Hvorfor skal eleverne programmere i folkeskolen?

Anita Lykke Clemmensen
Anja Emilie Madsen

Vejleder: Oluf Danielsen
4. semester, Master i IKT og læring

master i ikt og læring

TITELBLAD

Projekttitel:	*Hvorfor skal eleverne programmere i folkeskolen?*
Afleveringsmåned og -år:	*31. Maj 2017*
Antal tegn og sider:	*174.016 (72,5 normalsider)*
Semester:	*4. semester*
Gruppemedlemmer:	*Anita Lykke Clemmensen 20150173, Anja Emilie Madsen 20150270*
Vejlederens navn:	*Oluf Danielsen*

Masterprojektet er skrevet i en kollaborativ proces. Fordelingen af hovedansvaret for de forskellige kapitler ser sådan ud:

Anita: Kapitel 2, 3.4, 5.10 - 5.11

Anja: Kapitel 3.1 - 3.3, 3.5, 4, 5.1 - 5.9

it-vest
samarbejdende universiteter

AALBORG UNIVERSITET

CBS COPENHAGEN BUSINESS SCHOOL HANDELSHØJSKOLEN

UNIVERSITAS ROSKILDENSIS · IN TRANQUILLO MORS · IN FLUCTU VITA

AARHUS UNIVERSITET

ABSTRACT

The purpose of this master thesis is to study skills students in a Danish 6th grade develop while they are programming using the tool Scratch. Programming in elementary schools is currently part of the learning debate in Denmark and other countries due to the focus on 21st century skills and the national curriculum where ICT is a transverse issue in all subjects. In addition, technology is a basic condition of life now and in the future and it requires that students develop skills to use, understand and create technology. Programming can be an approach to developing these skills. To do so, we made research in a 6th grade with 17 students in five sessions. Through interviews, participant observations and video observations, we examined how students develop 21st century skills, computational thinking and digital literacy when programming. The result indicates that students develop skills as problem solving, collaboration and communication as well as concepts and approaches from computational thinking. There are also signs that the students develop digital literacy in the form of being able to handle and create technology through programming. The students also gain insight in what lies behind advanced video games when programming a simple game in Scratch.

FORORD

Dette masterprojekt og udarbejdet på fjerde semester på uddannelsen Master i IKT og Læring på Aalborg Universitet.

Målgruppen for dette projekt er fortrinsvis lærere i folkeskolen, skoleforvaltninger og andre, der er interesserede i spørgsmålet *Hvorfor skal eleverne programmere i folkeskolen?* Vi håber, at projektets resultater kan være med til at besvare spørgsmålet samt bidrage til debatten.

Vi vil gerne takke lærerne fra projekt #ProgrammeringNaturligvis, som undervejs i processen har givet feedback, delt deres viden og dermed inspireret os. De har også givet os et indblik i, hvordan programmering kan anvendes i andre fag end de naturfaglige fag.

En stor tak skal også lyde til læreren og den 6. klasse, som vi har fået lov til at følge. Vi sætter stor pris på deres fleksibilitet samt den værdifulde data, vi har indsamlet til dette projekt.

Vi vil også gerne takke bibliotekarerne på AAU for god hjælp og service i forbindelse med vores litteratursøgning.

Til sidst vil vi gerne takke Oluf Danielsen for vejledning gennem en spændende og lærerig proces.

LÆSEVEJLEDNING

Masterprojektets **kapitel 1** indledes med et overblik over aktualiteten af programmering i folkeskolen og derefter skitseres forskellige problemstillinger, der leder videre til problemformuleringen og tilhørende undersøgelsesspørgsmål. Derefter kommer afsnittene literature review og begrebsafklaring.

I **kapitel 2** redegør vi for forskellen mellem programmering og kodning samt udviklingen af programmeringssprog for derefter at beskrive programmeringsværktøjet Scratch og computational thinking.

Vi beskriver dannelsesbegrebet, digital dannelse samt dannelsespotentialet i programmering i **kapitel 3**, hvorefter vi beskriver begreberne viden, færdigheder og kompetencer inden afsnittet omkring 21st century skills.

Kapitel 4 er teoriafsnittet, hvor vi beskriver forskellige erfaringsbaserede læringsteorier og læringsmodeller, for derefter at uddybe læringsevnerne i Kolbs læringscirkel.

I **kapitel 5** beskriver vi de metoder, vi har anvendt til indsamling af empiri. Derefter kommer et afsnit om selve litteratursøgningsprocessen, inden vi afslutter med en beskrivelse af vores videnskabsteoretiske ståsted.

Kapitel 6 er masterprojektets analyse, hvor empiri analyseres med udgangspunkt i vores undersøgelsesspørgsmål.

I **kapitel 7** konkluderer vi på problemformuleringen *Hvilke kompetencer udvikler eleverne, når de programmerer i Scratch* med afsæt i undersøgelsesspørgsmålene.

Kapitel 8 indeholder perspektiveringen.

HVEM ER VI?

Dette masterprojekt er skrevet i en tværfaglig gruppe bestående af Anita Lykke Clemmensen og Anja Emilie Madsen. Anita er lektor på it-uddannelserne ved University College Nordjylland (UCN), Aalborg, og Anja er pædagogisk it-konsulent i Skoleforvaltningen i Aalborg Kommune. Vi har begge en baggrund som folkeskolelærer og har, udover at undervise eleverne, også haft en vejledningsfunktion på skolen inden for pædagogisk it. Efter 20 år i folkeskolen læste Anita videre til datamatiker og underviser nu kommende datamatikere (og andre) i programmering.

Til fælles har vi også projektet #ProgrammeringNaturligvis (www.programmeringnaturligvis.dk), hvor en gruppe lærere fra skolerne i Aalborg Kommune i perioden 2016 - 2018 bliver opkvalificeret gennem en række moduler på UCN for at blive klædt på til senere at undgå i et programmeringskorps, som alle 55 skoler i Aalborg Kommune kan booke til vejledning, workshops, fagdage og lignende. Anitas funktion er at stå for planlægning af moduler og gennemførelse af undervisning, og Anja har rollen som projektleder. Projekt #ProgrammeringNaturligvis var oplagt at undersøge nærmere i kernemodul 4: IKT og didaktisk design, og vi fik her et indblik i de muligheder, der var for at arbejde videre med emnet i vores Masterprojekt.

INDHOLDSFORTEGNELSE

Titelblad 1
Abstract 2
Forord 2
Læsevejledning 3
Hvem er vi? 4

1 Indledning 7
1.1 Hvorfor skal eleverne programmere i folkeskolen? 9
1.2 Problemstillinger 10
1.3 Problemformulering 13
1.4 Literature review 13

2 Programmering, Scratch og Computational Thinking 18
2.1 Programmering eller kodning? 18
2.2 Programmering og programmeringssprog 19
2.3 Programmering for begyndere 19
2.4 Scratch 20
2.5 Computational Thinking 24

3 Dannelse og kompetencer 28
3.1 Dannelse 29
3.2 Digital dannelse 30
3.3 Har programmering dannelsespotentiale? 32
3.4 Viden, færdigheder og kompetencer 33
3.5 21st century skills 42

4 Teoriafsnittet 48
4.1 Erfaringsbaserede læringsteorier 49
4.2 Hvad karakteriserer erfaringsbaseret læring? 50
4.3 Kolbs læringscirkel 50

5 Metode og empiri 53
5.1 Empiri 53
5.2 Etiske overvejelser 57
5.3 Videoobservation 57
5.4 Deltagerobservation 58
5.5 Interview 59
5.6 Eliteinterview 60
5.7 Opsamling på metoderne 60
5.8 Grounded theory 60
5.9 Litteratursøgning 64
5.10 Metodekritik 64
5.11 Videnskabsteori 67

6 Analyse 69
6.1 What's going on? 70
6.2 I hvilket omfang udvikler eleverne 21st century skills? 74
6.3 I hvilket omfang udvikler eleverne computational thinking? 80
6.4 I hvilket omfang udvikler eleverne digital dannelse? 85
7 Konklusion 86
8 Perspektivering 87
Litteraturliste 90
Figurliste 100
Bilag 103

1 INDLEDNING

Børn og unge i Danmark har i dag gode chancer for at møde programmering i både skole og i deres fritid. Programmering er nemlig kommet på den pædagogiske dagsorden, og flere skoler har allerede sat programmering på skoleskemaet eller overvejer, om - eller hvordan - de skal gøre det.

I fritidsdelen tilbyder den frivillige forening Coding Pirates (Coding Pirates, n.d.), at børn og unge kan mødes til klubaftener og workshops og få styrket deres produktive og kreative it-kompetencer gennem programmering. Coding Pirates blev grundlagt i 2014 og har i skrivende stund over 1400 medlemmer fordelt på 35+ lokale afdelinger rundt om i landet samt over 2000 børn og unge på venteliste (bilag 1). Derudover har IT-branchen, i samarbejde med Coding Pirates, Styrelsen for IT og Læring (STIL) samt skoler fra to kommuner, sat faget *Coding Class* på skoleskemaet som valgfag, med et mål om "at få børn til at tænde på it og teknologi, så de får en bedre forståelse for den verden, der omgiver dem nu og i fremtiden" (IT-Branchen, n.d.).

I folkeskolen er programmering nævnt få steder i Fælles Mål i faget fysik/kemi i udskolingen samt enkelte steder i matematik og natur/teknologi (EMU, n.d.-d). For IT-branchen er dette ikke tilstrækkeligt og argumenterer for, at kodning skal indføres som et selvstændigt fag (IT-Branchen, n.d.), således at de nye generationer får en grundlæggende forståelse af, hvordan teknologien virker og påvirker os, så de får en mere aktiv og producerende rolle i stedet for en passiv og konsumerende brug af teknologi (IT-Branchen, 2017a). På baggrund af erfaringer fra pilotprojektet Coding Class, sendte IT-branchen et nyt it-fag *Computational Thinking* i høring d. 1. februar 2017, hvor politikere og forskere debatterede, hvorvidt der var behov for et nyt it-fag i folkeskolen (IT-Branchen, 2017b). I december 2016 udgiver Danmarks Vækstråd også en rapport, hvori én af "her og nu" anbefalingerne lyder "'Computational Thinking' gøres til en integreret og obligatorisk del af undervisningen på relevante uddannelser"(Danmarks Vækstråd, 2016, p. 6) herunder også folkeskolen. Ideen om computational thinking kommer fra datalogien (Wing, 2006) og handler om at opnå en bestemt type problemløsende kompetence, som kan anvendes i alle tænkelige hverdagssituationer (Wing, 2008). Den 31. marts 2017 offentliggjorde undervisningsministeren, Merete Riisager, som også deltog i IT-Branchens høring om et nyt it-fag i folkeskolen, at hun har nedsat en rådgivningsgruppe for digital læring. Rådgivningsgruppen skal bl.a. sparre med undervisningsministeren om en langsigtet handleplan for digital læring samt komme med input til udviklingen af et nyt valgfag i udskolingen, som har arbejdstitlen *Teknologiforståelse* (UVM, 2017c). Målet er "at udvikle Danmarks styrkeposition i forhold til anvendelse af it i undervisningen og styrke danske børn, unge og voksnes digitale kompetencer på alle uddannelsesniveauer" og Riisager fortsætter: "Folkeskolen skal ikke uddanne brugere af it. Folkeskolen skal uddanne skabere og analytikere med it, der hvor det giver fagligt mening" (UVM, 2017c).

HVAD ER DET NYE?

