10 **Learning Tech**
Tidsskrift for læremidler, didaktik og teknologi

LÆRE MIDDEL DK

Computational thinking karakteriseret som et sæt af kompetencer

En begrebskortlægning

Af Stine Ejsing-Duun, Morten Misfeldt & Daniel Gorm Andersen

Korrekt citering af denne artikel efter APA-systemet (American Psychological Association System, 7th Edition):
Ejsing-Duun, S., Misfeldt, M. & Andersen, D. G. (2021). Computational thinking karakteriseret som et sæt af kompetencer. En begrebskortlægning. *Learning Tech – Tidsskrift for læremidler, didaktik og teknologi*, (10), 405-429.
DOI 10.7146/lt.v6i10.125258

Abstract

Når Teknologiforståelse rammer klasseværelset, står lærere og elever overfor et nyt fag, de endnu ikke er bekendte med. Faget er defineret igennem læreplan og fælles mål blandt andet gennem kompetenceområdet computationel tankegang[1]. I denne artikel sætter vi fokus på, hvilke kompetencer der knytter sig til computational thinking ved at undersøge, hvordan de er beskrevet og defineret i tidligere forskning. Formålet er igennem denne begrebskortlægning at skabe et mere nuanceret sprog for indholdet af kompetenceområdet. Artiklen bygger på en litteraturgennemgang af international forskningslitteratur, der omtaler "computational thinking" og kompetencer fra 2013, hvor en større litteraturgennemgang blev fortaget, og frem til 2018, hvor forsøget om Teknologiforståelse blev udrullet i Danmark, samtidigt med at en lang række andre lande påbegyndte egentlig implementering af programmering og computational thinking i grundskolen.

When "Technology Comprehension" hits the classroom, teachers and students face a subject they are not yet familiar with. The subject is defined through curriculum and common goals. Many words and working methods are new. In this article, we focus on competencies related to computational thinking and how they have been defined in research. This is done to create a more nuanced language and understanding of the content of the area of competence. The article is based on a literature review of international research literature that discusses and maps "computational thinking" and competencies from Grover and Peas literature review in 2013 until 2018, when "Technology Comprehension" was rolled out.

1 I faget Teknologiforståelse bruges begrebet "computationel tankegang", der er en direkte oversættelse af det engelske begreb "computational thinking". Vi vælger at bruge det danske begreb, når vi omtaler fagets definitioner, og ellers det engelske begreb computational thinking, da det er lettere at udtale.

Computational thinking karakteriseret som et sæt af kompetencer

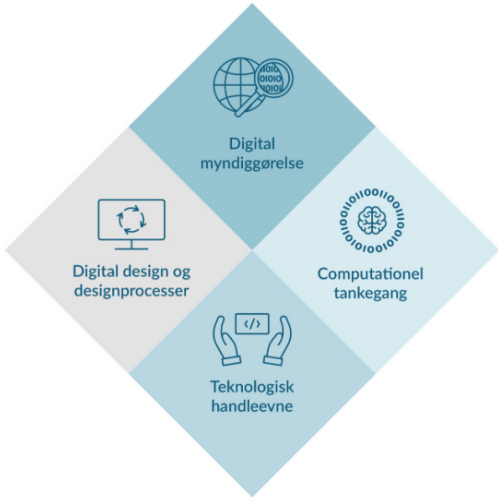
En begrebskortlægning

Af Stine Ejsing-Duun, Aalborg Universitet, Morten Misfeldt, Københavns Universitet, & Daniel Gorm Andersen, Aalborg Universitet/Clio

Teknologiforståelse: kompetenceområder

I foråret 2019 satte 46 danske folkeskoler gang i undervisning i forsøgsfaget Teknologiforståelse. Et fag, der rummer fire kompetenceområder (Undervisningsministeriet, 2018): digital myndiggørelse, digital design og designprocesser, teknologisk handleevne samt computationel tankegang.

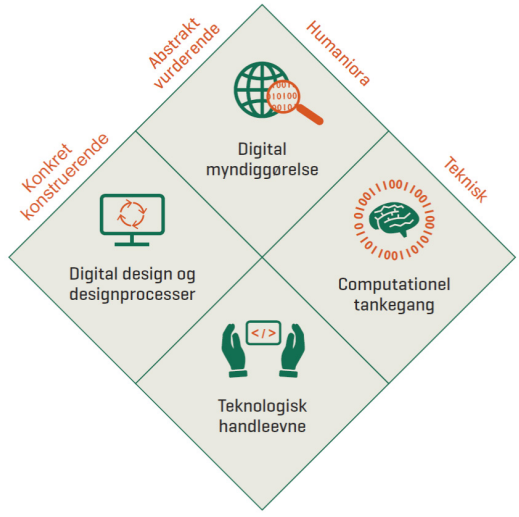
Figur 1.
Teknologiforståelsens fire kompetenceområder (Undervisningsministeriet, 2018, s. 8).



I vores læsning af de fire kompetenceområder omfatter de dels en humanistisk tilgang til teknologi (digital design og designprocesser samt digital myndiggørelse), dels en teknisk (teknologisk handleevne og computationel tankegang). Et andet snit i vores læsning af kompetencerne er, at digital design og designprocesser samt teknologisk handleevne begge er konkrete og konstruerende kompetencer. Digital myndiggørelse og computationel tankegang omfatter derimod de mere abstrakte og analytiske kompetencer. Se dette illustreret på vores revision af figuren fra fagets læseplan nedenfor (Figur 2). I denne artikel fokuserer vi på de tekniske og abstrakte kompetencer, der i læseplanen karakteriseres som "computationel tankegang", der er en direkte oversættelse fra det engelske begreb "computational thinking". I læseplanen fastlægges det, at computationel tankegang "omhandler analyse, modellering og strukturering af data og dataprocesser" (Undervisningsministeriet, 2018, s. 12). Eleverne skal efter at have haft faget kunne "anvende computationel tankegang til at beskrive velkendte og afgrænsede fænomener i hverdagen", de skal kunne "følge og anvende computationel tankegang i arbejdet med konkrete problemstillinger", og endeligt sigter faget mod, at eleverne "kan reflektere over og anvende computationel tankegang på problemstillinger fra omverdenen" (Undervisningsministeriet, 2018, s. 12).

Denne artikel sigter mod at udvikle et nuanceret sprog for, hvad computationel tankegang (computational thinking) er ved at afsøge den internationale litteratur på området med fokus på perioden 2013-2018, lige før mange lande valgte at implementere computational thinking som en del af skolens faglighed.

Figur 2.
Revideret model for de fire kompetence-områder, der specificerer temaer på tværs. Den orange tekst er tilføjet af Stine Ejsing-Duun i tidligere præsentationer. Den grafiske fremstilling af denne tilføjelse skal krediteres til Hans Reitzels Forlag i Philipps & Fougt (2020).



