ERHVERVSAKADEMI

Programmeringsundervisning i Norden:

Et litteraturstudie af didaktiske tilgange på erhvervsakademiniveau samt praksisperspektiver

Mikael Tranekjer Debel

Erhvervsakademi Dania

Dato:
21-01-2026

Abstract

This working paper presents a scoping review of didactic approaches to programming education in Nordic higher vocational education, with a particular focus on Danish Business Academies.

Through a thematic analysis of selected studies, the review identifies five overarching didactic tendencies: (1) variation in learning objectives and didactic approaches; (2) practice-oriented and project-based teaching designs, often linked to authentic or industry-related contexts; (3) challenges related to novice learners' conceptual understanding, motivation, and progression; (4) increasing integration of computational thinking as a didactic framework; and (5) recurring challenges related to student diversity and teacher didactic confidence.

Across the Nordic region, programming education is characterised by strong traditions of collaborative learning, authenticity, and applied problem-solving. At the same time, the review highlights tensions between local didactic flexibility and the need for clearer structures to support progression and conceptual understanding. The paper concludes by discussing implications for teaching practice at the business academy level and outlining perspectives for future research in programming didactics.

Introduktion

Programmering er i de seneste år blevet et centralt element i mange videregående uddannelser, særligt inden for tekniske og erhvervsrettede uddannelser. Samtidig peger forskningen på, at programmeringsundervisning rummer en række didaktiske udfordringer, herunder store variationer i studerendes forudsætninger, vanskeligheder med abstraktion og begrebsforståelse samt udfordringer relateret til motivation og fastholdelse. Disse forhold har ført til en voksende forskningsinteresse i, hvordan programmeringsundervisning kan tilrettelægges didaktisk, så den understøtter både læring, progression og engagement.

Dette litteraturstudie fokuserer på didaktiske tilgange til programmeringsundervisning på erhvervsakademisk niveau i en nordisk kontekst. Flere studier peger på fælles tendenser i nordiske erhvervsrettede videregående uddannelser, herunder en stærk vægtning af praksisnær undervisning, professionsrettethed og anvendelsesorientering, hvorfor dette studie afgrænser sig til at undersøge didaktiske tilgange i den nordiske kontekst (Brabrand et al., 2024; Madsen et al., 2018; Svanæs, 2015).

Erhvervsakademier udgør i denne sammenhæng et særligt interessant uddannelsesniveau, idet de befinder sig i et krydsfelt mellem akademiske traditioner og professionsrettede krav. Programmeringsundervisning på dette niveau skal både understøtte udviklingen af teoretisk forståelse og praktiske kompetencer, der kan anvendes direkte i professionelle

sammenhænge. Samtidig viser eksisterende forskning, at undervisningen ofte er præget af lokale variationer i didaktisk praksis og underviseres faglige baggrunde, hvilket kan påvirke både progression og læringsudbytte (Brabrand et al., 2024; Madsen et al., 2018).

Gennem et scoping review søges der identificeret centrale mønstre, variationer og begrundelser for valg af didaktiske tilgange i forskningen i programmeringsundervisningen på erhvervsakademisk niveau i nordiske lande. Studiet har dermed til hensigt at bidrage med et samlet overblik over et forskningsfelt, der er metodisk varieret og fortsat under udvikling, samt at skabe et grundlag for videre diskussion og forskning i programmeringsdidaktik i erhvervsrettede videregående uddannelser.

Forskningsspørgsmål:

Hvad karakteriserer de didaktiske tilgange til programmeringsundervisning i nordiske uddannelser på erhvervsakademiniveau i den eksisterende forskning?

For at besvare dette overordnede spørgsmål inddrages to underspørgsmål:

1. Hvilke begrundelser gives der i studierne for valg af didaktiske tilgange?

2. Hvilke mønstre og variationer kan identificeres i de didaktiske tilgange på tværs af studierne?

Baggrund

Programmeringsundervisning udgør et centralt element i uddannelser, der sigter mod at udvikle digitale og tekniske kompetencer. Forskningen peger på, at programmering adskiller sig fra mange andre fag ved at forudsætte både abstrakt tænkning, problemløsningsstrategier og forståelse for komplekse logiske strukturer (Eckerdal & Berglund, 2005; Lahtinen et al., 2005). Disse krav betyder, at studerende ofte møder betydelige vanskeligheder i begyndende programmeringsforløb, hvilket kan påvirke motivation, progression og læringsudbytte.

I en nordisk uddannelseskontekst fremhæves programmeringsundervisning i flere studier som indlejret i uddannelser, hvor der lægges vægt på praksisnærhed, anvendelsesorientering og sammenhæng mellem uddannelse og arbejdsliv, særligt inden for erhvervsrettede og professionsorienterede uddannelser (Brabrand et al., 2024; Madsen et al., 2018; Svanæs, 2015). Dette stiller særlige krav til programmeringsundervisningen på erhvervsakademierne, hvor formålet ikke alene er at udvikle tekniske færdigheder, men også at understøtte studerendes evne til at anvende programmering i konkrete, professionsrettede problemstillinger. Studerende forventes at opnå både teoretisk forståelse og praktisk handlingskompetence, hvilket gør didaktiske valg centrale for læringsudbyttet (Studieordning for Datamatikeruddannelsen, 2025).

Samtidig viser forskningen betydelige variationer i, hvordan undervisere i Norden fortolker og implementerer programmering i undervisningen (Rolandsson, 2013; Vinnervik, 2023). Forskellene omfatter både læringsmål, undervisningsmetoder og forståelser af, hvad programmeringskompetence indebærer. I Finland har man eksempelvis arbejdet med at

integrere computational thinking på tværs af uddannelsesniveauer (Mannila et al., 2018), mens programmeringsundervisning i Danmark i højere grad beskrives som praksisnær og tværfagligt orienteret (Brabrand et al., 2024).

Denne variation betyder, at programmeringsundervisning på erhvervsakademier både rummer potentialer og udfordringer. På den ene side kan praksisorienterede undervisningsformer understøtte studerendes motivation og evne til at arbejde med autentiske problemstillinger. På den anden side peger forskningen på, at manglende fælles didaktiske rammer på tværs af uddannelser og institutioner kan vanskeliggøre sammenhængende progression og stilladsering, hvilket kan udfordre studerendes begrebsudvikling i programmering (Eckerdal, 2024; Eckerdal et al., 2005). Et studie peger på, at underviserens didaktiske sikkerhed og evne til at koble teori og praksis har stor betydning for de studerendes læring (Madsen et al., 2018).