Debatten om indførelsen af et it-fag er dog ikke ny i folkeskolen. Helt tilbage i starten af 70'erne blev faget *datalære* indført som valghold i udskolingen, selvom den såkaldte EDB-betænkning anbefalede at gøre faget obligatorisk på alle klassetrin (Johnsen et al., 1972). Op gennem 80'erne og 90'erne bølgede det frem og tilbage, hvorvidt datalære skulle indføres som et obligatorisk fag, men det blev aldrig til mere end et valgfag og senere til et obligatorisk emne, som ingen lærere følte sig ansvarlige for (Hvid, 2001). Hvorvidt der lå pædagogiske eller økonomiske hensyn bag afskaffelsen af faget i 1990, stillede den daværende it-vejlederforening spørgsmålstegn ved (Hvid, 2001).

Til forskel fra 70'erne, hvor it i undervisningen var udfordret pga. manglende eller dyrt udstyr, har skolerne nu alle muligheder for at anvende og undervise i it i folkeskolen. I 2014 blev Danmark kåret som det mest it-parate land ud fra en samlet score på adgangen til it, brugen af it og it-kompetencer (ITU, 2014). Ifølge Kommunernes Landsforening (KL) har staten og kommunerne planlagt eller gennemført it-investeringer for to milliarder på folkeskoleområdet i perioden 2012 – 2017 (Kudahl, 2015). Derudover er der let adgang til adskillige gratis læringsressourcer fx Swift Playgrounds, Hour of Code og Scratch (Apple, n.d.; Hour of Code, n.d.; MIT Media Lab, n.d.), som børn og unge kan anvende, når de skal lære at programmere.

INFORMATIK I GYMNASIET

Men det er ikke kun i folkeskolen, der er fokus på programmering. I gymnasiet vil de studerende fremover møde det obligatoriske it-fag *Informatik*, der er indført i forbindelse med gymnasiereformen 2016 (UVM, 2016), efter at have været et forsøgsfag med titlen Informationsteknologi siden 2010 (Møller, Tosca, Husfeldt, & Thomassen, 2014). Informatik beskrives som et moderne, almendannende og studieforberedende it-fag, og i forsøgslæreplanen fremgår det, at programmering også er centralt i faget, da eleverne skal kunne "identificere basale strukturer i programmeringssprog, udarbejde it-produkter i form af simple programmer og tilpasse eksisterende programmer" (UVM, 2010b). Evalueringen af forsøgsfaget Informationsteknologi viste dog, at underviserne havde svært ved at udvikle elevernes skaberkompetencer, da eleverne i gymnasiet manglede basale it-brugerkompetencer og ligeledes digital dannelse (Møller et al., 2014, p. 37).

Ved Aarhus Universitet er Center for Computational thinking (CCT) oprettet med støtte fra IT-Vest. CCT står bag udviklingen af gymnasiets nye fag informatik, men CCT har også fokus på at brede computational thinking ud i andre fag i gymnasiet samt i folkeskolen (Center for Computational Thinking, 2017). CCT's udviklingschef, datalog og ph.d. Michael E. Caspersen, var derfor også en central person til høringen om et nyt it-fag i folkeskolen, og han er også formand for

skrivegruppen, der skal udvikle mål til folkeskolens nye valgfag *Teknologiforståelse* (IT-Branchen, 2017b) (UVM, 2017c).

HVAD SKER DER INTERNATIONALT?

Når vi vender blikket mod andre lande i Europa, er programmering også aktuelt på folkeskoleniveau. I England har programmering været en del af deres læseplaner siden 2013 (Department for Education, 2013), og i Estland starter eleverne med at kode fra 1. klasse (Exner, 2012; Hitsa, n.d.). I efteråret 2016 blev programmering også indført i Finlands nationale læseplan som en obligatorisk og tværfaglig aktivitet (Toikkanen, 2015). Lignende tendenser ses også uden for Europa i bl.a. Australien, Singapore, Israel og Sydkorea (Wing, 2014).

1.1 HVORFOR SKAL ELEVERNE PROGRAMMERE I FOLKESKOLEN?

Der er altså ingen tvivl om, hvorvidt programmering er aktuel i den uddannelsespolitiske debat, men hvis vi stiller det simple spørgsmål: *Hvorfor skal eleverne programmere i folkeskolen?* vil vi møde forskellige svar og argumenter, alt efter hvem vi spørger.

Regeringen understreger i deres regeringsgrundlag fra 2016, at der skal uddannes flere ”dygtige it-specialister og avancerede it-brugere til det fremtidige arbejdsmarked” (Regeringen, 2016, p. 82). I regeringsgrundlaget er der også et særligt fokus på, at fremme kvindernes interesse for it og programmering.

Erhvervslivet efterspørger også flere uddannede it-medarbejdere, der kan programmere, og de videregående it-uddannelser efterspørger flere ansøgere til deres it-uddannelser og ikke mindst kvinder, der i 2016 kun udgjorde 26 % af de optagne på it-uddannelserne (UFM, 2016). Dette tal skal ses i lyset af, at der er en bred definition af it-uddannelser (UFM, 2016), som kan have relation til fx ledelse, design, sundhed, musik og selvfølgelig mere tekniske fagområder som hardware og software (“Future People,” n.d.). Ofte tiltrækkes kvinderne af de mere ”bløde” it-uddannelser (Behrendt, 2017). På Københavns Universitet blev der fx kun optaget 7,65 % kvinder på datalogistudiet i 2016, mens der på it og sundhedsstudiet blev optaget 42,22% (“Antal optagne fordelt på køn,” 2016).

De gymnasiale ungdomsuddannelser har til formål at forberede de studerende til videregående uddannelse og begrunder deres nye almendannende fag Informatik med, at det bl.a. skal udvikle elevernes ”evne til at forholde sig til den enkeltes, uddannelsens og samfundets brug af it gennem teoretisk indsigt i og praktisk arbejde med at skabe it-produkter” (UVM, 2010b). Faget er dog udfordret af, at deres elever, som jo kommer direkte fra folkeskolen, mangler grundlæggende it-kompetencer og digital dannelse (Møller et al., 2014, p. 37).

Ifølge undervisningsministeren, Merete Riisager, er det vigtigt, at folkeskolen udvikler elevernes it-skabende kompetencer samt deres forståelse for digital teknologi, således at Danmarks position ift. anvendelsen af it i undervisningen står stærkt (UVM, 2017c).

Regeringen, erhvervslivet og de videregående it-uddannelser har altså et særligt sigte på fremtidens arbejdsmarked og vækst, hvis de skulle svare på spørgsmålet: *Hvorfor skal eleverne programmere i folkeskolen?* De gymnasiale ungdomsuddannelser har brug for, at deres elever i højere grad udvikler it-kompetencer og digital dannelse i folkeskolen, så de parate til gymnasiets nye fag informatik. Hvis undervisningsminister Merete Riisager skulle svare på spørgsmålet *Hvorfor skal eleverne programmere i folkeskolen?* kan hendes udtalelse: ”Folkeskolen skal ikke uddanne brugere af it. Folkeskolen skal uddanne skabere og analytikere med it” (UVM, 2017c) måske være svaret.

1.2 PROBLEMSTILLINGER

IT ER ET GRUNDVILKÅR

Teknologien er omkring os alle vegne - både i skole, på arbejde og i vores fritid. Danmark er ét af de mest it-parate lande, og der stilles krav om, at vi alle kan anvende digitale systemer, når vi fx skal booke tid ved lægen, betale kontingent via netbank, tjekke lønsedlen og bestille plads i dagplejen. It er dermed et grundvilkår, som vi ikke kan fravælge. Professor Ole Sejer Iversen mener, at undervisningen i it skal starte i folkeskolen, som har til opgave at favne og uddanne alle elever til dette grundvilkår (Iversen, 2017a). Iversen udtaler, at it vedrører os alle, og at it ikke kun er et anliggende for programmører. Derfor skal alle børn tilbydes en stærk uddannelse i it, og Iversen afslutter artiklen således ”Lad os i fællesskab forfølge ambitionen om at uddanne danskerne til de mest IT-kompetente og IT-kritiske borgere i verden” (Iversen, 2017a).

Ph.d. - studerende Jesper Balslev er dog mere kritisk over for digitaliseringen af folkeskolen og mener ikke, at der er evidens for, at it i undervisningen virker set ud fra elevernes faglige resultater. OECD-rapporten fra 2015 viser nemlig, at de lande, som har investeret mest i digitalisering, har et fald i elevernes faglige resultater (Bach, 2017). Balslev mener, at eleverne bedre kan tilegne sig viden gennem bøger og håndskrevne notater, da it distraherer og svækker elevernes koncentration (Bach, 2017). Han er også kritisk over for programmering, for hvis alting forandrer sig, hvordan kan vi så vide, om de programmeringssprog, vi lærer lige nu kan bruges om 10 år – og har vi overhovedet behov for at programmere i fremtiden? Balslev afslutter ”Det ville give meget mere mening, at lære børnene mere matematik” (Bach, 2017), for der vil altid være behov for at kunne tænke logisk.

Vi har altså to lejre, hvor den største består af fortalere, der ser it som et grundvilkår og dermed vil have mere it i undervisningen. I den lidt mindre lejr har vi fx Balslev, som mener, at digitaliseringen af folkeskolen er tvivlsom set ud fra elevernes faglige testresultater. Vi mener dog ikke, at formålet med it i undervisningen alene skal skabe elever, som får bedre testresultater i fx PISA og nationale test. De it-kompetencer, som eleverne skal udvikle, er mere komplekse og dermed sværere at måle i digitale og selvrettende test end fx læse- og matematikfærdigheder. Iversen stiller derfor krav om, at der udvikles nye prøveformer i form af et eksamensfag i kreative it-kompetencer, "der stiller skarpt på børnenes evne til at arbejde kreativt og anvende teknologien som materiale til at finde innovative løsninger på komplekse problemstillinger" (Iversen, 2017a).

Iversen (2017b) taler om nødvendigheden af digital dannelse gennem udvikling af kompetencer, så alle i de kommende generationer kan blive både kritiske og innovative digitale designere i en verden, der bliver stadig mere digitaliseret. Professor Eevi Beck (2011) argumenterer også for nødvendigheden af at inddrage dannelsesaspektet i forhold til inddragelse af teknologi i skolen, så man er i stand til at forholde sig teknologien. Dette fører os over i en anden debat omkring 21st century skills, som kan oversættes til en række færdigheder og kompetencer, den enkelte skal besidde for at lykkes i det 21. århundrede. Betegnelsen 21st century skills har stor politisk bevågenhed i det amerikanske uddannelsessystem, og det begynder også at få en større fylde i folkeskolen (Berthelsen, 2017). Mitch Resnick, som har været med til at udvikle programmeringsværktøjet Scratch, fortæller, at når eleverne programmerer, udvikler de automatisk kompetencer til kollaboration, kreativitet, problemløsning og design, hvilket af flere bliver betegnet som vigtige kompetencer i det 21. århundrede (Resnick, 2012). Resnick er også enig med Iversen i, at programmering ikke kun er forbeholdt programmørerne, men at programmering er for alle. Eleverne skal lære både at læse og skrive i skolen, på trods af at mange ikke skal være journalister eller forfattere, når de bliver voksne. På samme måde mener Resnick også, at eleverne både skal lære at anvende og producere teknologi, selvom det ikke er alle, der skal uddannes som programmører og dataloger (Resnick, 2012).

HVEM SÆTTER DAGSORDENEN?

