ijme.2023.100790

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer science education*, 260-264. https://doi.org/10.1145/1539024.1508959

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in human behavior*, *41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Mascolo, M. F. (2005). Change processes in development: The concept of coactive scaffolding. *New Ideas in Psychology*, *23*(3), 185-196. https://doi.org/10.1016/j.newideapsych.2006.05.002

Mascolo, M. F. (2009). Beyond student-centered and teacher-centered pedagogy: Teaching and learning as guided participation. *Pedagogy and the Human Sciences*, *1*(1), 3–27.

Miao, F., & Shiohira, K. (2024). *AI competency framework for students*, UNESCO Publishing

Microsoft (2024) Makecode. Hentet fra: https://makecode.microbit.org/

Nazir, A., & Wang, Z. (2023). A comprehensive survey of ChatGPT: advancements, applications, prospects, and challenges. *Meta-radiology*, 100022. https://doi.org/10.1016/j.metrad.2023.100022

Papert, S. (1980). *Computers for children. Mindstorms: Children, computers, and powerful ideas*, 3-18.

Processing (2024). Welcome to Processing. Hentet fra: https://processing.org/

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

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education, 14*, 1-15. https://doi.org/10.1186/s41239-017-0080-z

Sabzalieva, E., & Valentini, A. (2023). *ChatGPT and artificial intelligence in higher education: Quick start guide*, UNESCO

Selby, C. & Woollard, J. (2013). *Computational Thinking: The Developing Definition*. ePrints Soton, Southampton. https://eprints.soton.ac.uk/356481

Sharples, M. (2023). Towards social generative AI for education: theory, practices and ethics. *Learning: Research and Practice, 9*(2), 159-167. https://doi.org/10.1080/23735082.2023.2261131

Schulte, C., & Bennedsen, J. (2006). What do teachers teach in introductory pro-gramming? *In Proceedings of the second international workshop on Computing education research,* 17-28. https://doi.org/10.1145/1151588.1151593

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes.* (M. Cole, V. Jolm-Steiner, S. Scribner, & E. Souberman, Eds.) Harvard University Press. https://doi.org/10.2307/j.ctvjf9vz4

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based pro-gramming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE), 18*(1), 1-25. https://doi.org/10.1145/3089799

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

Forfattere

Anders Kalsgaard Møller

Lektor, ph.d.

Aalborg Universitet

Kristine Bundgaard

Lektor, ph.d.

Aalborg Universitet