Denne baggrund danner udgangspunkt for en systematisk kortlægning af de didaktiske tilgange, der beskrives i den nordiske forskning. Studiet bidrager med et overblik over centrale fællestræk og variationer i feltet og kan dermed belyse, hvordan forskellige didaktiske valg relaterer sig til udfordringer i programmeringsundervisning på erhvervsakademisk niveau.

Teoretisk ramme

Dette studie tager udgangspunkt i tre læringsteoretiske perspektiver, som på forskellig vis belyser de processer, der kendetegner programmeringsundervisning på erhvervsakademiniveau. Perspektiverne er valgt, fordi de giver et analytisk grundlag for at forstå, hvordan studerende udvikler faglige kompetencer i et felt, der forudsætter både abstrakt tænkning, praktisk afprøvning og deltagelse i faglige fællesskaber.

Det første perspektiv er den konstruktivistiske læringsteori, hvor viden opfattes som noget, studerende aktivt konstruerer gennem udforskning og problemløsning. I en programmeringskontekst kan læring forstås som en iterativ proces, hvor studerende eksperimenterer med kode, identificerer fejl og gradvist udvikler begrebslig forståelse. Konstruktivismen synliggør dermed betydningen af undervisningsdesign, der giver plads til afprøvning og gradvis udvikling af begreber (Piaget, 1970).

Det andet perspektiv er sociokulturelle teorier, der fremhæver læring som deltagelse i sociale og faglige praksisser (Vygotsky, 1978; Lave & Wenger, 1991). Programmering læres ikke alene som individuel aktivitet, men gennem samarbejde, dialog, feedback og brug af fælles redskaber. Dette gør perspektivet relevant i en nordisk uddannelseskontekst, hvor samarbejde, dialog og praksisfællesskaber fremhæves som centrale pædagogiske idealer (Bocconi et al., 2018; Lisborg et al., 2020).

Det tredje perspektiv er Kolbs erfaringslæringsmodel, der beskriver læring som en cyklisk proces mellem konkret handling, refleksion, begrebsliggørelse og eksperimentering ((Kolb, 1984). Programmeringsundervisning følger ofte denne cyklus, idet studerende skriver kode, analyserer resultater, abstraherer til principper og afprøver nye løsninger. Modellen giver dermed et teoretisk grundlag for at forstå, hvordan undervisningsaktiviteter kan understøtte progression og dybdelæring.

Tilsammen udgør disse tre perspektiver den teoretiske ramme, der anvendes til at fortolke og diskutere de didaktiske tilgange, som beskrives i den eksisterende forskning. Rammen understøtter studiets fokus på handling, refleksion, samarbejde og begrebsudvikling og bidrager til at belyse variationer og mønstre i programmeringsundervisning på praksisnære, professionsrettede uddannelser på erhvervsakademisk niveau i Norden.

Metode

Dette studie anvender en scoping review-tilgang til at kortlægge den eksisterende forskning om didaktiske tilgange til programmeringsundervisning i en nordisk kontekst. Scoping reviews er særligt velegnede i forskningsfelter, hvor litteraturen er metodisk varieret, fragmenteret eller hvor der mangler et samlet overblik (Arksey & O'Malley, 2005). Formålet er ikke at vurdere studiernes kvalitet, men at identificere og beskrive de didaktiske principper, variationer og udfordringer, der relaterer sig til forskningsspørgsmålet.

Søgestrategi og databaser

Litteratursøgningen blev gennemført i fem forskningsdatabaser med relevans for uddannelsesforskning og computer science education: EBSCOhost, ScienceDirect, JSTOR, Google Scholar og Det Kongelige Biblioteks forskningsportal. Kombinationen af internationale databaser og KB.dk sikrer både bred dækning og adgang til nordiske publikationer, som ofte ikke er indekseret internationalt. Søgestrategien tog udgangspunkt i begreber relateret til informatikdidaktik, programmeringsundervisning og nordiske uddannelseskontekster. Søgetermer blev tilpasset de enkelte databaser, så både engelske og nordiske termer inden for programmering og didaktik indgik, men byggede generelt på kombinationer af termer som "programming education", "didactics", "Nordic", "higher education" og "vocational".

Inklusions- og eksklusionskriterier

Studier blev inkluderet, hvis de:

1. omhandlede didaktiske tilgange, undervisningsdesign eller pædagogiske principper i programmeringsundervisning
2. var publiceret i en nordisk kontekst
3. vedrørte videregående eller beslægtede uddannelser
4. var peer-reviewed eller offentliggjort i anerkendte videnskabelige tidsskrifter eller forskningsrapporter
5. var udgivet mellem 2010 og 2025

Studier blev ekskluderet, hvis de havde rent teknisk fokus, ikke omhandlede undervisning eller lå uden for den nordiske region. Studier om begyndervanskeligheder (fx Lahtinen et al., 2005) blev inkluderet, når de havde væsentlig betydning for feltets udvikling.

Udvælgelsesproces

Søgeprocessen resulterede i 214 publikationer. Efter fjernelse af dubletter og screening af titel og abstract blev 39 studier udvalgt til fuldtekstlæsning. Heraf opfyldte 22 studier udvælgelseskriterierne og indgik i den endelige analyse. Udvælgelsen fulgte scoping review-principper (Arksey & O'Malley, 2005), hvor inklusion sker bredt og med fokus på relevans frem for metodisk ensartethed.

Analysestrategi

De udvalgte studier blev systematisk registreret i en litteraturmatrix med oplysninger om kontekst, metode, teoretisk forankring og didaktiske pointer. Den efterfølgende tematiske analyse blev gennemført som en åben og eksplorativ proces uden foruddefinerede kategorier med fokus på at identificere mønstre, variationer og udfordringer på tværs af studierne. Analysen var inspireret af scoping review-tilgangen, hvor fokus er at afdække feltets bredde og strukturere eksisterende viden uden at reducere kompleksiteten gennem for stram kategorisering.

Metodiske begrænsninger

Scoping reviews giver overblik, men indebærer begrænsninger. Studiet inkluderer ikke en kvalitetsvurdering af de enkelte artikler, og konklusionerne afspejler derfor forskningsfeltets mangfoldighed. Dertil er litteraturen begrænset af forskelle i nationale kontekster, publiceringspraksis og metodiske traditioner, hvilket påvirker sammenligneligheden på tværs af studier.