Videnskløft mellem implementering og teoretisk beskrivelse af computational thinking

I de seneste år er kompetencer inden for computational thinking gået fra at være specialiserede og knyttet til bestemte faggrupper til at være grundlæggende færdigheder, der er relevante for alle (Bocconi, Chioccariello, Dettori, Ferrari, Engelhardt, Kampylis & Punie, 2016). Denne tendens sætter verdens uddannelsessystemer over for en udfordring med at beslutte, hvad eleverne skal lære om computational thinking og hvordan (Bocconi, Chioccariello & Earp, 2018). Grover og Pea (2013) sætter i deres litteraturgennemgang ord på udfordringen:

> Clearly, much remains to be done to help develop a more lucid theoretical and practical understanding of computational competencies in children. What, for example, can we expect children to know or do better once they've been participating in a curriculum designed to develop [computational thinking] and how can this be evaluated?
> (Grover & Pea, 2013, s. 42)

Grover og Pea spørger dermed til, hvilken viden og hvilke kompetencer vi forventer, at børn har tilegnet sig for at kunne tænke og handle som værende kompetente i computational thinking. Disse spørgsmål er blevet aktuelle, efterhånden som computational thinking og forskellige former for teknologiforståelse er blevet, eller er i færd med at blive, udrullet i uddannelsessystemer rundt om i verden. I Estland og Israel har børn haft kodning i skolen siden 1990'erne, og i Storbritannien blev et curriculum for design og teknologi lanceret i 2014. I Sverige har man fra sommeren 2018 implementeret programmering i matematikundervisningen, og i Danmark afprøves Teknologiforståelse som fag i skolefagene fra 2018-2021 efter, at det har været afprøvet på valgfagsbasis i et år (Misfeldt, Jankvist, Geraniou & Bråting, 2020). Vi står, ifølge Bocconi et al. (2016), således i en situation, hvor implementeringen af computational thinking som kompetenceområde i skolen, nogle steder går væsentligt hurtigere end afklaringen af, hvad der ligger i denne kompetence.

Denne artikel tager fat i denne videnskløft mellem praksis og begrebsudvikling, som Grover og Pea (2013) og Bocconi et al. (2016) har påvist. Vores tilgang er at undersøge, hvordan computational thinking beskrives og defineres som kompetencer for elever i grundskole og på gymnasielt niveau i forskningslitteraturen i perioden 2013 til 2018. Dette gør vi dels for at bidrage til at skabe klarhed omkring, hvad en "computational thinker" forventes at være i stand til. Grunden til, at vi fokuserer på 2013 til 2018 er, at det er en periode, hvor anerkendelsen af, at der er behov for at se på computational thinking som en kompetence, der er vigtig for alle at stifte bekendtskab med, er udbredt. Samtidigt er det før denne implementering udbredt verden over. I perioden 2019-2021 er der kommet en række bidrag til litteraturen om computational thinking, der har en anden, mere systemisk og mere empirisk natur (se for eksempel Tikva & Tambouris, 2021). Disse artikler har i højere grad fokus på de brede erfaringer med at implementere computational thinking, for eksempel på hvordan bestemte lande har valgt at indskrive computational thinking i læseplanen (Misfeldt, Jankvist, Geraniou & Bråting 2020), eller på hvilke faglige relationer til for eksempel matematik (Kallia, van Borkulo,

Drijvers, Barendsen & Tolboom, 2021), der fremskrives eller undertrykkes i denne proces.

Vi vil begynde med at beskrive oprindelsen af begrebet computational thinking. Derefter vil vi undersøge, hvordan computational thinking beskrives som en kompetence i litteraturen om computational thinking i skolen. Dette gøres gennem en begrebskortlægning (Grant & Booth, 2009). Hovedindholdet af artiklen præsenterer de kompetencer, der er knyttet til computational thinking i litteraturen, og uddyber de fire mest fremtrædende kompetencer. Slutteligt konkluderer vi og fremstiller en model, der indfanger de i litteraturen mest fremhævede kompetenceområder knyttet til computational thinking.

Hvad er computational thinking?

Vi bygger vores forståelse af computational thinking på tre centrale kilder. Seymour Paperts (1980) konstruktionistiske tænkning, Jeanette Wings (2006, 2011) italesættelse af det brede behov for tekniske kompetencer, og Weintrop og kollegers (Weintrop, Beheshti, Horn, Orton, Jona, Trouille & Wilensky, 2016) beskrivelse af samspillet mellem computational thinking og naturvidenskabelig/matematisk skolefaglighed.

Viden om, hvordan computere fungerer, hvordan de programmeres, og hvilke typer konstruktioner og udforskning, børn kan foretage med dem, har været et fokus for uddannelsesforskning siden slutningen af 70'erne. Blandt de mere ansete tiltag og tilgange er matematikeren og pædagogen Seymour Paperts arbejde (1980) med konstruktionisme og udviklingen af programmeringssproget LOGO. For Papert var programmering og computational thinking langt hen ad vejen et middel til at opnå en bedre og mere motiverende skole, hvor eleverne bygger på deres forståelser af verden ved at interagere med matematiske begreber i et simuleret miljø. Datalogen Mitchel Resnick (2017) fortsætter denne konstruktionistiske tilgang med udviklingen af programmeringssproget Scratch, som kan muliggøre kreativ og legende læring. I de seneste 15 år har diskussionen om computational thinking i skolen i stigende grad handlet om at understøtte, at flere elever vælger naturvidenskabelige og tekniske karrierer, blandt andet for at øge den fremtidige arbejdsstyrkes digitale kompetencer (Danmarks Vækstråd, 2016). I den forbindelse har datalogen Jeannette M. Wings artikel fra 2006 spillet en vigtig rolle i at sætte fokus på vigtigheden af computational thinking. Ifølge Wing (2006) bør alle lære computational thinking fordi:

> Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability.
> (Wing, 2006, s. 33)

Wing mener således, at computational thinking er relevant for alle og i alle sammenhænge (2006, s. 35). Wings artikel er meget citeret og er blevet en vægtig stemme i forsøget på at få skolesystemer og regeringer til at øge fokus på dette område. I 2011 præciserer Wing sin definition af computational thinking:

> Computational thinking is 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.
> (Wing, 2011, s. 20)

I denne definition peger Wing på, at computational thinking er en kognitiv kompetence, der vedrører formulering af problemer og at skabe repræsentationer, der kan omsættes til noget en computationel enhed kan udføre. Vi ønsker i artiklen at komme nærmere ind på, hvordan vi kan tale om disse processer, hvor problemer formuleres og repræsenteres.

Et vigtigt bidrag til afklaringen af, hvordan vi kan beskrive computational thinking som kompetence, kommer fra datalogen Weintrop og kolleger (Weintrop et al., 2016). De fokuserer på "computational thinking practices", der sker gennem en kombination af datapraksisser, problemløsningspraksisser, modellering og simuleringspraksisser og systemtænkningspraksisser (Tabel 1).

Tabel 1.
Datalogiske tænkningspraksisser fra Weintrop et al. (2016, Tabel 2).