På forholdsvis kort tid har debatten om programmering i folkeskolen fået et stort fokus fra forskellige interessenter, og nogle af de store stemmer er bl.a. det privat erhvervsliv, som mangler it-medarbejdere. Ifølge folkeskolens formålsparagraf er det folkeskolens opgave at "give eleverne kundskaber og færdigheder, der forbereder dem til videre uddannelse" (Ministeriet for Børn Undervisning og Ligestilling, 2016) og dermed ikke at gøre dem direkte klar til arbejdsmarkedet. Dog åbner Undervisningsministeriet op for, at folkeskolen i højere grad skal bidrage til udviklingen af eleverne skabende it-kompetencer med nye tiltag som strategiplan for digital læring og det nye teknologivalgfag. Dette mener Iversen også er vigtigt, "hvis folkeskolen skal forblive relevant som uddannelsesinstitution" (Iversen, 2017a).

Fra at tale helt konkret om, at eleverne skal programmere eller kode i folkeskolen, ser vi nu en tendens til, at programmering og kodning skal gemmes længere væk bag fagbetegnelser som *Informatik, Teknologiforståelse* og *Computational thinking.* Grunden til denne udvikling kan være, at ordene programmering og kodning kan give associationer til støvede computere med især mandlige nørder ved tastaturet, hvilket ikke leder tankerne hen på kreative, innovative og kollaborative kompetencer. Det kan være nemmere at acceptere de brede betegnelser som *Informatik, Teknologiforståelse* og *Computational thinking* i en folkeskolekontekst, selvom ingen helt har en fornemmelse eller forståelse af, hvad betegnelserne rent faktisk dækker.

HVAD OG HVOR MEGET SKAL DER TIL?

Der er altså flere, der stemmer for indførelsen af et nyt it-fag i folkeskolen, hvor eleverne bl.a. skal lære at programmere. Deriblandt kan vi nævne Siri-kommissionen, Digital Vækstpanel, Iversen og It-branchen (Digital Vækstpanel, 2017; IT-Branchen, 2017b; Iversen, 2017a, 2017b; Siri-kommissionen, 2016). Undervisningsministeriet har været mere forsigtige og har taget initiativ til oprettelse af valgfaget *Teknologiforståelse* til udskolingens elever (UVM, 2017c), som er et treårigt udviklingsforsøg (bilag 2). Men er det nok at tilbyde valgfaget *Teknologiforståelse* til interesserede elever i udskolingen, hvis alle elever skal uddannes til at blive de "mest it-kompetente og it-kritiske borgere i verden" (Iversen, 2017a) som Iversen har ambitioner om?

Undervisningsministeren udtaler, at "Folkeskolen skal uddanne skabere og analytikere med it, der hvor det giver fagligt mening" (UVM, 2017c), hvilket peger i retning af, at de it-skabende kompetencer skal udvikles i eksisterende fag. Dette har også været sigtet med både faghæfte 48 *It- og mediekompetencer i folkeskolen* fra 2009 og nu Fælles Mål, som har det tværgående emne It og medier (EMU, 2014b; UVM, 2010a, 2017a), men det har, set ud fra den nuværende debat, ikke været nok.

Fælles Mål er nationale mål, der beskriver, hvad eleverne skal lære i folkeskolens forskellige fag. Fælles mål er bindende og styrende for undervisningen og kunne dermed være et oplagt sted at starte, hvis programmering skal fylde mere i folkeskolen. Brian Ravnborg Christensen, der er CFU-konsulent og har været med til at skrive de Fælles Mål i naturfagene, ser gerne at programmering kommer til at fylde mere i Fælles Mål. Især i faget matematik, som har et stort overlap med programmering (bilag 3). Ravnborg udtaler, at inden programmering indføres for alvor i folkeskolen "skal man sikre, at der også er nogle, som kan undervise i det, og som ikke kun ser programmering som en færdighed, men som en kompetence. Derfor skal programmering indføres på læreruddannelsen, inden vi sætter det på skemaet i grundskolen" (bilag 3). Iversen er enig med Ravnborg og udtaler "Folkeskolen anno 2017 har for få undervisere med den fornødne ekspertise til at kunne tilbyde IT-undervisning i verdensklasse" (Iversen, 2017a), samt at der skal investeres i uddannelsen af underviserne, som er den vigtigste forudsætning for elevernes læring (Iversen, 2017a).

1.3 PROBLEMFORMULERING

Ovenstående problemfelter leder os frem til nedenstående problemformulering:

Hvilke kompetencer udvikler eleverne, når de programmerer i scratch?

UNDERSØGELSESSPØRGSMÅL

- *I hvilket omfang udvikler eleverne 21st century skills, når de programmerer i Scratch?*
- *I hvilket omfang udvikler eleverne computational thinking, når de programmerer i Scratch?*
- *I hvilket omfang udvikler eleverne digital dannelse, når de programmerer i Scratch?*

1.4 LITERATURE REVIEW

Forud for skrivningen af dette literature review har vi været igennem en systematisk litteratursøgning, som beskrives nærmere i metodeafsnittet. For overskuelighedens skyld har vi opdelt vores literature review i to områder:

PROGRAMMERING, COMPUTATIONAL THINKING OG SCRATCH I FOLKESKOLEN

I forhold til titlen på vores projekt *Hvorfor skal eleverne programmere i folkeskolen?* var det i vores søgning interessant at undersøge i hvilket omfang, der i litteraturen er skrevet om programmering/kodning på folkeskoleniveau både i Danmark og internationalt. I samme ombæring var det interessant at dykke ned i viden om og erfaringer fra litteraturen omkring computational thinking og Scratch i folkeskolen eller tilsvarende. Computational thinking er et begreb, der særligt på det seneste er dukket op i den danske debat om programmering i folkeskolen, derfor er begrebet også med i søgningen. Når vi også dykker ned i Scratch, er det fordi vores empiri indsamles i en klasse, der skal programmere i Scratch, derfor er det interessant at se hvilke erfaringer, der er med programmering i Scratch på folkeskoleniveau.

Efter en del søgning på ordene programmering/kodning og folkeskole (og lignende begreber på engelsk) var det noget begrænset, hvad der dukkede op af materiale. Først da vi begyndte at inddrage ordene Scratch og især Computational Thinking, dukkede der for alvor artikler op, der omhandle programmering på folkeskoleniveau.

De første elektroniske computere udvikles i 40'erne. De første programmeringssprog dukker op i 50'erne, og i 60'erne opstår videnskaben computer science, der på dansk kendes som datalogi (Shallit, 1995). Til trods for at faget er ret nyt, bliver der allerede i 60'erne talt om at brede

programmering ud til andre end matematikere og programmører, som det nævnes i artiklen *Education Paving the way for computational thinking* (Guzdial, 2008). I 70'erne udgiver Professor Seymour Papert og Dr. Cynthia Solomon artiklen *Twenty things to do with a computer* (Papert & Solomon, 1972), hvor de argumenterer for, at alle – også børn – vil have glæde af at lære at programmere, og at det kan anvendes på andre områder end matematik, fx musik. I 1972 udkom *Betænkning om EDB-undervisning i det Offentlige Uddannelsessystem* (Johnsen et al., 1972), som et resultat af arbejdet i Johnsen-udvalget, som den daværende undervisningsminister havde nedsat i 1970. EDB-betænkningen anbefalede, at der blev oprette et valgfag, datalære, for de ældste elever i folkeskolen. Det blev også anbefalet, at det skulle undersøges nærmere, om datalære skulle gøres obligatorisk, da faget vil fremme elevernes almene dannelse. I 1980 udgiver Papert bogen *Mindstorms: Children, Computers, and Powerful Ideas* (Papert, 1980), der tager udgangspunkt i programmeringssproget Logo, som er udviklet til børn. I bogen anbefaler og anviser Papert programmering for børn. Udtrykket computational thinking (Papert, 1980, p. 182) bliver nævnt første gang i denne bog, dog ikke som kompetence.

I 80'erne er der ikke mange artikler om programmering for børn og i 90'erne forsvinder faget datalære helt i Danmark jf. artiklen *Datalære - faget, som forsvandt* (Hvid, 2001). Det samme skete globalt, som det nævnes i artiklen *Code to learn with Scratch? A systematic literature review* (Moreno-Leon & Robles, 2016).

I 2006 publicerede Jeannette M. Wing, dengang professor i Computer Science og nu Corporate vice president for Microsoft research, sin artikel *Computational thinking* (Wing, 2006), hvor hun introducerede begrebet computational thinking første gang. Siden har hun udredt begrebet i flere artikler: *Computational thinking and thinking about computing* samt *Computational Thinking: What and Why?* og blogposts: *Computational Thinking Benefits Society* samt *Computational thinking, 10 years later* (Wing, 2008, 2010, 2014, 2016). Programmeringssproget Scratch blev udgivet i 2007 og er med til at understøtte computational thinking, forklarede Resnick m.fl. (2009) i deres artikel *Scratch: Programming for All*.

Både Scratch og computational thinking er i 10'erne blevet undersøgt af flere. Der er adskillige artikler om Scratch fx *Programming Concepts in Playful Programming Products* (Allsopp & Ejsing-Duun, 2016), *From Scratch to "Real" Programming* (Meerbaum-Salant, Armoni, & Ben-Ari, 2013) og *Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools* (Sáez-López, Román-González, & Vázquez-Cano, 2016), som undersøger, hvilke udbytter, der kommer af at programmere i Scratch. Der er dog ikke mange, der undersøger, hvad der sker, når børnene programmerer, og i *Review on teaching and learning of computational thinking through programming: What is next for K-12?* (Lye, Hwee, & Koh, 2014) anbefales det, at fremtidige studier har fokus på selve programmeringsprocessen, når børn programmerer.

Programmering og computational thinking er i 10'erne igen kommet med i læseplaner i både Danmark og internationalt (Department for Education, 2013; Hitsa, n.d.; Toikkanen, 2015; UVM, 2017a; Wing, 2014, 2016) med forskellig tyngde.

Mens vi har skrevet vores masterprojekt, har der været mange kritiske og diskuterende indlæg og artikler om programmering i folkeskolen (Bach, 2017; IT-Branchen, 2017a, 2017b, Iversen, 2017a, 2017b; Riise, 2017; UVM, 2017c). Vi har undervejs læst, taget stilling til og inddraget flere af disse i vores projekt, da indlæggene og artiklerne har talt direkte ind i vores projekt.