Anvendelse af generativ AI i analyseprocessen

Generativ AI (ChatGPT, version 5.2; OpenAI, 2025) er anvendt som et støtteværktøj i analyseprocessen med henblik på at skabe overblik og understøtte arbejdet med at identificere og samle temaer på tværs af de inkluderede studier. AI er anvendt i forbindelse med gennemgang af noter og sammenfatninger fra litteraturmatrixen som et eksplorativt redskab til at foreslå foreløbige tematiske sammenhænge.

De forslag, som AI genererede, er ikke anvendt direkte, men har fungeret som udgangspunkt for en efterfølgende manuel analyse. Temaerne er løbende gennemgået, justeret og valideret af forfatteren gennem gentagen læsning af de inkluderede studier og i relation til forskningsspørgsmålet. Det analytiske ansvar og den endelige tematisering påhviler forfatteren.

Analyse

Efter udvælgelsen af studierne blev der gennemført en tematisk analyse baseret på den litteraturmatrix, hvor alle studier var registreret med oplysninger om kontekst, metode, didaktiske pointer og begrundelser for valg af undervisningstilgange. Formålet med analysen var at identificere, hvad der karakteriserer de didaktiske tilgange til

programmeringsundervisning i nordiske erhvervsakademiske uddannelser, og at afdække de begrundelser, mønstre og variationer, som fremgår af forskningen.

Analysen fulgte en åben og eksplorativ tilgang inspireret af scoping review-principper. I første fase blev studierne gennemlæst med fokus på beskrivelser af undervisningsformer, læringsmål, progression, teori–praksis-integration og de begrundelser, som underviserne og forskerne angiver for deres didaktiske valg. På denne baggrund blev der gennemført en indledende kategorisering af materialet uden forhåndsdefinerede temaer for at sikre, at analysen tog udgangspunkt i studiernes egne temaer og interesser.

I anden fase blev kategorierne sammenlignet og organiseret i foreløbige tematiske grupper, som eksempelvis praksisnær undervisning, computational thinking, begrebslig forståelse, begyndervanskeligheder og underviserens rolle. I denne fase blev der lagt særlig vægt på at identificere både de didaktiske principper, der begrundes i studierne, og de variationer, der fremstår på tværs af de nordiske kontekster.

Herefter blev de tematiske grupper vurderet i lyset af studiets teoretiske ramme. De konstruktivistiske, sociokulturelle og erfaringsbaserede læringsperspektiver blev anvendt som analytiske linser til at fortolke, hvorfor bestemte didaktiske tilgange fremhæves som hensigtsmæssige eller nødvendige i programmeringsundervisning. Denne teoretiske forankring bidrog til at identificere den pædagogiske logik bag de didaktiske valg, der beskrives i studierne.

Analysen blev gennemført iterativt, hvor kategorisering og sammenligning fortsatte, indtil fem centrale didaktiske temaer stod klart. Temaerne repræsenterer både fællestræk og variation i de didaktiske tilgange på tværs af studierne og danner grundlag for resultatafsnittet. Hvert tema rummer (1) karakteristika ved den didaktiske tilgang, (2) de begrundelser, studierne fremhæver for denne tilgang, og (3) de mønstre og variationer, der udspiller sig mellem forskellige nordiske institutioner og uddannelseskontekster.

Resultater

På baggrund af den tematiske analyse blev der identificeret fem centrale temaer i den eksisterende forskning om didaktiske tilgange til programmeringsundervisning på nordiske erhvervsakademiske uddannelser. Temaerne beskriver, hvad der karakteriserer de didaktiske tilgange, hvilke begrundelser der gives for disse valg, samt hvilke mønstre og variationer der kan identificeres på tværs af studierne.

1. Variation i læringsmål og didaktiske tilgange

Et gennemgående fund i den nordiske forskning er, at programmeringsundervisning varierer betydeligt i både formål, læringsmål og didaktisk tilrettelæggelse. Studierne spænder fra undervisning med fokus på basal syntaks og grundlæggende begrebsforståelse til undervisning, der vægter problemløsning, modellering og professionsrettet anvendelse (Brabrand et al., 2024; Nygård, 2021). Denne variation hænger tæt sammen med underviseres forskellige forståelser af, hvad programmeringskompetence indebærer, og hvordan progression i undervisningen bør struktureres.

Flere studier peger på, at der ikke findes en entydig eller fælles forståelse af programmeringskompetence på tværs af uddannelser og institutioner. Rolandsson (2013) beskriver eksempelvis, at undervisere ofte "forhandler forståelsen af programmeringskompetence forskelligt afhængigt af faglig baggrund og lokale traditioner", hvilket betyder, at både læringsmål og undervisningsformer i høj grad formes lokalt. Denne pointe understøttes af nyere nordiske studier, som fremhæver, at underviseres faglige baggrunde og institutionelle rammer har stor betydning for, hvilke kompetencer der prioriteres i undervisningen (Vinnervik, 2023).

Baggrunden for denne variation knytter sig blandt andet til forskelle i uddannelsernes professionsrettede fokus, samarbejde med erhvervslivet og lokale fortolkninger af studieordninger. Brabrand et al. (2024) fremhæver i den forbindelse, at fraværet af fælles didaktiske rammer betyder, at "undervisere i praksis ofte udvikler deres egne didaktiske løsninger", hvilket kan føre til betydelige forskelle i undervisningspraksis, selv inden for samme uddannelsesniveau.

På tværs af de inkluderede studier fremstår betydelige variationer i didaktiske tilgange og forståelser af programmeringskompetence, hvilket i høj grad relaterer sig til lokale og institutionelle fortolkninger af læringsmål og professionsrettede krav. Forskningen viser, at undervisere træffer didaktiske valg på baggrund af forskellige forståelser af, hvad programmeringskompetence indebærer, herunder spændet fra syntaksbeherskelse til abstrakt problemløsning og anvendelsesorienteret problemløsning (Rolandsson, 2013, s. 9–11; Brabrand et al., 2024, s. 721–724).

Fra et didaktisk synspunkt indikerer denne variation, at programmeringsundervisning på erhvervsakademisk niveau kan mangle fælles begrebslige orienteringspunkter på tværs af uddannelser og undervisningsforløb, hvilket kan vanskeliggøre arbejdet med tydelig progression og sammenhæng for de studerende. Brabrand et al. (2024, s. 725–727) viser, at programmering indgår med meget forskellige roller og kompetencefokus på tværs af uddannelser, hvilket kan medføre uensartede læringsforløb og progression.