Datapraksisser	Modellering og simuleringspraksisser	Problemløsnings-praksisser	Systemtænknings-praksisser
Samle data	At bruge computermodeller til at forstå et begreb	At klargøre problemer til datalogiske løsninger	Udforske et komplekst system i sin helhed
Skabe data	At bruge computermodeller til at finde og teste løsninger	At programmere	Forstå relationerne internt i et system
Manipulere data	At vurdere computermodeller	At vælge effektive datalogiske værktøjer	Tænke i forskellige niveauer
Analysere data	At designe computermodeller	At vurdere forskellige tilgange til problemløsning	Kommunikere omkring et system
Visualisere data	At udvikle computermodeller	At udvikle modulære løsninger	Definere et system og håndtere kompleksitet
		At tænke abstrakt i datalogiske sammenhænge	
		Debugging – fejlsøgning og -rettelse	

Denne kategorisering af Weintrop et al. (2016) skal ifølge forfatterne ses i kontekst af matematik og naturfag. I denne artikel kortlægger vi derimod kompetencer for computational thinking i kontekst af teknologiforståelse gennem en litteraturgennemgang af nyere forskning i computational thinking, der bygger på erfaringer med, hvilke

kompetencer elever får gennem undervisning i computational thinking i skolekontekst. Begrebet computational thinking indkredses altså dels af den konstruktionistiske reformpædagogiske idé om, at eleverne skal lege og interagere sig til indsigt ved at udvikle konkrete artefakter. Wing sætter fokus på dels den brede vigtighed af computational thinking og dels den udpensling af, hvor tæt denne kompetence er knyttet til problemløsning og arbejdet med repræsentationer, der forudsætter forståelse for computationelle enheder. Endelig giver Weintrop et al., (2016) os en specificering af computational thinking i fire konkrete praksisser, der er tæt relateret til matematik og naturfagsundervisning.

Vi ser ikke de tre inspirationer som i intern modstrid, men deres fokus er forskelligt. Papert sætter fokus på leg og design samt matematiske begreber og problemer, Wing på problemløsning og repræsentation, og Weintrop tilføjer data, modeller og systemer samt kontinuiteten mellem computational thinking og de naturvidenskabelige skolefag.

Metode

Vores undersøgelse er baseret på en begrebskortlægning, der er en kortlægning og kategorisering af litteraturen i forhold til et afgrænset emne (Grant & Booth, 2009), her med fokus på spørgsmålet: Hvilke kompetencer knytter sig til computational thinking på grundskole og gymnasialt niveau? Vi har foretaget søgninger på databaserne Scopus og ProQuest. Søgningen, og den efterfølgende kriteriebaserede sortering, gav os 33 akademiske artikler, der hver især diskuterer computational thinking i forhold til kompetencer i grundskolen og på de gymnasiale uddannelser (K12). Artiklerne er udgivet fra 2013, hvor Grover og Peas litteraturgennemgang (2013) skabte overblik og viste et behov for fokus på kompetencer, og frem til 2018, da teknologiforståelse blev etableret som forsøgsfag i Danmark. Vi har brugt tre søgestrenge baseret på hovedemnerne i forskningsspørgsmålet og én streng til at ekskludere studier (se Tabel 2). De ord, der blev brugt til at ekskludere artikler fra søgningen, blev fundet iterativt ved at afprøve søgestrenge, scanne søgeresultater og derefter vælge ord, der udelukker emner, der ofte forekommer i resultaterne, men som vi betragtede som irrelevante. Vi har for eksempel fjernet kønsrelaterede diskussioner, der ikke fokuserede

på at definere computational thinking som kompetence, men snarere på at adressere kønsbarrierer (se kolonnen "Uden at medtage" i Tabel 2). Selve begrebet kompetence forstår vi i kontekst af Uddannelses- og forskningsministeriets specificering af begrebet kompetence, der omfatter: "(...) ansvar og selvstændighed og angiver evnen til at anvende viden og færdigheder i en arbejdssituation eller i studiemæssig sammenhæng" (Uddannelses- og forskningsministeriet, 2020). Kompetencer rummer ifølge denne forståelse et handlerum, evnen til samarbejde og at tage ansvar for egen læring.

Tabel 2.
Anvendte søgeblokke i gennemgangen.

Computational thinking	Kompetencer	K-12 (grundskolen og gymnasiale uddannelser)	Uden at medtage
"computational thinking" ELLER "computational concepts"	"21st century" ELLER competenc* ELLER "skill" ELLER "citizen" ELLER societ* ELLER abilit* ELLER "knowledge" ELLER learn* ELLER experience ELLER mastery	"K-12" ELLER "K-6" ELLER "school children" ELLER "secondary education" ELLER "primary education" ELLER "primary school" ELLER "secondary school" ELLER "school" ELLER children ELLER pupil ELLER citizen ELLER kid	gender ELLER "teaching methods" ELLER questionnaires ELLER "statistical analysis" ELLER "pretests" ELLER "gender differences" ELLER observation ELLER "teacher attitudes" ELLER "online courses" ELLER "college faculty" ELLER curricula ELLER teacher ELLER disability ELLER adhd

Søgningen gav 29 resultater i ProQuest og 46 resultater i Scopus. Disse 75 artikler blev screenet for gengangere, hvorved ni artikler blev udelukket. De resterende 66 artikler blev screenet ved at læse overskriften og manchet (abstract). Vi frasorterede derefter artikler på baggrund af overskrift (i forhold til relevans – handlede det om computational

thinking i grundskolen eller på de gymnasiale uddannelser, K12), sprog (vi fokuserede på engelske artikler) og alder (vi fokuserede på perioden 2013-2018). Tilbage var 43 relevante artikler, som blev gennemlæst i forhold til, hvordan forskningen blev udført (metoder), hvor forskningen blev foretaget (land), uddannelsestrin (alder/klasse), det emne, som forskningen vedrører, anvendte definitioner af computational thinking i artiklen, hvilke kompetencer forfatterne forbinder med computational thinking, konklusioner draget i relation til computational thinking i løbet af artiklen og motivationen til at introducere computational thinking i skolen, der blev udtrykt i artiklen. Baseret på gennemgangen af de 43 artikler blev 33 artikler vurderet som relevante i forhold til at forstå, hvilke kompetencer der er knyttet til computational thinking i de senere år, før Teknologiforståelse blev implementeret i Danmark.

Under gennemgangen af artiklerne har vi kigget efter de ord, som forfatterne bruger til at karakterisere computational thinking som kompetencer. Vi har herefter fremhævet ordene og samlet data om deres hyppighed i Tabel 3. Hvis to artikler bruger det samme begreb i en varierende kontekst (for eksempel "problemløsning" eller "problemløsningskompetencer") som en måde at danne sig et begreb om computational thinking, betragter vi dem som det samme begreb. Vi har dog ikke forenet nært beslægtede begreber eller synonymer, som for eksempel "algoritmisk tænkning" og "automatisering", som man kunne argumentere for hænger sammen. Ved at læse disse 33 identificerede vi på denne måde 17 kompetencer, der blev nævnt i forbindelse med computational thinking.

Mest fremhævede kompetencer for computational thinking

Vi har fokuseret på definition og karakteristik af de kompetencer, der er nævnt af de fleste forfattere som værende centrale for computational thinking. De 17 identificerede kompetencer forbundet med computational thinking fremgår af Tabel 3 nedenfor.

Tabel 3.
Kompetencer nævnt i de gennemgåede artikler som forbundet med computational thinking.