21ST CENTURY SKILLS, PROGRAMMERING OG DIGITAL DANNELSE I FOLKESKOLEN

I litteraturen møder vi begrebet 21st century skills i starten af 00'erne med først OECD's Definition and Selection of Competencies (DeSeCo). Det internationale DeSeCo-projektet har fokus på både unge og voksne og definerer fremtidens nøglekompetencer i tre brede kategorier (OECD, 2003). Disse kategorier har dog ikke et særligt pædagogisk sigte ift. elever i folkeskolen. Dette sigte har Partnership for 21st Century Learning (P21) til gengæld, der i 2002 udgiver rapporten *Learning for the 21st Century – A Report and Mile Guide for 21st Century Skills* (P21, 2002). P21 udvikler en model for 21st Century Learning (P21, 2015), som er målrettet K-12 området, hvilket svarer til den danske folkeskole. ITL Research kommer i 2011 også med et bud på, hvordan 21st century skills kan operationaliseres i folkeskolen med *21st Century Learning Rubrics* (ITL Research, 2014). Alle disse forskellige definitioner og modeller kalder også på en række reviews, som sammenligner og forholder sig kritisk til 21st century skills. Artiklen *Comparing Frameworks for "21st Century Skills"* fra 2009, sammenligner syv modeller og konkluderer, at der er en række fællestræk i modellerne, samt at de bygger oven på hinandens definitioner (Dede, 2009). I 2013 skriver Kereluik m.fl. (2013) et kritisk review af 21st century knowledge frameworks med et særligt fokus på betydningen for læreren og læreruddannelsen. Konklusionen er i stil med Dedes artikel (Dede, 2009), at 21st century skills bliver defineret forskelligt, men at der samtidig er en række overlapninger. Adjunkt ved Aarhus Universitet, Ulf Dalvad Berthelsen, introducerer begrebet det 21. århundredes kompetencer i artiklen *21st Century Skills – om det 21. århundredes kompetencer – fra arbejdsmarkedspolitik til allemandseje* (Berthelsen, 2017). Berthelsen problematiserer bl.a. at begrebet stammer fra arbejdsmarkedet og derfra presses ind i den uddannelsespolitiske dagsorden. Sirikommissionens rapport *Kunstig intelligens – morgendagens Job og Samfund* fra 2016 peger helt konkret på, at uddannelsessystemet skal rettes mod de 21st century skills, samt at programmering skal være et hovedfag i folkeskolen (Siri-kommissionen, 2016). De to tværgående emner It og medier samt Innovation og entreprenørskab er det tætteste, vi kommer på en definition af 21st century Skills i dansk folkeskoleskolekontekst, når vi søger efter spor af 21st century skills i Fælles Mål på folkeskoleområdet.

Når vi søger videre på 21st century skills og programmering, finder vi TED talken *Let's teach kids to code* med Mitch Resnick fra MIT Media Lab (Resnick, 2012). Resnick fortæller, at eleverne helt

automatisk udvikler 21st century skills, når de programmerer i Scratch. I artiklen *Digital Storytelling with Scratch in Middle School Classrooms* (Burke & Kafai, 2011), er der også sammenhæng mellem programmering og udviklingen af problemløsende kompetencer.

Under vores søgninger på 21st Century skills og dannelse kommer vi frem til, at det danske ord "dannelse" ikke direkte kan oversættes til et tilsvarende begreb på engelsk. I Nordic Journal of Digital Literacy, finder vi Eevi E. Becks artikel *Computers in Education: What for?* (Beck, 2011), hvor dannelse bliver bragt på banen i debatten om højere formål med brugen af it i undervisningen. Jeppe Bundsgaard udgiver i 2017 bogen *Digital dannelse,* som handler om, hvordan eleverne i skolen skal lære at håndtere, anvende og udvikle digitale teknologier. Bundsgaard beskriver også, hvordan computational thinking falder ind under den digitale dannelse.

Opsummering

I forhold til ovenstående literature review placerer vores masterprojekt sig på folkeskoleområdet, hvor vi undersøger, hvorfor eleverne skal programmere i folkeskolen, ved at have fokus på selve programmeringsprocessen og ikke resultatet. Vi ønsker at afdække, hvilke kompetencer eleverne på 6. årgang udvikler, når de blokprogrammerer i Scratch. Vores fokus er på kompetencer fra 21st century skills samt computational thinking. Vi har også fokus på, om eleverne udvikler digital dannelse, når de programmerer.

BEGREBSAFKLARING

21st century skills: 21st century skills kan oversættes til det 21. århundredes kompetencer. OECD definerer 21st century skills som "those skills and competencies young people will be required to have in order to be effective workers and citizens in the knowledge society of the 21st century" (Claro & Ananiadou, 2009, p. 8).

Blokprogrammering: Kodeblokke, hvor en kodestruktur er grupperet sammen i en blok, som har en form, der sikrer, at den kun kan sammensættes med en anden blok, hvor den passer ind i forhold til den bagvedliggende kode og syntaks (Resnick et al., 2009).

Computational thinking: En problemløsende kompetence, hvor problemer i alle sammenhænge, kan løses ved at tænke som en datalog.

Computer science: Datalogi

Computer scientist: Datalog

Dannelse: Dannelse er en "betegnelse for dels en pædagogisk norm ved valg af indhold i opdragelse og undervisning, dels en social norm, der udpeger en bestemt adfærd, væremåde, opførsel og viden som dannet" (Winther-Jensen, n.d.).

Digital dannelse: Ifølge Jeppe Bundsgaard (1970), professor i it og fagdidaktik, handler begrebet digital dannelse:

> både om at kunne anvende og opføre sig ordentligt med teknologi, erkende teknologiens rolle i vores fælles liv, forholde sig til udfordringer og deltage engageret i at forstå og handle i forhold til de muligheder og udfordringer teknologier giver for os i vores fællesskaber, i vores samfund og som individer. (Bundsgaard, 2017, p. 12)

EDB: Forkortelse for Elektronisk Databehandling, der beskrives som "bearbejdelse af digitale data i en datamaskine og det udstyr, der bruges hertil" ("EDB," n.d.).

Informationsteknologi/It: Informationsteknologi defineres som "digital teknologi, der omfatter indsamling, lagring, behandling, overførelse og præsentation af information" ("Informationsteknologi," n.d.). Informationsteknologi bliver forkortet således: *It.*

K-12: Forkortelsen K-12 betyder "Kindergarten through twelfth grade" (Berthelsen, 2017) og anvendes i bl.a. det amerikanske skolesystem for det område, der aldersmæssigt svarer til folkeskolens elever.

Kodning: Verbet *at kode* betyder "på forhånd fastlægge eller give anlæg for visse egenskaber eller en bestemt beskaffenhed ved individer eller ting" ("Kode, 3," n.d.) og er synonym med at *programmere*. Kodning er dermed synonym med programmering.

Kompetencer: "drejer sig om at være i stand til at handle i relation til bestemte kendte, ukendte og uforudsigelige situationer" (Illeris, 2012, p. 35).

Programmering: "udarbejdelse af anvisninger til hvordan en maskine, typisk en computer, skal udføre bestemte funktioner" ("Programmering," n.d.).

Skills: Det engelske ord *skills* dækker bredt herunder evner, færdigheder, kvalifikationer, kompetencer mm. ("Skills," n.d.).

STEAM: STEM + Art ("STEM to STEAM," n.d.).

STEM: Naturfaglige fag: Science, Technology, Engineering and Math (Rouse, 2013).

Teknologi: Teknologi beskrives som "læren om og den praktiske brug af maskiner, computere og lignende, der gør det muligt at udføre bestemte funktioner" ("Teknologi," n.d.). Flere steder bliver begrebet teknologi dog brugt med samme forståelse som informationsteknologi. I vores

masterprojekt bruger vi informationsteknologi, it og teknologi med den samme forståelse som informationsteknologi.

2 PROGRAMMERING, SCRATCH OG COMPUTATIONAL THINKING

I dette afsnit vil vi indkredse begrebet programmering og vil i den forbindelse også afdække begrebet kodning, da programmering og kodning ofte nævnes i samme forbindelse. Vi beskriver programmeringsværktøjet Scratch samt programmeringssproget Scratch. Sidst i kapitlet vil vi gå tæt på kompetencen computational thinking, som i særlig grad italesættes, når man diskutere programmering og børn.

2.1 PROGRAMMERING ELLER KODNING?

I mange år har der været diskussioner om forskellen på kodning og programmering (Bensav, 2016; Danyl, 2011; Destron, 2006; Prottsman, 2015; Shawwa, 2011). Både ordet programmering og ordet kodning benyttes ofte i flæng, når der tales om at skrive softwareprogrammer, og når programmering debatteres i folkeskolekontekst.

I Den Danske Ordbog har *programmering* to betydninger. I den ene betyder programmering ”udarbejdelse af anvisninger til hvordan en maskine, typisk en computer, skal udføre bestemte funktioner” (“Programmering,” n.d.), og ordet har været kendt siden 1958 (“Programmering,” n.d.). Ordet kode stammer fra det latinske ord *codex*[1]. Substantivet *kode* har fem betydninger og beskrives bl.a. som ”(kombination af) tegn i et edb-system, som fx sætter bestemte procedurer i gang eller fastlægger skriftstørrelse el. lign. i et tekstbehandlingsprogram; sæt af regler for hvordan tegn i et tegnsæt repræsenterer data” (“Kode, 1,” n.d.). Verbet *at kode* har to betydninger, og den mest nærliggende lyder: ”på forhånd fastlægge eller give anlæg for visse egenskaber eller en bestemt beskaffenhed ved individer eller ting” (“Kode, 3,” n.d.) og er synonym med at *programmere*. *Kode* har altså en bredere betydning end det at *programmere*. Efter udbredelsen af hjemmesider som Hour of Code (Hour of Code, n.d.) og foreninger som Coding Pirates (Coding Pirates, n.d.) o.l., bliver ordet *kodning* ofte anvendt med samme betydning som *programmering*.

[1] ”codex oprindelig 'træblok, bræt, skrivetavle af træ'”(“Kodeks,” n.d.)

Derfor vil der, særligt når der tales om programmering for børn, ikke være den store forskel på betydningen af ordene. Vi vil dog benytte os af ordet programmering, når det drejer sig om processen, mens at det programmerede (produktet) vil blive betegnet som kode.

2.2 PROGRAMMERING OG PROGRAMMERINGSSPROG

Computere er baseret på den abstrakte matematiske model for Turingmaskinen[2] (Hazewinkel, n.d.) som blev udviklet i 30'erne. De første computere blev hovedsageligt programmeret af matematikere, der kunne omskrive et problem til en matematisk repræsentation, som kunne eksekveres direkte af en computer (Parker & Davey, 2012). Det er dog først i begyndelsen af 50'erne, at vi ser de første programmeringssprog, som har symbolsk sammenhæng, men ikke har matematisk tilsnit (Hagen, 2006, p. 159). Udviklingen af programmeringssprog gik dog forholdsvis hurtigt, og allerede i slutningen af 1950'erne ser de første matematisk orienterede, funktionelle og deklarative programmeringssprog dagens lys (Hagen, 2006, p. 159). I løbet af 70'erne og 80'erne blev de personlige computere udviklet (Resnick et al., 2009) og medier, som fx tekst, billeder og lyd, begyndte at blive en del af computerne (Hagen, 2006, p. 160). Dette førte til udviklingen af objektorienterede programmeringssprog (Hagen, 2006, p. 160) fx Java, C++, C# og PHP, som i dag er nogle af de mest anvendte programmeringssprog ("TIOBE Index," n.d.).
I modsætning til de første sprog, som nærmest må betegnes som maskinsprog ("0" og "1"), er objektorienterede programmeringssprog sprog, der er udviklet og designet til at ligne naturlige sprog (Guzdial, 2008, p. 26). Alle programmeringssprog bygger på tidligere sprog og kan historisk føres tilbage til de første programmeringssprog[3]. Sprogene har hver især deres egen syntaks, som er "den korrekte notation af de enkelte bestanddele i et program, skrevet i vedkommende sprog" (Hansen, n.d.). Når programmer er skrevet i disse sprog, bliver sproget compiled (oversat) til maskinsprog. Compileren vil i den forbindelse tjekke syntaksen og melde fejl, såfremt der er noget galt med syntaksen. Det er programmørens opgave bl.a. gennem forskellige test, at sikre semantikken, som "er meningen med de forskellige syntaktiske konstruktioner"(Hansen, n.d.).