Samtidig peger nordiske curriculumstudier på, at variationen i didaktiske rammer i høj grad er knyttet til nationale, institutionelle og faglige kontekster snarere end til fælles nordiske standarder (Pajchel et al., 2024, s. 17–18). På baggrund af denne litteratur peger nærværende studie således på et behov for mere eksplicitte og systematiske didaktiske modeller i lokale undervisningskontekster, frem for etableringen af én fælles didaktisk referenceramme på tværs af hele den nordiske erhvervsakademiske sektor.

2. Praksisnær og anvendelsesorienteret undervisning

Et markant tema i den nordiske forskning er vægtningen af praksisnære og anvendelsesorienterede undervisningsformer i programmeringsundervisningen. Projektarbejde, casebaserede aktiviteter og iterative arbejdsprocesser fremhæves i flere studier som centrale didaktiske greb, særligt i erhvervsakademiske uddannelser, hvor undervisningen er tæt koblet til professionel praksis (Svanæs, 2015; Brabrand et al., 2024; Madsen et al., 2018). Hands-on aktiviteter, hvor studerende arbejder med autentiske problemstillinger, beskrives som både motiverende og læringsunderstøttende.

Begrundelserne for at anvende praksisnære undervisningsformer knytter sig blandt andet til ønsket om at styrke studerendes motivation, engagement og professionsidentitet. Svanæs (2015) fremhæver, at maker- og projektbaserede læringsformer skaber "et højt niveau af engagement, fordi studerende arbejder med konkrete artefakter og synlige resultater", hvilket bidrager til en oplevelse af mening og ejerskab i læringsprocessen. Tilsvarende peger flere studier på, at praksisnærhed gør det lettere for studerende at se relevansen af programmering i forhold til fremtidige arbejdsopgaver.

Flere studier peger på, at undervisning på erhvervsakademisk niveau har særligt gode betingelser, når den tager udgangspunkt i virkelighedsnære problemstillinger og konkrete anvendelsesscenarier. Koblingen mellem undervisning og praksis fremhæves som central for at understøtte både faglig forståelse og overførsel af kompetencer til professionelle kontekster, særligt i praksis- og professionsrettede uddannelser (Svanæs, 2015; Brabrand et al., 2024; Madsen et al., 2018).

Samtidig viser forskningen variationer i, hvordan praksisnærhed forstås og implementeres, idet nogle studier relaterer praksis primært til teknisk kodeproduktion, mens andre lægger vægt på professionspraksis gennem samarbejdsbaserede arbejdsformer, peer feedback og projektorienteret arbejde med større frihed for de studerende (Svanæs, 2015; Brabrand et al., 2024).

3. Integration af teori og praksis

To studier peger på, at studerende ofte har vanskeligt ved at forbinde teoretiske begreber med praktisk programmeringsarbejde, og forskningen beskriver et vedvarende spændingsfelt mellem at forstå programmeringskoncepter i abstrakt form og at kunne anvende dem i konkrete opgaver (Eckerdal & Berglund, 2005; Eckerdal, 2024).

Begrundelsen for at arbejde systematisk med integration af teori og praksis knytter sig til ønsket om at understøtte begrebslig forståelse og progression. Eckerdal & Berglund (2005) viser, at studerende ofte "forstår begreber i isolation, men har vanskeligt ved at anvende dem i praksis", hvilket kan føre til usikkerhed og stagnation i læringsprocessen. Senere forskning understreger, at progression i programmering forudsætter kontinuerlige skift mellem teoretisk forståelse og praktisk afprøvning. Eckerdal (2024) fremhæver i den forbindelse, at "progression i programmering kræver gentagne bevægelser mellem teori og praksis, snarere end en adskillelse af de to".

To studier peger på cykliske undervisningsforløb som en måde at imødekomme udfordringen med at forbinde teori og praksis, hvor studerende veksler mellem kodning, refleksion, begrebsliggørelse og ny afprøvning. Eckerdal og Berglund (2005, pp. 108–110) viser, at studerende kan arbejde med konkrete programmeringsopgaver uden at udvikle en tilsvarende begrebslig forståelse, hvis ikke undervisningen understøtter gentagne bevægelser mellem handling og refleksion.

Variationerne i forskningen relaterer sig blandt andet til, hvornår teori introduceres i undervisningsforløbet, hvor eksplicit koblingen mellem teori og praksis gøres, og hvilken rolle refleksion spiller. Nogle undervisningsdesigns tager udgangspunkt i teoretiske begreber, mens andre starter i praksis og bygger teorien op på baggrund af studerendes erfaringer. Denne variation kan forstås i lyset af erfaringsbaserede læringsmodeller, hvor

læring anskues som en cyklisk proces mellem konkret erfaring, refleksion, begrebsliggørelse og eksperimentering (Kolb, 1984, pp. 38–41).

4. Computational Thinking som didaktisk ramme

Computational Thinking optræder i to nordiske studier som en central didaktisk ramme for programmeringsundervisning. CT anvendes til at strukturere læringsmål, undervisningsindhold og progression og forstås som et sæt af kognitive processer, herunder algoritmisk tænkning, dekomposition, mønstergenkendelse og abstraktion (Ejsing-Duun et al., 2021; Sundtjønn et al., 2024).

Begrundelsen for at anvende CT som didaktisk ramme knytter sig til, at den giver et fælles begrebsapparat for programmeringskompetence og tydeliggør, hvilke kognitive processer undervisningen sigter mod at udvikle. Sundtjønn et al. (2024) beskriver CT som "et didaktisk metasprog, der gør det muligt for undervisere at analysere og strukturere progression i programmeringsundervisningen". Tilsvarende fremhæver Ejsing-Duun et al. (2021), at CT kan fungere som "a way of structuring learning activities that supports abstraction and problem decomposition", hvilket gør rammen anvendelig på tværs af både begyndende og mere avancerede forløb.

Samtidig viser forskningen variationer i, hvordan Computational Thinking implementeres i praksis. I nogle studier fungerer CT som et eksplicit læringsmål, hvor studerende forventes at udvikle specifikke CT-kompetencer, mens det i andre anvendes som et organiserende princip for undervisningsdesign eller som et analytisk redskab for undervisere (Ejsing-Duun et al., 2021; Sundtjønn et al., 2024). CT kobles desuden i forskellig grad til projektarbejde, maker-aktiviteter og problembaseret læring, hvilket afspejler forskellige didaktiske traditioner og institutionelle kontekster (Bocconi et al., 2018; Svanæs, 2015).