Kompetencer	Antal	Procentdel
Problem-solving (problemløsning)	33	100 %
Abstraction (abstraktion)	22	67 %
Algorithmic thinking (algoritmisk tænkning)	20	61 %
Modelling (modellering)	16	48 %
Decomposition (nedbrydning)	12	36 %
Conditional logic (betingelseslogik)	9	27 %
Analyzing (analyse)	9	27 %
Automation (automatisering)	7	21 %
Debugging (fejlfinding og -rettelse)	7	21 %
Data collection (dataindsamling)	5	15 %
System thinking (systemtænkning)	5	15 %
Parallelism (parallelisme)	4	12 %
Creativity (kreativitet)	4	12 %
Logical thinking (logisk tænkning)	4	12 %

Composition (komposition)	3	9 %
Collaboration (samarbejde)	3	9 %
Recursion (rekursion)	2	6 %

Alle disse kompetencer er naturligvis relevante, men her har vi valgt at fokusere på de fire hyppigst nævnte kompetenceområder, idet vi antager, at de er mest centrale. De omfatter: problemløsning, abstraktion, algoritmisk tænkning og modellering. I det følgende udfolder vi én efter én, hvordan litteraturen definerer disse kompetencer.

Problemløsning

Computational thinking er en tilgang til problemløsning. Weintrop et al. (2016) beskriver problemløsning som en af de fire computational thinking-praksisser, og Lye og Koh (2014) har ligeledes foretaget en gennemgang af empiriske undersøgelser i K-12 (svarende til grundskolen og gymnasiale uddannelser i Danmark) samt på relevante videregående uddannelser for at få indblik i, hvordan der undervises i computational thinking. De kritiserer projekterne for ikke at støtte eleverne i deres problemløsningsprocesser, samt at refleksion over processen får for lidt plads i undervisningen.

I samtlige 33 artikler nævnes problemløsning i forbindelse med computational thinking. Størstedelen af de gennemgåede artikler beskriver computational thinking som evnen til at formulere og løse problemer ved hjælp af forskellige datalogiske tilgange og metoder (for eksempel gennem abstraktion og algoritmisk tænkning), men uden at definere problemløsning i yderligere detaljer (Basu, Biswas & Kinnebrew, 2017; Brady, Orton, Weintrop, Anton, Rodriguez & Wilensky, 2017; Csernoch, Biró, Máth & Abari, 2015; Denner, Werner, Campe & Ortiz, 2014; Denning, 2017; Durak & Saritepeci, 2017; Gadanidis, 2017; Gadanidis, Clements & Yiu, 2018; Jun, Han & Kim, 2017; Lee & Soep, 2016; Leonard, Buss, Gamboa, Mitchell, Fashola, Hubert & Almughyirah, 2016; Martin, 2017; Matsumoto & Cao, 2017; Pugnali, Sullivan & Bers, 2017; Psycharis & Kallia, 2017; Sáez-López & Sevillano-García, 2017; Shein, 2014; Wang, Wang & Liu, 2014). De resterende artikler går noget tættere på, hvad problemløsning er i

relation til computational thinking.

Når eleven møder et problem, så skal eleverne ifølge Liu, Zhi, Hicks og Barnes (2017) lære at håndtere det igennem en proces, der omfatter at 1) identificere problemstillingen, 2) repræsentere den på en passende måde, 3) vælge en løsningsstrategi, 4) udføre strategien og 5) evaluere løsningen. Forfatterne sammenligner erfarne problemløsere (såkaldte "eksperter") med nye problemløsere og understreger, at eksperter bruger mere tid på de første faser af processen. Endvidere viser undersøgelsen, at eksperter håndterer problemerne i mindre dele og overvejer flere alternative løsninger. Liu et al. (2017) repræsenterer således en kognitiv, planmæssig og analytisk tilgang til problemløsning, der er sekventiel og ikke omfatter iterationer.

Rose, Habsgood & Jay (2017) foreslår en fleksibel og mere situeret tilgang til forståelse af problemløsning, der omfatter to tilgange til problemløsning. For det første er der den analytiske top-down-tilgang, som – ikke ulig den sekventielle tilgang præsenteret ovenfor – indebærer planlægning af processen og gennemførelse af planen. For det andet er der en nedefra-og-op-tilgang – den såkaldte "bricolage"-tilgang – hvor problemet løses gennem interaktion med materialer, og viden dermed opnås gennem oplevelsen af interaktionen. Denne tilgang bygger Rose et al. (2017) på konstruktionistiske tænkere, der beskriver, hvordan elever finder en løsning på et givent problem igennem arbejdet, som det skrider frem, snarere end at holde sig til en på forhånd fastlagt plan (Papert & Harel, 1991). Konstruktionismen sætter et pædagogisk fokus på eleverne som ansvarlige for deres egen læring. I stedet for at modtage og gengive viden, som læreren præsenterer for dem, lærer eleverne gennem bricolage: de tilrettelægger og omorganiserer velkendte materialer (Kafai & Burke, 2014). I modsætning til problemløsning, hvor elever nedbryder udfordringer ovenfra og ned i mere fordøjelige komponenter, beskriver bricolage problemløsning som en dialog med situationen, hvor den endelige løsning kommer til sidst (Kafai & Burke, 2014). Sullivan & Heffernan (2016) skelner også mellem disse to tilgange og viser endvidere, at elever udvikler deres tilgang til problemløsning over tid fra forsøg-fejl-metoden (bricolage) til mere avancerede heuristiske tilgange (planmæssige og analytiske). Kafai & Burke (2013) beskriver det at løse et problem med computational thinking som:

> [...] extending computer science principles to other disciplines in order to help break down the elements of any problem, determine their relationship to each other and the greater whole, and then devise algorithms to arrive at an automated solution. (Kafai & Burke, 2013, s. 62)

Her tager Kafai og Burke udgangspunkt i, at når vi løser problemer igennem computational thinking, så anvendes "datalogiske principper", hvilket er metoder, der almindeligvis hører til datalogi. Det kunne for eksempel være rekursion – en metode til at reducere kompleksitet i en proces (Cetin & Dubinsky, 2017). Kafai og Burke antyder, at begrebet "computational thinking" er for snævert. De foreslår i stedet at sætte begrebet "computational participation" på elevernes skema. Dette indebærer et fokus på problemløsning igennem algoritmisk tænkning, som omfatter at gå fra at lære kodning som et mål i sig selv til at lære at lave applikationer. Og at gå fra at lære eleverne værktøjer til i stedet at tilbyde dem at være en del af et fællesskab af kreative skabere af teknologi. Til sidst omfatter det et skift fra at bygge hele applikationer op fra bunden til en tilgang, der giver mulighed for at blande og kombinere og ændre eksisterende applikationer. Kafai og Burke (2013) betoner samtidigt, at skolen bør opmuntre eleverne til problemløsning på tværs af fagligheder, hvor problemerne nedbrydes i dele.

Alle artikler anerkender, at computational thinking overordnet handler om problemløsning. En del af artiklerne gør dog ikke ret meget ud af, hvordan problemløsning foregår og bidrager til computational thinking. De artikler, der behandler problemløsning i detaljer, ser enten problemløsning som en trinvis proces eller som en mere åben praksis, der på forskellige måder relaterer til de andre kompetencer inden for computational thinking. Endeligt bidrager begrebet "computational participation" med at koble computational thinking til en social og tværfaglig tilgang til problemløsning.