2.3 PROGRAMMERING FOR BEGYNDERE

At lære begyndere at programmere kan være en stor udfordring for underviseren (Caspersen & Kölling, 2009, p. 1), og det er en kognitiv udfordring på alle niveauer, for den lærende at lære et

[2] "Turingmaskine, abstrakt model af en enhed (maskine), der automatisk kan udføre en beregningsprocedure." (Clausen, 2009)

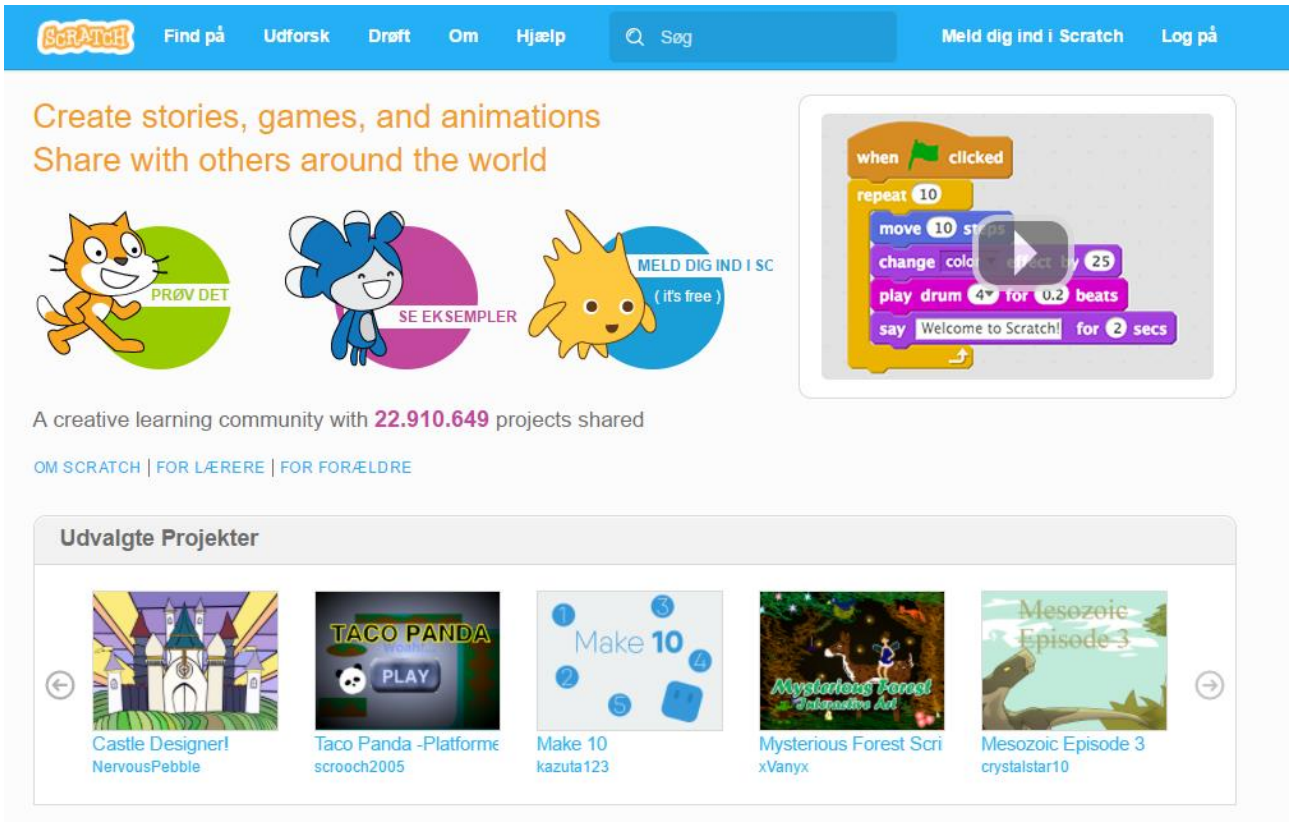
[3] Der kan ses en visuel oversigt over programmeringssprogenes historie og sammenhæng på https://www.levenez.com/lang/

programmeringssprog (Renumol, Jayaprakash, & Janakiram, 2009). I 80'erne satte et forskningsteam sig for at undersøge, hvordan man kan designe programmeringssprog, så de mere lignede naturlige sprog som fx engelsk (Guzdial, 2008, p. 26). Man kom bl.a. frem til, at nogle sprog frem for andre er bedre til at lære novicer programmering. I 00'erne undersøgte man igen, hvordan man designer programmeringssprog, der hjælper novicen med tænke som en Computer Scientist (Guzdial, 2008, p. 27), og der er udviklet flere begynderprogrammeringssprog.

Michael E. Caspersen og Professor of Computer Science, Michael Kölling (2009) argumenterer for, at det er vigtig at have øje for processen, når nybegyndere i programmering skal undervises og forslår, at der anvendes en agil tilgang med *stepwise improvement* (Caspersen & Kölling, 2009, p. 4), så processen er med til at hjælpe det kognitive på vej, når programmering skal læres. Valget af sprog kan dog også have betydning, og der findes efterhånden en del programmeringssprog, der er udviklet til begyndere, og en del at disse er udviklet til børn bl.a. Scratch (Resnick et al., 2009). Tilbage i 1970'erne blev de første programmeringssprog til børn og begyndere udviklet. Det førte til bl.a. Logo og Basic (Resnick et al., 2009).

2.4 SCRATCH

MIT Media Labs forskningsgruppe Lifelong Kindergarten har udviklet Scratch (Resnick et al., 2009), som både er et visuelt objektorienteret programmeringssprog med kodeblokke, der kan sammensættes til større komplekse sammenhænge, og et programmeringsværktøj, hvorigennem man kan lave forskellige projekter af visuel art som spil, animationer og simuleringer. Scratch, som programmeringsværktøj, findes i både en online og stand-alone udgave, så det også er muligt at arbejde offline i Scratch. Scratch i online version kan tilgås på https://scratch.mit.edu, hvor alle, gennem et stort community (Figur 1), kan gå ind for at afprøve og evt. videreudvikle på spil, animationer osv., som andre har lavet, få et indblik i koden eller selv bruge koden og ændre den.



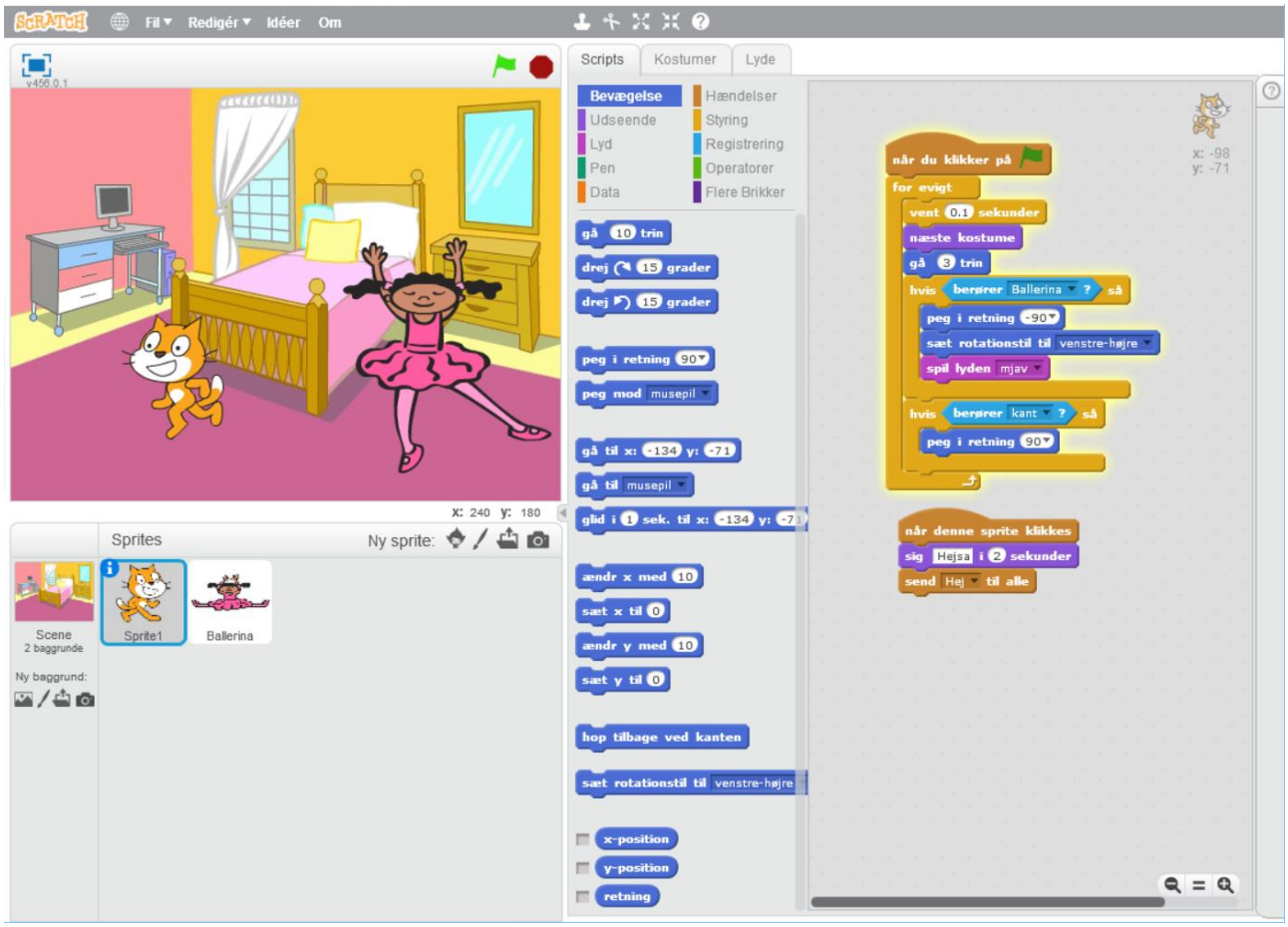
Figur 1: Scratch community (https://scratch.mit.edu/)

Programmeringssproget Scratch bygger på koncepterne fra Logo, som er udviklet og designet af bl.a. professor Seymour Papert, der allerede i 1967 begyndte på udviklingen af programmeringssproget specielt udviklet til børn. Papert var inspireret af psykolog Jean Piaget (1898 - 1980) og hans syn på børns kognitive udvikling, da han designede og udviklede Logo (Papert, 1980, pp. 7, 156). Papert forskede bl.a. i datalogi og pædagogik. I 1980 skrev Papert *Mindstorms: Children, Computers, and Powerful Ideas* (1980) om Logo, hvor sproget blev angivet til at være en hjørnesten i gentænkningen af tilgangen til læring og uddannelse (Resnick et al., 2009). Koncepterne, som både Logo og Scratch bygger på, er at 1) det skal være nemt at komme i gang = low floor, 2) det over tid skal være muligt at kreere stadigt mere komplekse projekter = high ceiling og 3) det skal være muligt at udvikle så mangeartede projekter som muligt = wide walls (Resnick et al., 2009).

Ifølge Resnick m.fl. så lærer Scratch-programmørerne "computational concepts, as well as how to think creatively, reason systematically, and work collaboratively: all essential skills for the 21st century" (Resnick et al., 2009, p. 60). Forfatterne advokerer for, at det er nødvendigt, at lære børn og unge at programmere, fordi det øger deres muligheder for at være kreative med en computer, og at programmering i særlig grad understøtter computational thinking (Resnick et al., 2009).

Brugerne af Scratch er især børn i alderen otte til 16 år, hvor de 12-årige topper de øvrige. Derudover har den også en større skare af voksne (Resnick et al., 2009, p. 60). Så det tyder på, at den 6. klasser, hvorfra vi henter vores empiri, passer godt ind i målgruppen.