5. Udfordringer i programmeringsundervisningen

Et tværgående tema i forskningen er de udfordringer, som både studerende og undervisere oplever i programmeringsundervisningen. Forskningen peger på begyndervanskeligheder relateret til abstraktion, kontrolstrukturer, variabler og fejlfinding som særligt fremtrædende udfordringer (Lahtinen et al., 2005). Disse vanskeligheder kan hæmme både motivation og progression, især blandt studerende uden forudgående programmeringserfaring.

Derudover fremhæves store forskelle i studerendes forudsætninger som en central udfordring for undervisere. Fagerlund et al. (2022) viser, at studerendes oplevelse af egen kompetence har stor betydning for deres engagement og læringsudbytte, mens Tellhed et al. (2023) peger på, at lav selvtillid kan føre til frafald eller passiv deltagelse. Samtidig beskriver Brabrand et al. (2024), at undervisere ofte oplever betydelig usikkerhed i valget af didaktiske strategier, særligt når deres egen baggrund ikke er informatikfaglig. De fremhæver, at "undervisere i sådanne situationer kan mangle didaktiske redskaber til at håndtere både faglig variation og progression".

Mange af de didaktiske tilgange, som identificeres i forskningen, begrundes netop som svar på disse udfordringer. Praksisnære undervisningsformer anvendes for at øge motivation og relevans, integration af teori og praksis anvendes for at styrke begrebslig forståelse, og

Computational Thinking anvendes for at skabe struktur og gennemsigtighed i læringsmål og progression. Variationerne i de temaer forskningen undersøger, relaterer sig til hvilke udfordringer der prioriteres at forstå, hvilke didaktiske løsninger der foreslås, og hvordan underviserrollen forstås, enten som facilitator, ekspert eller coach.

Opsummering af fund

Samlet set viser resultaterne, at programmeringsundervisning på erhvervsakademisk niveau i Norden er kendetegnet ved betydelig variation i både læringsmål og didaktiske tilgange. Forskningen peger samtidig på en gennemgående vægtning af praksisnære og anvendelsesorienterede undervisningsformer, som begrundes med hensyn til motivation, professionsrettethed og læringsudbytte. Et centralt fund er desuden, at integrationen af teori og praksis fremstår som en vedvarende didaktisk udfordring, hvor flere studier fremhæver behovet for cykliske undervisningsformer, der understøtter begrebslig forståelse og progression. Computational Thinking fremstår i denne sammenhæng som en udbredt, men forskelligt anvendt didaktisk ramme, der anvendes til at strukturere både læringsmål, undervisningsdesign og progression. På tværs af studierne peger forskningen desuden på en række gennemgående udfordringer relateret til begyndervanskeligheder, variation i studerendes forudsætninger og underviseres didaktiske sikkerhed. Tilsammen indikerer fundene et behov for tydeligere didaktiske rammer og mere systematisk understøttelse af progression i programmeringsundervisningen.

Diskussion

Dette litteraturstudie viser, at programmeringsdidaktik på nordiske erhvervsakademier er kendetegnet ved betydelig variation i undervisningsdesign, didaktiske tilgange og læringsmål. I det følgende diskuteres de fem identificerede hovedtemaer i relation til studiets teoretiske ramme samt den nordiske erhvervsakademiske uddannelseskontekst.

Variation i didaktiske tilgange og læringsmål

Den store variation i didaktiske praksisser kan forstås gennem et konstruktivistisk perspektiv, hvor undervisere udformer deres undervisning ud fra lokale erfaringer og faglige forståelser. Forskningen peger på, at undervisere i nordiske uddannelseskontekster fortolker og forhandler forståelsen af programmeringskompetence forskelligt afhængigt af faglig baggrund og institutionelle rammer (Rolandsson, 2013, s. 214–216; Vinnervik, 2023, s. 89–91).

Fra et didaktisk synspunkt indikerer denne variation, at programmeringsundervisning på erhvervsakademisk niveau kan mangle fælles begrebslige orienteringspunkter på tværs af uddannelser og undervisningsforløb, hvilket kan vanskeliggøre arbejdet med tydelig progression og sammenhæng for de studerende. Brabrand et al. (2024, s. 721–724) viser eksempelvis, at underviseres didaktiske valg spænder fra fokus på syntaksbeherskelse til vægtning af abstrakt problemløsning og design, hvilket kan resultere i betydelige forskelle i læringsmål og undervisningspraksis.

Der er både styrker og udfordringer forbundet med denne variation i didaktiske praksisser. På den ene side muliggør fleksibiliteten en tilpasning af undervisningen til lokale behov og samarbejde med erhvervslivet, hvilket harmonerer med erhvervsakademiets praksisnære profil. På den anden side peger flere studier på, at fraværet af fælles didaktiske referencerammer på forløbs- og institutionsniveau kan udfordre sammenhængen i undervisningen og gøre progressionen mindre tydelig for de studerende (Rolandsson, 2013, s. 218–219; Brabrand et al., 2024, s. 725–726).

På baggrund af denne litteratur peger nærværende studie således på et behov for mere eksplicitte og systematiske didaktiske modeller i lokale undervisningskontekster, snarere end på tværs af hele den nordiske erhvervsakademiske sektor.

Praksisnær undervisning som bærende didaktisk princip

Den stærke vægt på praksisnære undervisningsformer, som projektarbejde og learning-by-doing, kan forstås gennem Kolbs erfaringslæringsteori (Kolb, 1984). Den iterative proces mellem handling, refleksion og abstraktion stemmer godt overens med den måde, hvorpå studerende tilegner sig programmering. Forskningen viser, at studerende ofte oplever øget motivation og læringsudbytte, når de arbejder med autentiske problemstillinger og udvikler konkrete produkter som en del af undervisningen (Svanæs, 2015; Brabrand et al., 2024). Dette understøtter erhvervsakademiets mål om professionsrettet læring, hvor undervisningen tilrettelægges med henblik på anvendelse i professionelle og erhvervsmæssige sammenhænge.