Abstraktion

Abstraktion nævnes i cirka to tredjedele af de gennemgåede artikler, og der er derfor bred enighed om, at kompetencen udgør en del af computational thinking. Grover og Pea (2013) siger endvidere, at netop abstraktion er et hovedprincip i computational thinking, der adskiller denne tænkning fra andre former for tænkning. Abstraktion er et udtryk med en række betydninger, og det giver derfor mening at undersøge, hvad den specifikke betydning er, eller kan være, i de forskellige forskningsbidrag, som vi har gennemgået, og hvordan

begrebet forholder sig til de andre kritiske elementer i computational thinking.

En væsentlig del af artiklerne nævner abstraktion i en computational thinking-kontekst uden at definere den (Brady et al., 2017; Cho & Lee, 2017; Denner et al., 2014; Denning, 2017; Durak & Saracipeti, 2018; Gadanidis, 2017; Jun et al., 2017; Lee, Mauriello, Ahn & Bederson, 2014; Leonard et al., 2016; Lye & Koh, 2014; Psycharis & Kallia, 2017; Román-González, Pérez-González & Jiménez-Fernández, 2016; Sáez-López & Sevillano-García, 2017; Sanford & Naidu, 2016; Segredo, Miranda & León, 2017; Shein, 2014; Werner, Denner & Campe, 2014). Nedenfor gennemgår vi de artikler, der eksplicit forholder sig til abstraktion.

Flere af artiklerne betragter abstraktion som en kognitiv tilgang til at navigere i store mængder empiriske observationer og fokusere på væsentlige delelementer. For eksempel definerer National Science Foundation abstraktion som en "reduktion af information og detaljer med henblik på at fokusere på begreber, der er relevante for forståelse og løsning af problemer" (Grover & Pea, 2013, s. 39). Det betyder, at abstraktion betragtes som gruppering af informationer, hvilket reducerer kompleksiteten, og overflødig information udelades. Denne tilgang til abstraktion deles af Wang, Wang og Liu, der definerer abstraktion, som "evnen til at finde et passende detaljeniveau med henblik på at definere og løse et problem" (2014, s. 4). I samme stil skriver Wing i sin artikel fra 2011 (citeret i Cetin og Dubinsky, 2017 samt i Grover og Pea, 2013) om abstraktion som et udtryk, der har at gøre med at "definere mønstre og generalisere fra det specifikke tilfælde samt at beskrive noget gennem parametre". Ifølge Wing (2011) kan man gennem "abstraktion af et givent problem reducere kompleksiteten ved at lade et objekt repræsentere mange, hvorved de essentielle egenskaber ved det, der adresseres, indfanges". Abstraktionen tillader altså, at vi "fjerner detaljer fra problemet og formulerer en løsning i mere generelle termer" (Rose et al., 2017).

Opfattelsen af abstraktion som en metode til at repræsentere en problemstillings karakteristika i klasser, der deler væsentlige egenskaber, suppleres i visse artikler af en mere raffineret opdeling mellem empirisk og reflekterende abstraktion. Et kritisk eksempel på dette er Cetin og Dubinskys (2017) brug af Piagets begreber. De tager udgangspunkt i flere forskningsbidrag, der beskriver abstraktion i forhold til programmering og computational thinking, og konkluderer på denne baggrund, at den eksisterende litteratur i vid udstrækning definerer abstraktion som en metode til at beslutte, hvilke detaljer man skal fokusere på, og hvilke man skal ignorere. Forfatterne sammenligner denne forståelse med det, som Piaget beskriver som "empirisk

abstraktion", hvilket er interessant, fordi Piaget har peget på, at dette ikke er et stærkt abstraktionsniveau. Cetin og Dubinsky skriver, at "empirisk abstraktion tager udgangspunkt i, at et individ udleder viden fra et objekts egenskaber ved at vælge at ignorere nogle af dem og fokusere på andre" (2017, s. 76). Årsagen til, at denne metode ikke er tilstrækkelig, er, at den empiriske abstraktion beror på individets oplevelser af et givent fænomen. For eksempel kan et barn se mange hunde og genkende et dyr som en hund men også se løven og hyænen som en hund ud fra samme karakteristika. Cetin og Dubinsky (2017) argumenterer for at følge Piagets forslag om at bruge en stærkere form for abstraktion nemlig den refleksive: "In reflective abstraction, knowledge is not drawn from the properties of the object, rather it is drawn from the general coordination[s] of actions" (Cetin & Dubinsky, 2017, s. 76). Her vil eleven observere et givent fænomen, reflektere over det og selv konstruere en abstraktion af fænomenet. Det kan for eksempel være at overskue en mængde og forstå, at uanset hvordan den tælles, vil man nå samme antal.

Shein (2014) understreger betydningen af abstrakt tænkning i computational thinking og sætter spørgsmålstegn ved, om børn er i stand til at tænke abstrakt og dermed er klar til at lære at kode (Shein, 2014, s. 16). Gadanidis (2017) understreger, at idéen om at kaste børn ud i abstraktion ikke er vidt accepteret blandt andet på grund af Piagets beskrivelse af udviklingsstadier, der sætter abstraktion på det højeste udviklingstrin. Gadanidis argumenterer dog for, at børn dagligt bruger en (empirisk) abstraktion for at genkende og klassificere verden omkring sig. Leonard et al. (2016) giver også et konkret eksempel på børn, der anvender abstraktion i programmering, da en gruppe elever evner at afvige fra den traditionelle spilgenre, ved at studere disse spil, før de skaber deres eget spil.

Opsummerende kan abstraktion forstås som en analytisk tilgang til at betegne og beskrive elementerne ved et givent problem, der leder til en reduktion af problemets kompleksitet og gør det håndterbart for et computerprogram. Abstraktion kan foregå empirisk ved at tage udgangspunkt i oplevelsen af fænomener og refleksivt og iterativt ved at skabe stærke grupperinger, der bygger på fænomenernes karakteristika. Det er omdiskuteret, i hvilken udstrækning børn kan tænke i abstraktioner, men der er eksempler på, at det lykkes, når det stilladseres.

Algoritmisk tænkning

Algoritmisk tænkning fremhæves som en central kompetence i 20 af de gennemgåede artikler. Kafai og Burke (2013) fremhæver algoritmer

som en nøglekomponent i computational thinking, der ifølge dem består af tre processer: 1) nedbryd et problem til mindre delproblemer og komponenter, 2) bestem forholdet mellem disse, og 3) udarbejd algoritmer til at løse problemet. Udarbejdelsen af en algoritme kan forstås som en række trin til at løse et problem (Cho & Lee, 2017). Algoritmisk tænkning er således tæt forbundet til problemløsning; en algoritme er en proces, der løser en udfordring. Grover og Pea (2013) definerer ikke algoritmisk tænkning direkte, men henviser til Aho, der definerer computational thinking gennem algoritmer, som: "(...) the thought processes involved in formulating problems so 'their solutions can be represented as computational steps and algorithms'" (Grover & Pea, 2013, s. 39). Grover and Pea beskriver desuden algoritmer som "(...) tools for developing and expressing solutions to computational problems" (2013, s. 39). Denning (2017) ser, ligesom Grover og Pea, algoritmisk tænkning som et syntetisk supplement til mere analytisk problemløsning, men han gør endnu mere ud af menneskelig dømmekraft og ræsonnement i den forbindelse: "(...) an algorithm is not any sequence of steps, but a series of steps that control some abstract machine or computational model without requiring human judgment" (Denning, 2017, s. 33).