SÅDAN FUNGERER SCRATCH



Figur 2: Scratch udviklingsværktøj

Scratchværktøjet er bygget op af flere dele, som det ses på Figur 2. Nederst i venstre side finder man en værktøjslinje, hvorfra man kan oprette Baggrund og Sprite. Begge dele kan betegnes som objekter. Øverst til venstre ses visualiseringen af det projekt, man arbejder på. Området bruges også til test. I midten findes tre faneblade, som repræsentere de tre forskellige typer værktøj, der kan anvendes til udvikling af projekter. Området til højre er arbejdsfladen, hvor projektet redigeres gennem værktøjsfanerne.

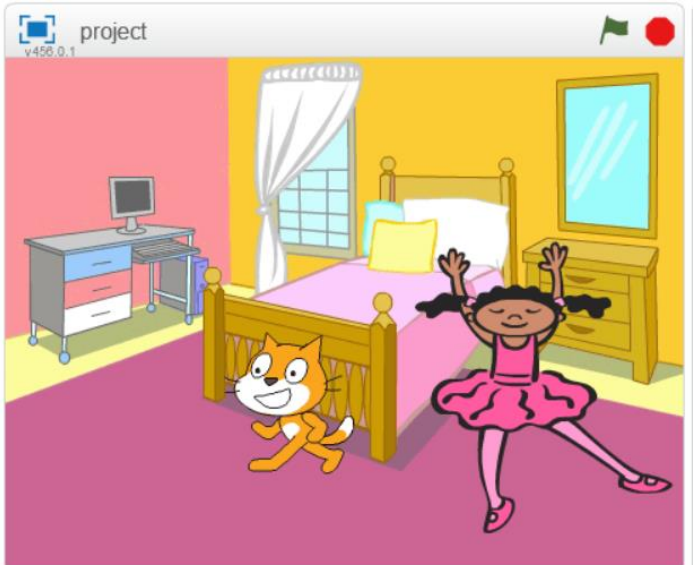
Sprite, som betyder todimensionel figur, er et objekt repræsenteret med et billede, som enten er en tegning, man tegner direkte i Scratch, et billede man har fundet og uploadet eller et billede fra Scratchs bibliotek. En Sprite er et objekt, der kan udføre en række handlinger i et Scratchprojekt (Scratch-wiki, n.d.). Baggrundsobjektet er også repræsenteret ved et todimensionelt billede. Når

man har valgt en baggrund, ses baggrunden øverst til venstre. Når man opretter en Sprite, vil den automatisk kunne ses på baggrunden.

I værktøjsfanen Scripts finder man programmeringsblokkene. Blokkene er delt i forskellige kategorier. Hver enkelt kodeblok er farvet i en bestemt farve, der repræsenterer en bestemt kodetype. Hver kodeblok har en form, som sikrer, at den kun kan sammensættes med en anden blok, hvor den passer ind i forhold til den bagvedliggende kode og syntaks (Resnick et al., 2009). Fx er bevægelsesblokke blå, hændelsesblokke brune og styringsblokke orange. Hver blok har tekst, der ligner hverdagssprog, som ikke kan ændres (Figur 3). I nogle af blokkene er det muligt at ændre værdier eller trække andre blokke ind i. I Kostumer kan der sættes forskellige præinstallerede eller egne billeder på Spriten eller Baggrunden, som kan programmeres til at skifte undervejs i spillet. I Lyd kan man vælge lyde fra et bibliotek eller optage forskellige lyde, som også kan programmeres. Der programmeres ved at vælge en Sprite eller Baggrund og vælge fanebladet Scripts, og herfra trækker man blokke ind på fladen til venstre (Figur 2).

Koden testes ved at trykke på det lille flag over baggrunden øverst til venstre (Figur 4). Man stopper testen ved at trykke på den røde sekskant ved siden af flaget.

Figur 3: Kodeblokke, med hverdagsagtig sprog - her dansk

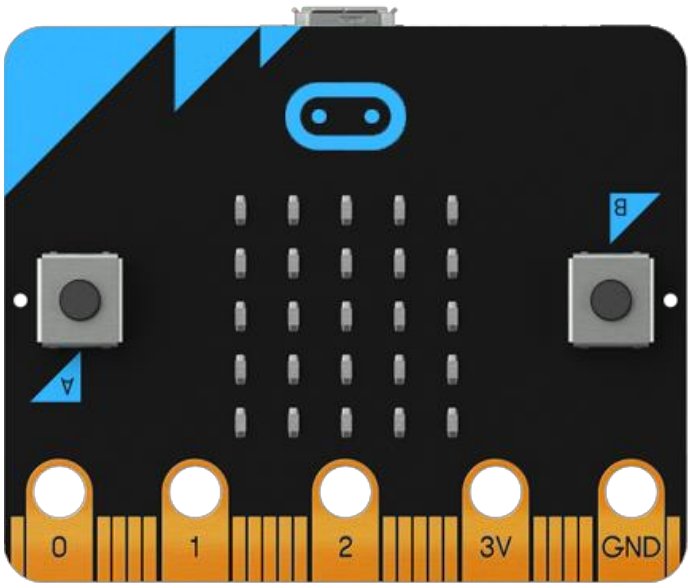


Figur 4: Testområde i Scratch

2.5 COMPUTATIONAL THINKING

Computational thinking handler om at få kompetencer til at løse problemer og tænke, som man gør i den datalogiske verden. Computational thinking, som blev navngivet af Jeanette Wing i 2006 (Wing, 2006), har som kompetence en historie, der strækker sig tilbage til 60'erne. I 1962 udtalte Alan Perlis, som betegnes som en af prionerne inden for datalogien (Nofre, n.d.), at "teaching everyone on campus to program is a noble goal"(Guzdial, 2008, p. 25). Perlis mente, at programmering skulle være en del af enhver liberal uddannelse, da undersøgelse af selve programmeringsprocessen gav indsigt i og forståelse for "theory of computation" (Guzdial, 2008, p. 25), og denne viden ville de studerende kunne bruge i andre sammenhænge.

På samme måde som 21st century skills har fået større fylde, er computational thinking kommet til at indgå i skoleregi flere steder i verden. I UK udkom der en ny læseplan "Computing National Curriculum" i 2014, hvilket har sat computational thinking på skoleskemaet i de engelske skoler (Dorling, 2015). I samarbejde med BBC, blev der desuden uddelt Micro:bits (Figur 5), der er en programmerbar computer i lommestørrelse, til alle 11-12 årige elever i de britiske skoler (BBC, 2016).



Figur 5: Micro:bit, programmerbar mini computer

I USA gav den tidligere præsident Obama et løfte om at støtte Computer Science med 4 milliarder dollars, for at sikre at børn og unge fra børnehave til High School udvikler de computational thinking skills, der er nødvendige for at være aktive borgere i et teknologidrevet samfund (Smith, 2016). Dansk Erhverv og IT-Branchen vil sætte computational thinking på skoleskemaet som et nyt kreativt og innovativt it-fag og inviterede ministre og andre interessenter til høring d. 1. feb. 2017 (IT-Branchen, 2017b). Denne tendens til at sætte eller ville sætte computational thinking på

skoleskemaet ses også i mange andre lande som Australien, Israel, Rusland, Sydafrika og New Zealand (Grover & Pea, 2013).

HVAD ER COMPUTATIONAL THINKING?

Tilbage i 2006 hvor Wing publicerede sin indflydelsesrige artikel om computational thinking, beskrev hun, at computational thinking er en grundlæggende problemløsende kompetence, der ikke kun er forbeholdt dataloger (Wing, 2006). Computational thinking er ifølge Wing en forkortelse af "thinking like an computer scientist" (Wing, 2006, p. 33) og defineres således "Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer - human or machine - can effectively carry out" (Wing, 2014).

Computational thinking indeholder, ifølge Wing (2006, 2008, 2016), en række kerneområder, som udvikles, når der programmeres, og tilsammen er med til at udvikle computational thinking. Forståelse for disse kerneområder giver den enkelte en række værktøjer, som kan anvendes i mange sammenhænge i det daglige liv til at løse store og små problemer (Wing, 2006). Computational thinking kan overordnet defineres ud fra tre kerneområder, som er tæt forbundne. Kerneområderne er:

1) *Abstraktion* (Wing, 2006, p. 33, 2008, p. 3717, 2010, p. 1), det at kunne abstrahere fra detaljer. Abstraktion handler bl.a. om at kunne genkende mønstre, lave og bruge algoritmer til løsninger af problemer, kunne formulere problemer og opsplitte problemer i mindre dele og at kunne genbruge løsninger fra tidligere.
2) *Rekursion* (Wing, 2006, p. 33, 2008, p. 3721, 2010, p. 1), det at gentage det samme flere gange. Rekursion handler om genbrug og gentagelse. Når man er i stand til at abstrahere fra detaljen og se på helheden, vil man opdage mønstre og algoritmer, som kan bruges på detaljen. Man kan så afprøve mønstret eller algoritmen på detaljen for at se, om det var dette rette abstraktion, man gjorde sig. Ellers forsøger man sig med nye abstraktioner.
3) *Logisk tænkning* (Wing, 2010, pp. 1–2), det at kunne lave følgeslutninger. Når der skal abstraheres fra detaljen, er der brug for, at man kan se helheden gennem detaljen, så man kan beslutte, om man ser et mønster, der kan genbruges. Dette gøres gennem forsøg, test og afprøvninger.

Da Wing i 2006 skrev om computational thinking, forestillede hun sig ikke, at der 10 år senere ville være skoler og lande – og slet ikke i den grad – der havde programmering og *Computational Thinking*, som en del af læseplanerne (Wing, 2016). Den store udbredelse har medført mange fortolkninger af computational thinking.

INTERNATIONALT SOCIETY FOR TECHNOLOGY IN EDUCATION (ISTE)

ISTE definerer computational thinking som havende fem centrale færdigheder, der skal trænes for at opnå computational thinking:

1. Decomposition
2. Generalizing
3. Algorithmic thinking
4. Evaluation
5. Abstraction
   (Nash, 2017)

Gennem disse fem trin eller færdigheder undervises eleverne i, hvordan der løses problemer i en datalogisk sammenhæng (Nash, 2017).

GOOGLE FOR EDUCATION

Ifølge Google for Education består computational thinking af en række kognitive processer, det medfører en række håndgribelige resultater, når der løses datalogiske problemer (Google for Education, n.d.) (Figur 6). Google for Education skriver, at computational thinking er essentiel, når der løses problemer i udviklingen af computerprogrammer, men at det også kan bruges til løse problemer inden for andre discipliner bl.a. humaniora.