Samtidig peger studier på, at praksisnær undervisning kan udfordre studerende med svagere forudsætninger, hvis arbejdet med konkrete programmeringsopgaver ikke ledsages af eksplicit didaktisk stilladsering og begrebslig støtte. Eckerdal og Berglund (2005, s. 108–110) viser eksempelvis, at studerende kan løse programmeringsopgaver på et operationelt niveau uden at udvikle en tilsvarende begrebslig forståelse, hvilket kan hæmme læring og progression. Tilsvarende peger Lahtinen et al. (2005, s. 1–2; 4–5) på, at begyndere har vedvarende vanskeligheder, hvis undervisningen ikke understøtter systematisk opbygning af forståelse.

Forskningen peger på, at variation i studerendes forudsætninger stiller særlige krav til undervisningens didaktiske tilrettelæggelse. Flere studier fremhæver, at undervisere må balancere mellem åbne, eksperimenterende arbejdsformer og mere strukturerede, begrebsorienterede aktiviteter for at understøtte både engagement og faglig progression (Eckerdal & Berglund, 2005; Brabrand et al., 2024). Dette peger i litteraturen på et behov for blandede didaktiske tilgange, hvor praksisnære og kreative elementer kombineres med eksplicit begrebsliggørelse og stilladsering.

Set fra et underviserpraktisk perspektiv vurderer forfatteren, at denne balance er særlig central i programmeringsundervisning på erhvervsakademisk niveau, hvor spredningen i studerendes forudsætninger ofte er stor. I denne kontekst kan for åbne arbejdsformer uden tydelig struktur skabe usikkerhed hos nogle studerende, mens for stærkt styrede forløb kan begrænse engagement og oplevet relevans hos andre. På den baggrund fremstår didaktiske tilgange, der bevidst veksler mellem struktur og åbenhed, som særligt velegnede til at understøtte både progression og motivation.

Integration af teori og praksis som didaktisk udfordring

Flere af de udvalgte studier peger på, at studerende ofte har vanskeligt ved at forbinde teoretiske begreber med konkret programmeringspraksis. Eckerdal og Berglund (2005, s. 108–110) viser eksempelvis, at studerende kan arbejde med programmeringsopgaver på et operationelt niveau uden at udvikle en tilsvarende begrebslig forståelse, mens Lahtinen et al. (2005, s. 1–2; 4–5) peger på, at disse vanskeligheder er udbredte blandt begyndere og kan hæmme progression. Set i forlængelse heraf kan disse udfordringer forstås i lyset af sociokulturelle perspektiver på læring, hvor begrebslig forståelse udvikles gennem deltagelse i meningsfulde praksisser.

Undervisning, der adskiller teori og praksis, risikerer således at skabe en kløft mellem det, studerende kan gøre, og det, de kan forklare. Eksempelvis viser Eckerdal, Berglund og Thuné (2024, s. 331–332), at studerende kan skrive fungerende kode uden nødvendigvis at have udviklet en dyb forståelse af de underliggende begreber.

Som en didaktisk konsekvens heraf fremstår et fokus på cykliske læringsprocesser, hvor studerende bevæger sig fra konkret erfaring over refleksion til begrebsliggørelse, som afgørende for at understøtte sammenhængen mellem teori og praksis. Undervisningsdesigns, der eksplicit kobler praktisk programmeringsarbejde med refleksionsøvelser, kontekstualisering og modellering, kan bidrage til at gøre begreber og principper mere tilgængelige for de studerende.

De didaktiske implikationer heraf er, at programmeringsundervisning i højere grad må planlægges med henblik på at skabe rum for fælles refleksion, begrebsafklaring og systematisk bearbejdning af erfaringer fra praksis. Set fra et underviserpraktisk perspektiv viser erfaringer fra klasserummet, at studerende ofte har svært ved at gøre de teoretiske pointer eksplicitte i deres praktiske arbejde, medmindre disse aktivt gøres til genstand for undervisning. Uden sådanne didaktiske greb risikerer praksisnære forløb at fastholde studerende i en instrumentel tilgang til programmering, hvor fokus er på at få koden til at virke frem for at opnå en dybere forståelse af de underliggende principper.

På den baggrund fremstår cykliske undervisningsdesigns, hvor handling, refleksion og begrebsliggørelse gentagne gange kobles, som centrale for at understøtte både faglig progression og studerendes evne til at anvende programmeringsviden i nye og mere komplekse sammenhænge.

Computational Thinking som didaktisk ramme

Brugen af computational thinking (CT) i de udvalgte studier viser, at CT fungerer som et didaktisk metasprog, der kan tydeliggøre læringsmål, kompetencer og progression. CT-baserede rammer tilbyder strukturer, som undervisere kan bruge til at planlægge undervisningsforløb, der går ud over syntaks og tekniske færdigheder.

Fra et didaktisk perspektiv hjælper CT med at konkretisere, *hvad* studerende skal lære (fx decomposition, abstraktion, algoritmisk tænkning), og *hvordan* disse kompetencer kan udvikles gennem undervisningsdesign. Dette er særligt nyttigt i uddannelseskontekster lig

den erhvervsakademiske, hvor undervisningen skal være både praktisk relevant og teoretisk forankret.

Dog viser studierne også, at Computational Thinking ikke altid implementeres systematisk, og at undervisere kan være usikre på, hvordan CT omsættes til konkrete undervisningsaktiviteter i praksis. Dette understreger behovet for didaktisk støtte og efteruddannelse.

Set fra et underviserpraktisk perspektiv vurderer forfatteren, at Computational Thinking rummer et betydeligt didaktisk potentiale i programmeringsundervisning på erhvervsakademisk niveau, særligt som et fælles begrebsligt sprog for problemløsning, strukturering og refleksion. Samtidig peger erfaringer fra undervisningspraksis på, at CT sjældent fungerer hensigtsmæssigt som et selvstændigt læringsmål løsrevet fra konkret programmeringsarbejde. I stedet fremstår CT mest meningsfuldt, når det integreres implicit i arbejdet med programmeringsopgaver, hvor begreber som dekomposition, abstraktion og algoritmisk tænkning gøres eksplicitte i forbindelse med løsning og refleksion over konkrete problemer.

Udfordringer i programmeringsundervisningen

De tilbagevendende udfordringer, som identificeres i litteraturen – herunder begyndervanskeligheder, manglende progression, lav motivation og underviserusikkerhed – kan samlet set forstås som udtryk for et strukturelt didaktisk problem i programmeringsundervisningen. Set i lyset af konstruktivistiske og sociokulturelle læringsperspektiver peger disse udfordringer på, at studerende har behov for tydelige læringsmål, gradvis progression, stilladserede læringsaktiviteter samt mulighed for social læring og feedback.