Samspillet mellem problemløsning, dømmekraft og argumentation står centralt hos både Grover og Pea (2013), Matsumoto og Cao (2017), Denning (2017) og Segredo, Miranda og León (2017). Segredo, Miranda og León (2017) giver desuden en meget specifik trinmodel for algoritmisk tænkning med reference til Futschek (2006), der går fra (1) analyse over (2) repræsentation (3) til fastlægning af instruktioner, (4) opbygning af en algoritme, (5) analyse af mulige tilfælde og endelig (6) iterativ forbedring af algoritmen. Segredo, Miranda og León konkluderer, at algoritmisk tænkning: "(...) includes a deep understanding of a given problem and an identification of the operations needed to solve that particular problem. The algorithm itself is an expression of this understanding, a synthesis of the understanding" (2017, s. 41).

Kort sagt indebærer algoritmisk tænkning, at et problem analyseres og forstås, så man kan danne sig et klart billede over, hvordan man løser det, udtrykker det i trin, der er forståelige for en computer, og kritisk reflekterer over denne syntese.

Modellering

Knap halvdelen af de inkluderede studier nævner modellering som en kompetence, der er knyttet til computational thinking. Ti af disse artikler udfolder dog ikke begrebet (Denner et al., 2014; Gadanidis, 2017; Durak & Saritepeci, 2018; Psycharis & Kallia, 2017; Cho & Lee,

2017; Durak & Saritepeci, 2018; Psycharis & Kallia, 2017; Cho & Lee, 2017; Román-González et al., 2017; Basu et al., 2017; Sáez-López & Sevillano-García, 2017). Det er måske ikke overraskende – i alle fald påpeger Denning (2017), at det i de dominerende definitioner af computational thinking er en mangel, at den datalogiske model ikke nævnes. Dette er ifølge Denning (2017) en fejltagelse. Hele formålet med at engagere sig i abstraktion, nedbrydning og datarepræsentation er at skabe en model til at udføre et bestemt stykke arbejde (Denning, 2017, s. 35). De analytiske processer har simpelthen modellering som mål. Modellen er i sig selv en abstraktion, og algoritmerne er en serie af trin til kontrollere denne model uden, at et menneske skal indblandes (Denning, 2017).

En datalogisk model er et løsningsforslag, hvori relationer mellem alle parametre er indeholdt (Sanford & Naidu, 2017). Modellering bygger på en analyse af problemstillingen, antagelser om hvilke parametre, der har betydning for problemet og en prioritering af hvilke, der er mest centrale samt hvilke relationer, der er mellem dem. Disse relationer udgør grundlaget for selve modellen (Sanford & Naidu, 2017). Gadanidis (2017) og Gadanidis et al., (2018) argumenterer for, at arbejdet med modellering har et potentiale i forhold til at forstå komplekse sammenhænge, idet elever umiddelbart kan se konsekvenserne ved at ændre på en models kode. For at undgå misforståelser og frustrationer er det dog vigtigt, at eleverne ved, at modellen altid vil være en forsimpling af virkeligheden (Sullivan & Heffernan, 2016).

Modellering er en mere sofistikeret løsningstilgang end 'trial and error' og, ifølge Sullivan og Heffernan (2016), en tilgang, der skaber indsigt i helheder. Gennem arbejdet med modellering udfordres eleverne til at tænke i hypoteser ("hvad nu hvis…?") og omsætte dem til matematiske og datalogiske modeller (Sanford & Naidu, 2016). Når eleverne modellerer, så repræsenterer de data og gengiver mønstre. Denne proces kan bruges epistemisk til at undersøge og skabe forståelse for for eksempel (natur-)fænomener i verden (Matsumoto & Cao, 2017; Sullivan & Heffernan, 2016; Gadanidis, 2017; Gadanidis et al., 2018) eller videnskabelige simulationer (Román-González et al., 2016). Sullivan og Heffernan (2016) viser eksempelvis, at eleverne, der skulle skabe en fungerende model af et biologisk system, først måtte forstå og undersøge systemets mekanismer.

Sanford og Naidu (2016) knytter modellering sammen med en undersøgende tilgang til matematik. Dette åbner op for, at eleverne kan afsøge og få forståelse for de potentielle forskellige løsninger, der kan være på en problemstilling, som kan præsentere sig igennem

simulationen. Erkendelsen af, at der ikke bare findes en løsning på et problem, kan øge elevernes lyst til at kaste sig ud i innovation og eksperimenter ifølge Sanford og Naidu (2016). Denne slags simulationer hjælper ligeledes med at forstå fænomener i verden og kan flytte grænserne for, hvad vi ved (Sanford & Naidu, 2016).

Konklusion: en model for computational thinking på tværs af fag

Vi har i denne artikel undersøgt, hvad forskningslitteraturen lægger vægt på, når kerneelementerne i computational thinking gøres til en undervisningsorienteret kompetence. Litteraturen fra 2013 til 2018 viser, at problemløsning, abstraktion, algoritmer og modellering alle er centrale tematikker.

Vores udgangspunkt for at undersøge, hvad forskningslitteraturen peger på som centrale elementer af computational thinking, er hentet hos Papert (1980), Wing (2006 & 2011) samt hos Weintrop et al. (2016). Gennemgangen viser, at Wings kognitive forståelse af computational thinking, som den tydeligt er udtrykt i hendes paper fra 2011, er i overensstemmelse med det, der fremhæves i litteraturen. Det er ikke overraskende, da Wings arbejde på mange måder har været startskud til det store fokus, som området har fået i grundskolesammenhænge.

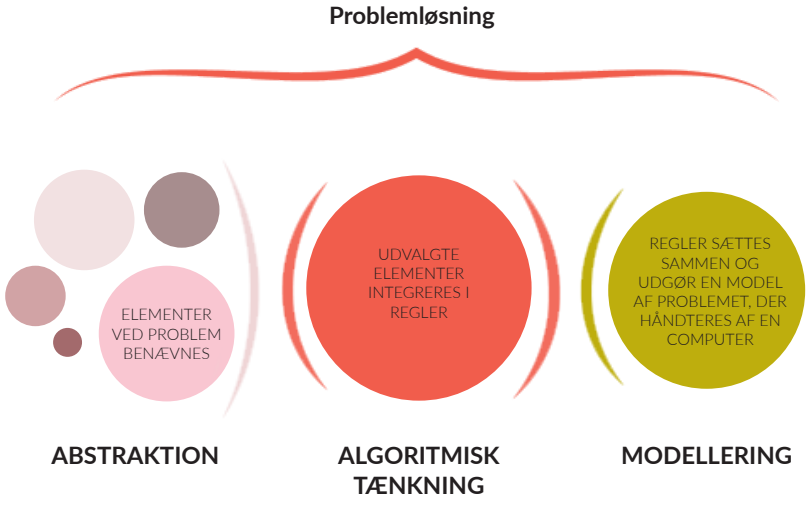
I læseplanen for teknologiforståelse sættes computationel tankegang lig med " (...) analyse, modellering og strukturering af data og dataprocesser" (Undervisningsministeriet, side 12). Denne forståelse er tæt på den, der beskrives i Weintrop et al. (2016) som relevante computational thinking-praksisser i relation til matematik og naturfag dog uden den samme fokus på data og systemtænkning.