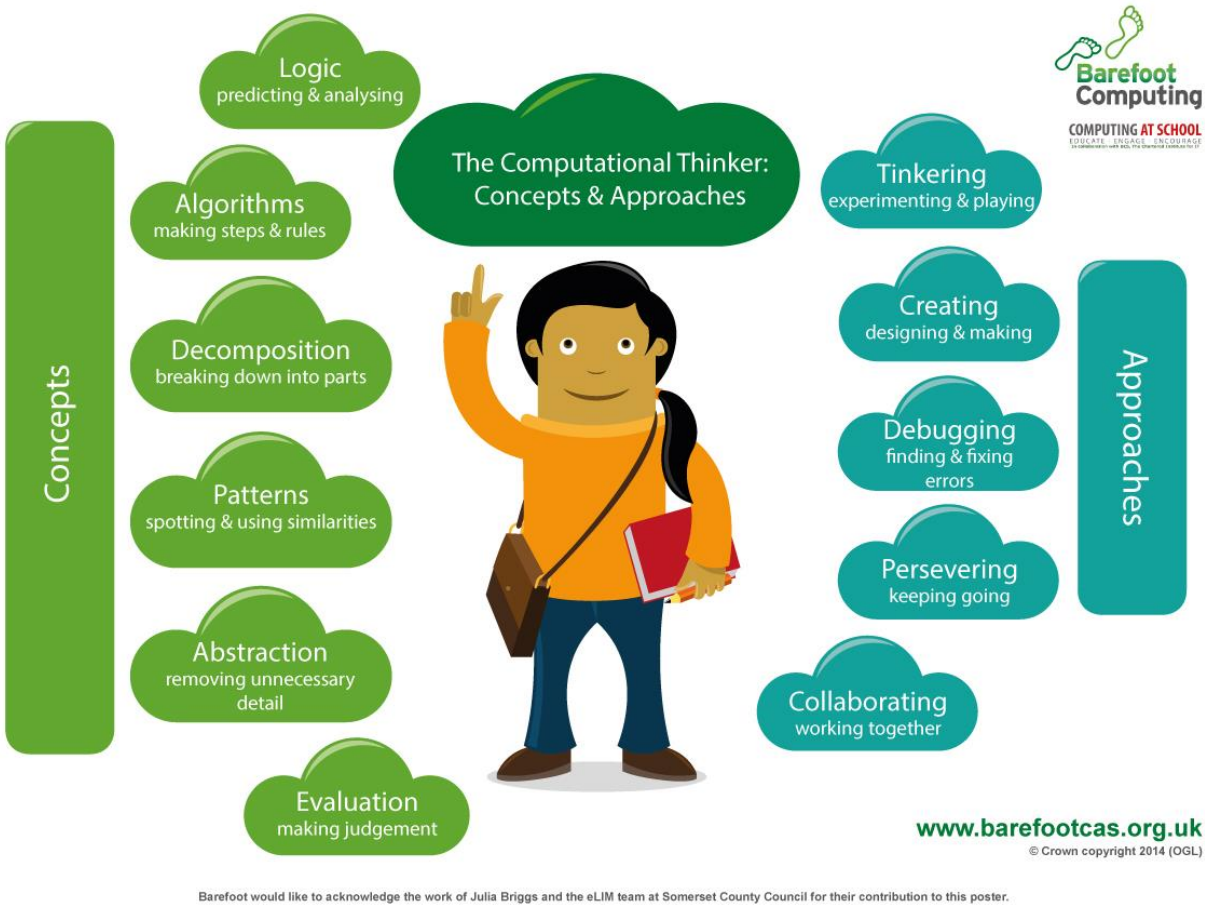
- **Abstraction:** Identifying and extracting relevant information to define main idea(s)
- **Algorithm Design:** Creating an ordered series of instructions for solving similar problems or for doing a task
- **Automation:** Having computers or machines do repetitive tasks
- **Data Analysis:** Making sense of data by finding patterns or developing insights
- **Data Collection:** Gathering information
- **Data Representation:** Depicting and organizing data in appropriate graphs, charts, words, or images
- **Decomposition:** Breaking down data, processes, or problems into smaller, manageable parts
- **Parallelization:** Simultaneous processing of smaller tasks from a larger task to more efficiently reach a common goal
- **Pattern Generalization:** Creating models, rules, principles, or theories of observed patterns to test predicted outcomes
- **Pattern Recognition:** Observing patterns, trends, and regularities in data
- **Simulation:** Developing a model to imitate real-world processes

Figur 6: Google for Educations beskrivelse af koncepter i Computational Thinking (Google for Education, n.d.)

BAREFOOT COMPUTING

Da Department for Education (det engelske undervisningsministerium) i 2013 bekendtgjorde læseplaner for *Computing programmes of study* (Department for Education, 2013) på folkeskoleniveau, medførte det bl.a. at Barefoot Computing (Barefoot Computing, n.d.) opstod i

2014, for at understøtte engelske lærere i at undervise i datalogi. Barefoot Computing har fortolket *Computational Thinking* i seks koncepter og fem tilgange (Figur 7).



Figur 7: Barefoot Computing's definition på computational thinking

Overordnet set ses computational thinking, ifølge Barefoot Computing (2014), som en måde at løse problemer på, hvor man i gennem undersøgelser og afprøvninger undersøger, om man løser problemer på den bedst mulige måde. Der bliver ikke kun undersøgt, om det er den rigtige løsning, men også om det er den bedste løsning. Derudover bliver der også fremhævet, at kompetencen er med til at udvikle kollaborative færdigheder, når der testes og afprøves (Barefoot Computing, 2014). Jo flere færdigheder man opøver inden for de seks koncepter og fem tilgange, jo mere er man i stand til at udnytte kompetencen computational thinking på andre områder.

DISKUSSION

Til trods for at der er mange bud på definitioner af computational thinking, synes der at være et fokus på de naturfaglige STEM-fag. Google for Education nævner dog, at computational thinking kan overføres til andre fag eller discipliner (Google for Education, n.d.). Der synes dog at være en lille bevægelse mod en bredere forståelse for og tanker om, at det naturvidenskabelige område

ikke er adskilt fra humaniora i dagligdagen, men at de er tæt sammenbundne. I USA er man begyndt at tale STEAM i stedet for STEM, hvor A står for Art, som i denne kontekst handler om at inddrage kunst og design ("STEM to STEAM," n.d.), som en naturlig del af de øvrige naturvidenskabelige fag. Scratch, som er opbygget med baggrund i computational thinking, kan lige så vel bruges til at lave en animation i dansk, eller der kan laves "intelligent" håndværk og design ved hjælp af fx Micro:bit.

Vi lægger os tæt op af Barefoot Computing's forståelse af computational thinking som værende en problemløsende kompetence, der består af en række koncepter og tilgange, hvor der er fokus på en lang række vidensområder og færdigheder, der opøves, når der programmeres. Tilsammen er koncepterne og tilgangene med til at udvikle børn og andre til at blive computational thinkers.

Computational thinking vil kunne hjælpe og udnyttes i mange hverdagssituationer, eller som Wing udtrykker det: "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, p. 33). Vi har også taget Perlis' udtryk til os: "teaching everyone on campus to program is a noble goal" (Guzdial, 2008, p. 25). Vi finder, at computational thinking er vigtig, og at det burde sættes på dagsordenen inden for mange fag og emner, som en naturlig del af enhver læseplan. Dette vil naturligvis kræve opkvalificering af lærerne, der ikke er uddannet til computational thinking.

At kunne programmere kan være med til at danne eleverne til kritiske borgere. I artiklen *Datalære - faget, som forsvandt* (Hvid, 2001) fortæller den pensionerede folkeskolelærer Gerd Belhage om et undervisningsforløb i forbindelse med et valg, hvor elever i en klasse udviklede et edb-program, der undersøgte, hvordan eleverne på hele skolen stemte. Klassen fandt undervejs ud af, at de, ved at kun at tilføje en ekstra linje i programmets kode, kunne undersøge, om der var forskel på hvad store og små elever stemte. Klassen begyndte at tilføje andre linjer i koden, der kunne afdække andre statistiske informationer. Dette førte til en etisk diskussion på klassen om, hvad man kunne tillade sig at undersøge. Belhage udtaler "'Den indsigt og kritiske erkendelse…havde de aldrig fået, hvis de ikke selv havde haft fingrene nede i faget'" (Hvid, 2001).

3 DANNELSE OG KOMPETENCER

I dette afsnit vil vi beskrive begrebet dannelse med udgangspunkt i tyske dannelsesteorier, som den danske dannelsestænkning er meget inspireret af. Derefter vil vi beskrive digital dannelse for til sidst at identificerer dannelsespotentialet i programmering ud fra en didaktisk analyse.

3.1 DANNELSE

Dannelse er en "betegnelse for dels en pædagogisk norm ved valg af indhold i opdragelse og undervisning, dels en social norm, der udpeger en bestemt adfærd, væremåde, opførsel og viden som dannet" (Winther-Jensen, n.d.). Dannelse er både en proces, hvorigennem mennesket hverver sig et bestemt indhold af viden, og dannelse kan også betegnes som et resultatet af denne proces (Winther-Jensen, n.d.). Dannelse kan oversættes til bildning på svensk, dannelse eller danning på norsk og bildung på tysk, men på engelsk findes der ikke et tilsvarende og dækkende ord.

Dannelsesbegrebet blev et centralt begreb i det tysktalende pædagogiske område i perioden 1770 – 1830. Som vigtige karakteristika for de klassiske dannelsesteorier, fremhæver den tyske didaktiker Wolfgang Klafki (1927-2016) følgende; "Dannelse forstås altså som evne til fornuftig selvbestemmelse, der forudsætter eller indbefatter emancipation for det fremmedbestemte, som evne til autonomi, til fri selvstændig tænkning og til frit at træffe selvstændige moralske afgørelser" (Klafki, 2016, p. 32). Dannelse opstås gennem konfrontation med den verden, vi lever i dvs. dannelse udvikles i en kontekst af historiske, samfundsmæssig, politiske og kulturelle begivenheder (Jank & Meyer, 2006, p. 169). Dannelse er for alle og ikke kun for en afgrænset samfundsgruppe eller elite. Dette er ét af principperne for almendannelsen. Derudover er almendannelsen alsidig, og derfor skal den enkelte udvikles på flest mulige områder, hvilket de forskellige fag i folkeskolen bl.a. bidrager til.

Ifølge professor Karsten Schnack adskiller dannelse sig fra begrebet uddannelse, ved at dannelse er noget, der sker med det enkelte individ, hvor uddannelse sker i formelle rammer og har til formål at forberede individet til at varetage opgaver i dets arbejdsliv og dermed "tage del i samfundets værdiskabelse" (Nordenbo, 2011, p. 45). I praksis er grænsen mellem dannelse og uddannelse dog mere flydende, da flere kompetencer, som fx samarbejde, kommunikation og kritisk sans, udvikles gennem uddannelse og anvendes i arbejdslivet, samtidig med at disse kompetencer kan have et dannelsesperspektiv for det enkelte individ (Schnack, 2011, pp. 33–34). Hvor kompetencer henviser til noget "individet gør eller kan gøre"(Nordenbo, 2011, pp. 52–53) forbindes dannelse med en tilstand, som en dannet person besidder (Nordenbo, 2011, p. 52), og dannelse er, modsat kompetencer, umuligt eller meget vanskeligt at måle (Schnack, 2011, p. 42).

Dannelsesbegrebet bliver af nogle kritiseret for at være et idealiserende og forældet begreb, der ikke kan leve op til de pædagogiske behov, som et demokratisk samfund stiller krav om (Klafki, 2016, pp. 59–60). Alligevel argumenterer Klafki for, at dannelsesbegrebet fortsat skal være en grundkategori for de pædagogiske opgaver både nu og i fremtiden, for at undgå "et virvar af usammenhængende sideløbende og endog modsatrettede enkeltaktiviteter, hvis pædagogiske hensigter, tiltag, handlinger og individuelle læringsbestræbelser skal være noget, man kan *begrunde* og tage *ansvaret* for" (Klafki, 2016, p. 60). Schnack er også enig i, at dannelsesdiskussionen er aktuel, da målene, i fx folkeskolen, må have et formål, og at "formålet

må rumme en idé om, hvad der er menneskeligt og samfundsmæssigt værdifuldt" (Schnack, 2011, p. 43).