Sammenhængen mellem disse faktorer indikerer, at udfordringerne ikke alene kan tilskrives individuelle studerendes forudsætninger, men i høj grad relaterer sig til undervisningens didaktiske organisering. Når læringsmål er uklare, progressionen utydelig og stilladsering begrænset, øges risikoen for, at studerende mister motivation og oplever vanskeligheder ved at udvikle en stabil faglig selvforståelse. Særligt problemstillinger omkring motivation og selvopfattelse (Tellhed et al., 2023; Fagerlund et al., 2022) understøtter betydningen af læringsmiljøer, hvor studerende oplever mestring, kompetenceudvikling og sammenhæng i undervisningen.

I en dansk erhvervsakademisk kontekst kan disse udfordringer være særligt fremtrædende, idet uddannelserne typisk rekrutterer studerende med meget forskellige uddannelsesmæssige og faglige forudsætninger. Dette stiller øgede krav til undervisningens didaktiske struktur og tydelighed, sammenlignet med mere homogent sammensatte universitetsuddannelser. Set i dette lys fremstår behovet for eksplicitte didaktiske rammer ikke som et spørgsmål om standardisering, men som et middel til at understøtte progression, motivation og faglig sammenhæng for en bred studentergruppe.

Sammenfatning af diskussionen

Sammenfattende viser litteraturen, at programmeringsdidaktik i Norden er kendetegnet ved:

- stor variation i undervisningspraksis
- stærk fokus på praksis og autentiske opgaver
- udfordringer i integration af teori og praksis
- begyndende brug af CT som didaktisk ramme
- gennemgående behov for didaktisk struktur og progression

Disse resultater peger samlet på et felt i udvikling, hvor der er behov for stærkere teoretisk forankrede didaktiske modeller og mere systematisk kompetenceudvikling for undervisere.

Begrænsninger og perspektiver for videre forskning

Dette litteraturstudie har bidraget med et samlet overblik over didaktiske tilgange og udfordringer i programmeringsundervisning på erhvervsakademisk niveau i en nordisk kontekst. Studiet kan desuden bruges som afsæt for at identificere videnshuller, og særligt temaet om udfordringer i programmeringsundervisningen indikerer et behov for at undersøge, hvordan nye didaktiske greb kan understøtte studerendes læring, begrebsforståelse og progression i praksis. På baggrund af de identificerede mønstre og udfordringer peger nærværende litteraturstudie på et behov for empiriske studier, der afprøver praksisnære didaktiske greb i programmeringsundervisningen, særligt med fokus på begrebsforståelse, progression og læring i autentiske undervisningskontekster. Et eksempel på et sådant igangværende forskningsarbejde er et projekt om reality testing som didaktisk værktøj i programmeringsundervisningen (Hansen, kommende). En parallel afprøvning af sådanne tilgange i forskellige erhvervsakademiske kontekster vil kunne bidrage med viden om læringseffekter, didaktisk anvendelighed og overførbarhed på tværs af institutioner.

En sådan empirisk undersøgelse kan ses som et muligt perspektiv for videre forskning, der kan uddybe og nuancere de didaktiske mønstre og udfordringer, som er identificeret i dette litteraturstudie. Et muligt afledt fokus kan her være studerendes sproglige og begrebslige forståelse i programmeringsundervisningen, herunder hvordan programmeringssprogets terminologi og faglige begreber forstås, anvendes og udvikles i læringsprocessen.

Konklusion

Formålet med dette studie har været at undersøge, hvad der karakteriserer didaktiske tilgange til programmeringsundervisning på erhvervsakademisk niveau i Norden i den eksisterende forskning. Studiet viser, at programmeringsundervisningen er kendetegnet ved betydelig variation i læringsmål og didaktisk tilrettelæggelse, men også ved nogle gennemgående træk. Særligt praksisnære og anvendelsesorienterede undervisningsformer fremstår som centrale, ligesom integrationen af teori og praksis og anvendelsen af Computational Thinking som didaktisk ramme spiller en væsentlig rolle i mange studier.

I relation til underspørgsmålet om begrundelser for valg af didaktiske tilgange peger forskningen på flere gennemgående rationaler. Didaktiske valg begrundes blandt andet med et ønske om at øge studerendes motivation og engagement, styrke professionsrettethed og sikre progression i læringsforløb. Derudover fremhæves behovet for at understøtte begrebslig forståelse og håndtere begyndervanskeligheder som centrale begrundelser for

anvendelsen af praksisnære aktiviteter, cykliske læringsforløb og strukturerende didaktiske rammer.

Med hensyn til mønstre og variationer viser studiet, at der på tværs af nordiske uddannelseskontekster eksisterer betydelige forskelle i, hvordan programmeringskompetence forstås, hvordan progression planlægges, og hvordan didaktiske tilgange implementeres. Samtidig fremtræder der fælles tendenser, herunder en gennemgående vægtning af praksisnær undervisning og et udbredt fokus på at skabe sammenhæng mellem teori og praksis. Variationerne peger på et felt, hvor didaktiske valg i høj grad formes af lokale kontekster, underviserkompetencer og institutionelle rammer.

Afslutningsvis peger studiets samlede fund på et behov for tydeligere didaktiske referencerammer på forløbs- og institutionsniveau samt mere systematisk understøttelse af progression i programmeringsundervisningen på erhvervsakademisk niveau. Et styrket fokus på didaktiske begrundelser og fælles begrebsrammer kan bidrage til at reducere ubegrundet variation og understøtte kvalitet og sammenhæng i undervisningen. Studiet kan dermed danne grundlag for videre forskning og didaktisk udviklingsarbejde inden for programmeringsundervisning på erhvervsakademisk niveau i en nordisk kontekst.

Anvendelse af AI

Generativ AI (ChatGPT, version 5.2) er anvendt som støtteværktøj i forbindelse med sproglig bearbejdning, strukturering af teksten samt som analytisk støtte i tematiseringsprocessen. Anvendelsen af AI i analyseprocessen er yderligere beskrevet i afsnittet “Metode”. AI har ikke bidraget med selvstændige analytiske konklusioner eller fortolkninger. Det faglige ansvar for analyse, fortolkning og indhold påhviler forfatteren.