Vores begrebskortlægning viser, at computational thinking ofte bruges til at betegne to overordnede problemløsningsmekanismer: abstraktion og algoritmisk tænkning. Begge anvendes til opbygning af modeller for et givet problem. Abstraktion er analytisk og handler om at beskrive elementerne i et problem på en måde, der leder til en reduktion af problemets kompleksitet og gøre det håndterbart for et computerprogram. Algoritmisk tænkning er en syntetisk proces, hvor problemets elementer integreres i regler, der griber problemet an som helhed. Gennem denne rammesætning igennem regler (algoritmer) fremhæves visse elementer af problemet, mens andre

ignoreres. Kompetencemæssigt er computational thinking således en vekselvirkning mellem analyse og syntese samt kvalificerede gæt. I denne sammenhæng udgør modellering en repræsentation af det aktuelle problem samt de foreslåede regler til håndtering af det. Hvor abstraktion og algoritmisk tænkning er argumenterende kompetencer, ser vi i denne sammenhæng modellering som evnen til at aktualisere og repræsentere en forståelse af et omverdensfænomen, der rent faktisk kan løse problemet eller beskrive problematikken.

I figuren nedenfor (Figur 3) beskriver vi forholdet mellem hovedelementerne i computational thinking, der blev afledt i vores litteraturgennemgang. Vi forstår forholdet mellem de fire begreber problemløsning, abstraktion, algoritmisk tænkning og modellering på den måde, at computational thinking grundlæggende er en problemløsningsstrategi, men at denne strategi bygger på opbygningen af modeller af problemerne, så de kan løses af et informationsprocesseringssystem. Praktisk set involverer dette mindst to kritiske processer, der er gensidigt afhængige, nemlig abstraktion og algoritmisk tænkning.

Figur 3.
Computational thinking som et sæt af kompetencer. Modellen omfatter de mest centrale kompetencer knyttet til computational thinking baseret på begrebskortlægningen.



Computational thinking er en problemløsningsstrategi, der gennemfører passende modellering ved hjælp af abstraktion og algoritmisk tænkning. En god løsning forudsætter en god forståelse af problemfeltet, og at de rette elementer udvælges (abstraktion) og integreres i regler (algoritmisk tænkning), således at modellen håndterer problemet uden at skabe nye problemer.

Vores håb med denne begrebskortlægning har på den ene side været at vise, hvordan computational thinking opfattes i den internationale forskningslitteratur, og på den anden side at bidrage til, at computational thinking kan blive en del af den danske skole på en god måde. Den model, vi bringer som opsamling på arbejdet, giver en empirisk funderet beskrivelse af, hvilke elementer der i forskningslitteraturen anses som vigtige for computational thinking.

Referencer

Basu, S., Biswas, G. & Kinnebrew, J. S. (2017). Learner Modeling for Adaptive Scaffolding in a Computational Thinking-Based Science Learning Environment. *User Modeling and User-Adapted Interaction*, *27*(1), 5-53. DOI: 10.1007/s11257-017-9187-0

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P. & Punie, Y. (2016). *Developing Computational Thinking in Compulsory Education: Implications for Policy and Practice*. European Commision JRC Science for Policy Reports. DOI: 10.2791/792158

Bocconi, S., Chioccariello, A. & Earp, J. (2018). *The Nordic Approach to Introducing Computational Thinking and Programming in Compulsory Education*. Nordic@BETT2018 Steering Group. https://doi.org/10.17471/54007

Brady, C., Orton, K., Weintrop, D., Anton, G., Rodriguez, S. & Wilensky, U. (2016). All Roads Lead to Computing: Making, Participatory Simulations, and Social Computing as Pathways to Computer Science. *IEEE Transactions on Education*, *60*(1), 59-66. DOI: 10.1109/TE.2016.2622680

Børne- og Undervisningsministeriet. (2018). Lokaliseret [22. maj 2018] på: https://uvm.dk/aktuelt/nyheder/uvm/2018/maj/180516-forsoeg-med-teknologiforstaaelse-i-folkeskolen-i-udbud

Cetin, I. & Dubinsky, E. (2017). Reflective Abstraction in Computational Thinking. *The Journal of Mathematical Behavior*, *47*, 70-80. http://dx.doi.org/10.1016/j.jmathb.2017.06.004

Cho, Y. & Lee, Y. (2017). Possibility of Improving Computational Thinking Through Activity Based Learning Strategy for Young Children. *Journal of Theoretical and Applied Information Technology*, *95*(18), 4385-4393.

Csernoch, M., Biró, P., Máth, J. & Abari, K. (2015). Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments. *Informatics in Education*, *14*(2), 175-197. http://dx.doi.org/10.15388/infedu.2015.11

Danmarks Vækstråd (2016). *Rapport om kvalificeret arbejdskraft*. Lokaliseret [marts 2021] på: https://www.co-industri.dk/files/2021-07/Rapport_om_kvalificeret_arbejdskraft.pdf

Denner, J., Werner, L., Campe, S. & Ortiz, E. (2014). Pair Programming: Under What Conditions Is It Advantageous for Middle School Students? *Journal of Research on Technology in Education*, *46*(3), 277-296. https://doi.org/10.1080/15391523.2014.888272

Denning, P. J. (2017). Remaining Trouble Spots with Computational Thinking. *Communications of the ACM*, *60*(6), 33-39. DOI: 10.1145/2998438

Durak, H. Y. & Saritepeci, M. (2018). Analysis of the Relation between Computational Thinking Skills and Various Variables with the Structural Equation Model. *Computers & Education*, *116*, 191-202. https://doi.org/10.1016/j.compedu.2017.09.004

Futschek, G. (2006). Algorithmic Thinking: The Key for Understanding Computer Science. In R. T. Mittermeir (Ed.), *Informatics Education – The Bridge between Using and Understanding Computers* (s. 159-168). Springer. DOI: http://dx.doi.org/10.1007/11915355_15

Gadanidis, G. (2017). Artificial Intelligence, Computational Thinking, and Mathematics Education. *The International Journal of Information and Learning Technology, 34*(2), 133-139. https://doi.org/10.1108/IJILT-09-2016-0048

Gadanidis, G., Clements, E. & Yiu, C. (2018). Group Theory, Computational Thinking, and Young Mathematicians. *Mathematical Thinking and Learning, 20*(1), 32-53. https://doi.org/10.1080/10986065.2018.1403542

Grant, M. J. & Booth, A. (2009). A Typology of Reviews: An Analysis of 14 Review Types and Associated Methodologies. *Health Information & Libraries Journal, 26*(2), 91-108. http://dx.doi.org/10.1111/j.1471-1842.2009.00848.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. http://dx.doi.org/10.3102/0013189X12463051

Jun, S., Han, S. & Kim, S. (2017). Effect of Design-Based Learning on Improving Computational Thinking. *Behaviour & Information Technology*, *36*(1), 43-53. https://doi.org/10.1080/0144929X.2016.1188415

Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E. & Tolboom, J. (2021). Characterising Computational Thinking in Mathematics Education: A Literature-Informed Delphi Study, *Research in Mathematics Education*, *23*(2), 159-187, DOI: 10.1080/14794802.2020.1852104

Kafai, Y. B. & Burke, Q. (2013). Computer Programming Goes Back to School. *Phi Delta Kappan*, *95*(1), 61-65. https://doi.org/10.1177%2F003172171309500111

Kafai, Y. B. & Burke, Q. (2014). *Connected Code: Why Children Need to Learn Programming*. The MIT Press. https://doi.org/10.7551/mitpress/9992.001.0001

Lee, C. H. & Soep, E. (2016). None But Ourselves Can Free Our Minds: Critical Computational Literacy as a Pedagogy of Resistance. *Equity & Excellence in Education, 49*(4), 480-492. https://doi.org/10.1080/10665684.2016.1227157

Lee, T. Y., Mauriello, M. L., Ahn, J. & Bederson, B. B. (2014). CTArcade: Computational Thinking with Games in School Age Children. *International Journal of Child-Computer Interaction*, 2(1), 26-33. https://doi.org/10.1016/j.ijcci.2014.06.003

Leonard, J., Buss, A., Gamboa, R., Mitchell, M., Fashola, O. S., Hubert, T. & Almughyirah, S. (2016). Using Robotics and Game Design to Enhance Children's Self-Efficacy, STEM Attitudes, and Computational Thinking Skills. *Journal of Science Education and Technology*, *25*(6), 860-876. DOI: 10.1007/s10956-016-9628-2

Liu, Z., Zhi, R., Hicks, A. & Barnes, T. (2017). Understanding Problem Solving Behavior of 6–8 graders in a Debugging Game. *Computer Science Education*, *27*(1), 1-29. https://doi.org/10.1080/08993408.2017.1308651

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. http://dx.doi.org/10.1016/j.chb.2014.09.012

Martin, C. (2017). Expressing Youth Voice through Video Games and Coding. *Knowledge Quest*, *45*(4), 50-57.

Matsumoto, P. S. & Cao, J. (2017). The Development of Computational Thinking in a High School Chemistry Course. *Journal of Chemical Education*, *94*(9), 1217-1224. https://doi.org/10.1021/acs.jchemed.6b00973

Misfeldt, M., Jankvist, U. T., Geraniou, E. & Bråting, K. (2020). Relations between Mathematics and Programming in School: Juxtaposing Three Different Cases. I: A. Donevska-Todorova, E. Faggiano, J. Trgalova, Z. Lavicza, R. Weinhandl, A. Clark-Wilson & H.-G. Weigand (red.), *Mathematics Education in the Digital Age (MEDA). Proceedings of the 10th ERME Topic Conference* (s. 255-262). Johannes Kepler University.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas.* Basic Books.

Papert, S. & Harel, I. (1991). Situating constructionism. I: I. Harel & S. Papert, *Constructionism* (s. 1-11). Ablex Publishing corporation.

Psycharis, S. & Kallia, M. (2017). The Effects of Computer Programming on High School Students' Reasoning Skills and Mathematical Self-Efficacy and Problem Solving. *Instructional Science*, *45*(5), 583-602. DOI: 10.1007/s11251-017-9421-5

Philipps, M. R. & Fougt, S. S. (2020). *Teknologiforståelse i et scenariedidaktisk perspektiv: indskoling, mellemtrin og udskoling*. Hans Reitzels Forlag.

Pugnali, A., Sullivan, A. & Bers, M. U. (2017). The Impact of User Interface on Young Children's Computational Thinking. *Journal of Information Technology Education: Innovations in Practice*, *16*(1), 171-193. https://doi.org/10.28945/3768

Resnick, M. (2017). *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*. The MIT press.

Román-González, M., Pérez-González, J. C. & Jiménez-Fernández, C. (2016). Which Cognitive Abilities Underlie Computational Thinking? Criterion Validity of the Computational Thinking Test. *Computers in Human Behavior, 72*, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Rose, S., Habgood, J. & Jay, T. (2017). An Exploration of the Role of Visual Programming Tools in the Development of Young Children's Computational Thinking. *Electronic Journal of e-Learning*, *15*(4), 297-309.

Sáez-López, J. M. & Sevillano-García, M. L. (2017). Sensors, Programming and Devices in Art Education Sessions. One Case in the Context of Primary Education / Sensores, programación y dispositivos en sesiones de Educación Artística. Un caso en el contexto de Educación Primaria. *Culture and Education*, *29*(2), 350-384. https://doi.org/10.1080/11356405.2017.1305075

Sanford, J. F. & Naidu, J. T. (2016). Computational Thinking Concepts for Grade School. *Contemporary Issues in Education Research (CIER)*, *9*(1), 23-32. https://doi.org/10.19030/cier.v9i1.9547

Sanford, J. F. & Naidu, J. T. (2017). Mathematical Modeling and Computational Thinking. *Contemporary Issues in Education Research (CIER)*, *10*(2), 158-168. https://doi.org/10.19030/cier.v10i2.9925

Segredo, E. M., Miranda, G. & León, C. (2017). Towards the Education of the Future: Computational Thinking as a Generative Learning Mechanism. *Education in the Knowledge Society (EKS), 18*(2). https://doi.org/10.14201/eks2017182335

Shein, E. (2014). Should Everybody Learn to Code? *Communications of the ACM, 57*(2), 16-18. https://doi.org/10.1145/2557447

Sullivan, F. R. & Heffernan, J. (2016). Robotic Construction Kits as Computational Manipulatives for Learning in the STEM Disciplines. *Journal of Research on Technology in Education*, *48*(2), 105-128. https://doi.org/10.1080/15391523.2016.1146563

Tikva, C. & Tambouris, E. (2021). Mapping Computational Thinking through Programming in K-12 Education: A Conceptual Model Based on a Systematic Literature Review. *Computers & Education*, *162*. https://doi.org/10.1016/j.compedu.2020.104083

Uddannelses- og Forskningsministeriet. (2020, 7. december). *Begreber*. Lokaliseret [25. november, 2021] på https://ufm.dk/uddannelse/anerkendelse-og-dokumentation/dokumentation/kvalifikationsrammer/begreber

Undervisningsministeriet. (2018). *Læseplan for forsøgsfaget teknologiforståelse.* https://www.uvm.dk/-/media/filer/uvm/aktuelt/pdf18/181221-laeseplan-teknologiforstaaelse.pdf

Wang, D., Wang, T. & Liu, Z. (2014). A Tangible Programming Tool for Children to Cultivate Computational Thinking. *The Scientific World Journal*, *2014*. http://dx.doi.org/10.1155/2014/428080

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Werner, L., Denner, J. & Campe, S. (2014). Children Programming Games: A Strategy for Measuring Computational Learning. *ACM Transactions on Computing Education (TOCE)*, *14*(4), 1-22. DOI: 10.1145/2677091

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2011, November 17). Computational Thinking – What and Why. *The Link Magazine*. http://www.cs.cmu.edu/~CompThink/papers/TheLinkWing.pdf