Referencer

Arksey, H., & O'Malley, L. (2005). *Scoping studies: Towards a methodological framework*. International Journal of Social Research Methodology, 8(1), 19–32. https://doi.org/10.1080/1364557032000119616

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The Nordic approach to introducing computational thinking and programming in compulsory education.* In S. Bocconi, A. Chioccariello, & J. Earp (Eds.), *Computational thinking in compulsory education* (pp. 39–64). European Commission. https://publications.jrc.ec.europa.eu/repository/handle/JRC110359

Brabrand, C., Nicolajsen, S. M., Nielsen, S., & Carlsen, L. M. (2024). Programming education across disciplines: A nationwide study of Danish higher education. *Higher Education, 90*, 711–737. https://doi.org/10.1007/s10734-024-01345-4

Buitrago-Flórez, F., Mifsud, L., Rehder, M. M., & Frågåt, T. (2020). Designing a socio-cultural approach for teaching and learning computational thinking. *Nordic Journal of Digital Literacy, 15*(2), 106–124.

Erhvervsakademi Dania. (2025). Studieordning for Datamatiker AK. https://eadania.dk/media/6388/studieordning-datamatiker092025.pdf

Eckerdal, A. (2024). Learning programming practice and programming theory in undergraduate computing education. *European Journal of Engineering Education, 49*(1), 115–134. https://doi.org/10.1080/03043797.2023.2294953

Eckerdal, A., & Berglund, A. (2005). What does it take to learn 'programming thinking'? In Proceedings of the First International Workshop on Computing Education Research (ICER '05) (pp. 135–142). Association for Computing Machinery. https://doi.org/10.1145/1089786.1089799

Ejsing-Duun, S., Misfeldt, M., & Andersen, R. (2021). Computational thinking karakteriseret som et sæt af kompetencer. *Learning Tech, 10*(2). https://tidsskrift.dk/learningtech/article/view/125258/175552

Elicer, R., Tamborg, A. L., Bråting, K., & Kilhamn, C. (2023). Comparing the integration of programming and computational thinking into Danish and Swedish elementary mathematics curriculum resources. *LUMAT, 11*(3), 77–102. https://doi.org/10.31129/LUMAT.11.3.1940

Elicer, R., Fondo, L., Bråting, K., Kilhamn, C., & Tamborg, A. L. (2021). Designing informatics curriculum for K–12 education: From concepts to implementations. *Informatics in Education, 20*(3), 333–360. https://doi.org/10.15388/infedu.2021.22

Fagerlund, J., Leino, K., Kiuru, N., & Niilo-Rämä, M. (2022). Finnish teachers' and students' programming motivation and their role in teaching and learning computational thinking. Frontiers in Education, 7, Article 948783. https://doi.org/10.3389/feduc.2022.948783

Hansen, J. W. (kommende). *Reality testing som værktøj til læring i programmeringsundervisningen*. EA-viden. https://www.eaviden.dk/project/reality-testing-som-vaerktoej-til-laering-i-programmeringsundervisning/

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development.* Prentice Hall.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin, 37*(3), 14–18. https://doi.org/10.1145/1151954.1067453

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation.* Cambridge University Press.

Lisborg, A., Pettersson, F., Guðmundsdóttir, G. B., & Mifsud, L. (2020). Digital competences in Nordic teacher education. *Nordic Journal of Comparative and International Education, 4*(4), 41–59. https://journals.oslomet.no/index.php/nordiccie/article/view/4295/4152

Madsen, S. S., Thorvaldsen, S., & Archard, S. (2018). Students' experiences with learning programming in higher education. *Nordic Journal of Digital Literacy, 13*(1), 5–21. https://doi.org/10.18261/ISSN.1891-943X-2018-01-02

Mannila, L., Peltomäki, M., & Salakoski, T. (2018). Computing education in Finland: Curriculum and implementation. *Informatics in Education, 17*(3), 357–375. https://doi.org/10.15388/infedu.2018.18

OpenAI. (2025). *ChatGPT (version 5.2)* [Large language model]. https://chat.openai.com/

Pajchel, K., Mifsud, L., Frågåt, T., Rehder, M. M., & Juuti, K. (2024). Sign of the times: The framing of computational thinking in Danish, Finnish, and Norwegian curricula. *Nordic Journal of Comparative and International Education, 8*(4). https://doi.org/10.7577/njcie.5744

Piaget, J. (1970). *Science of education and the psychology of the child.* Viking Press.

Rehder, M. M., Juuti, K., Pajchel, K., Frågåt, T., & Mifsud, L. (2024). Computational thinking in Nordic teacher education and schools. *Nordic Journal of Comparative and International Education, 8*(4). https://doi.org/10.7577/njcie.6120

Refvik, K. A. S., & Bjerke, A. H. (2024). Computational thinking as a tool in primary and secondary mathematical problem solving: A literature review. *NOMAD: Nordic Studies in Mathematics Education, 27*(3). https://doi.org/10.7146/nomad.v27i3.149191

Rolandsson, L. (2013). Changing computer programming education: The dinosaur that survived in school: An explorative study about educational issues based on teachers' beliefs and curriculum development in secondary school. In Proceedings of the 2013 International Conference on Teaching, Assessment and Learning for Engineering (TALE) (pp. 220–223). IEEE. https://doi.org/10.1109/LaTiCE.2013.47

Sundtjønn, T., Aalbergsjø, S. G., Møller, T. E., & Schrøder, V. (2024). Review on pedagogical practices for computational thinking in teacher education: Characterizing an

emerging field. Nordic Journal of Comparative and International Education, 8(4). https://doi.org/10.7577/njcie.5742

Svanæs, D. (2015). A maker approach to computer science education: Lessons learned from a first-year university course. In Proceedings of the Workshop of Making as a Pathway to Foster Joyful Engagement and Creativity in Learning (Make2Learn 2015) (CEUR-WS Vol. 1450, pp. 15–20). CEUR-WS. https://ceur-ws.org/Vol-1450/Make2Learn2015_proceedings.pdf

Tellhed, U., Björklund, F., Kallio Strand, K., & Schöttelndreier, K. (2023). “Programming Is Not That Hard!” When a Science Center Visit Increases Young Women’s Programming Ability Beliefs. Journal for STEM Education Research, 6, 252–274. https://doi.org/10.1007/s41979-023-00094-w

Vinnervik, P. (2023). An in-depth analysis of programming in the Swedish school curriculum: Rationale, knowledge content and teacher guidance. *Journal of Computers in Education, 10*(1), 237–271. https://doi.org/10.1007/s40692-022-00230-2

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