DANNELSE I FOLKESKOLEN

Folkeskolens formålsformulering handler både om dannelse til menneske og dannelse til borger i samfundet og indeholder dermed både en individ- og fællesskabsdimension (Schnack, 2011, p. 35). I folkeskolens formålsparagraf, stk. 1, fremhæves bl.a. livslang læring, alsidig udvikling, samt en form for kulturhistorisk dannelse, hvor eleven skal være fortrolige med egen og andre landes kulturer (Schnack, 2011, p. 36):

> Folkeskolen skal i samarbejde med forældrene give eleverne kundskaber og færdigheder, der forbereder dem til videre uddannelse og giver dem lyst til at lære mere, gør dem fortrolige med dansk kultur og historie, giver dem forståelse for andre lande og kulturer, bidrager til deres forståelse for menneskets samspil med naturen og fremmer den enkelte elevs alsidige udvikling. (Ministeriet for Børn Undervisning og Ligestilling, 2016)

I stk. 2 beskrives det, hvordan "Folkeskolen skal udvikle arbejdsmetoder og skabe rammer for oplevelse, fordybelse og virkelyst, så eleverne udvikler erkendelse og fantasi og får tillid til egne muligheder og baggrund for at tage stilling og handle" (Ministeriet for Børn Undervisning og Ligestilling, 2016). Her er det centralt, at eleven aktivt skal tage stilling og handle.

Den sidste del af formålsparagraffen handler om fællesskabsdimensionen, hvor folkeskolen skal medvirke til elevernes demokratiske og politiske dannelse (Schnack, 2011, p. 35) ved at "forberede eleverne til deltagelse, medansvar, rettigheder og pligter i et samfund med frihed og folkestyre. Skolens virke skal derfor være præget af åndsfrihed, ligeværd og demokrati" (Ministeriet for Børn Undervisning og Ligestilling, 2016).

3.2 DIGITAL DANNELSE

Tilbage i 2009 optræder betegnelsen digital dannelse i faghæfte 48 *It- og mediekompetencer i folkeskolen:* "Børn og unges formelle og uformelle tilstedeværelse i web 2.0-omgivelser betyder, at skolen fremover skal kunne rumme disse uformelle kompetencer og støtte eleverne i tilegnelsen af en digital og tidssvarende dannelse" (UVM, 2010a, p. 5). Ifølge Jeppe Bundsgaard (1970), professor i it og fagdidaktik, handler begrebet digital dannelse

> både om at kunne anvende og opføre sig ordentligt med teknologi, erkende teknologiens rolle i vores fælles liv, forholde sig til udfordringer og deltage engageret i at forstå og handle i forhold til de muligheder og udfordringer teknologier giver for os i vores fællesskaber, i vores samfund og som individer. (Bundsgaard, 2017, p. 12)

Digital dannelse handler både om at tilegne sig konkrete kompetencer til at anvende teknologier, men det handler også om en teknologiforståelse, og det at kunne anvende teknologi i fællesskabets tjeneste. Bundsgaard mener også, at computational thinking kan falde ind under digital dannelse, idet computational thinking netop handler om at forstå, hvordan teknologien fungerer, men også om at have kompetencerne til selv at udvikle og styre teknologierne (Bundsgaard, 2017, p. 13). Med andre ord handler digital dannelse om alt det, børn og unge skal lære, for at de kan håndtere digital teknologier, og ifølge Bundsgaard, er en stor del af den digitale dannelse skolens opgave (Bundsgaard, 2017, p. 14).

Den Fællesoffentlige digitaliseringsstrategi, *Et stærkere og mere trygt digitalt samfund 2016-2020,* sætter også fokus på børn og unges digitale dannelse, som defineres således:

> Digital dannelse handler om, at vores børn og unge bliver i stand til at begå sig læringsmæssigt, socialt og etisk i den digitale virkelighed. Børn og unge skal blandt andet opnå de nødvendige digitale kompetencer, så de bliver i stand til at bruge it som arbejdsredskab og aktivt kunne bruge it til at kunne tilegne sig viden og løse problemer. (Regeringen/KL/Danske-Regioner, 2016, p. 55)

En række initiativer skal derfor sætte fokus på digital dannelse, således at børn og unge har de nødvendige forudsætninger, der gør dem i stand til aktivt at anvende relevante teknologier, når de løser komplekse og virkelighedsnære problemstillinger (Regeringen/KL/Danske-Regioner, 2016).

DISKUSSION

Betegnelsen *Digital dannelse* bliver kritiseret af bl.a. adjunkt Ove Christensen, som mener at "dannelsen handler om det at blive til nogen - det har ikke forskellige analoge og digital komponenter" (Christensen, 2017). Når digital sættes foran dannelse, bliver det, ifølge Christensen (2017), reduceret til digital viden, digitale kompetencer og færdigheder og er dermed det modsatte af dannelse. Med udgangspunkt i de klassiske dannelsesteorier, hvor dannelse opstår igennem konfrontation med den verden, vi lever i, dvs. dannelse udvikles i en kontekst af historiske, samfundsmæssig, politiske og kulturelle begivenheder (Jank & Meyer, 2006), kan dannelse i et teknologidrevet samfund også betragtes som værende almendannelse – uden digital foran.

Vi erkender dog behovet for at udskille en særlig del af dannelsen og benævne den som værende digital fx taler vi også om demokratisk og kulturhistoriske dannelse. Dermed bliver det lettere for forskellige parter, både indenfor og uden for skoleverdenen, at diskutere og få en forståelse af digital dannelse. Da dannelsesbegrebet er meget bredt, vil vi fokusere på den del af dannelsen, som betegnes som digital dannelse.

3.3 HAR PROGRAMMERING DANNELSESPOTENTIALE?

Ifølge Klafki er det lærerens opgave at tilrettelægge undervisningen ud fra læseplanernes indhold og de fem grundspørgsmål til didaktisk analyse (Figur 8) "og derigennem afdække dannelsespotentialet i det foreskrevne dannelsesindhold med henblik på skoleklassen og egne dannelseshensigter" (Jank & Meyer, 2006, p. 177). Med udgangspunkt i Klafkis fem grundspørgsmål vil vi undersøge, om programmering har dannelsespotentiale:

Figur 8: Frit efter skema i Didaktiske modeller – grundbog i didaktik (Jank & Meyer, 2006, p. 165)

Set i lyset af Danmarks høje it-parathed, den lette tilgang til teknologi og programmeringsværktøjer samt aktualiteten af programmering, mener vi, at programmeringens *betydning her og nu* er stor. Programmering vil også have en stor *betydning i fremtiden*, hvor digitale teknologier kommer til at fylde endnu mere. *Indholdets struktur* i programmering kan forklares som en indholdsskitse, der angiver den grundlæggende viden om programmering, som eleverne skal opnå, samt de forskellige niveauer indholdet kan opdeles i (Jank & Meyer, 2006). I vores forløb skal eleverne programmere et spil i Scratch, og forløbet kan have *eksemplarisk betydning* ift. andre programmeringssprog, spiludvikling, computational thinking mm. Programmeringens *tilgængelighed* er stor, og det vil være forholdsvis let at relatere til situationer og problemstillinger fra elevernes hverdag, set i lyset af deres brug af teknologi i både skole og fritid.

DELKONKLUSSION

Ud fra ovenstående didaktiske analyse kan vi konkludere, at programmering har dannelsespotentiale i den danske folkeskole, og svaret på projektets titel *Hvorfor skal eleverne programmere i folkeskolen?* kan dermed lyde: Fordi programmering giver eleverne kompetencer til at forstå, håndtere og skabe teknologi, hvilket er en vigtig del af elevernes digitale dannelse.

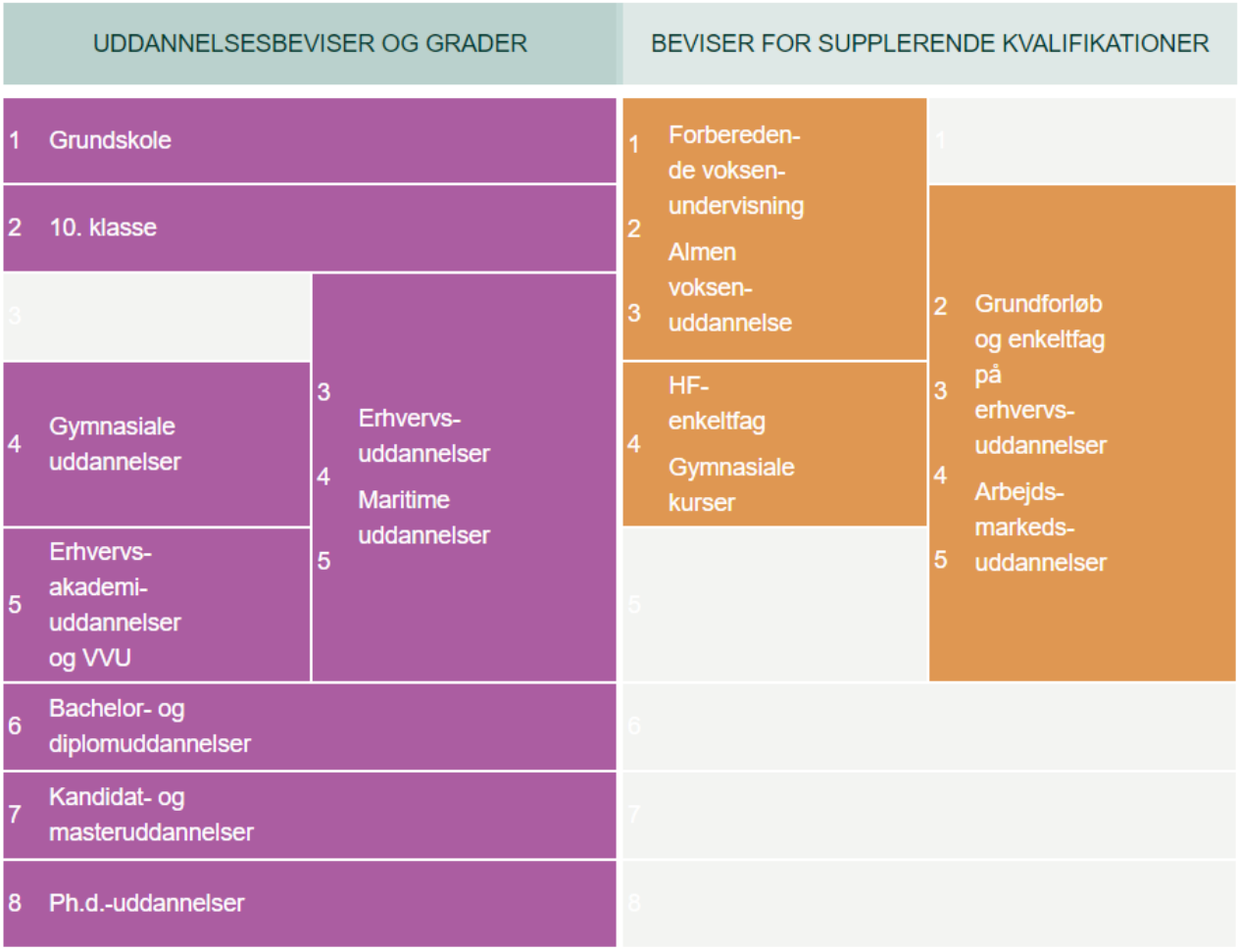
Dette åbner op for nye spørgsmål, som fx hvor meget skal programmeringen fylde i folkeskolen? Hvor meget skal eleverne kunne? Hvilken viden, færdigheder og kompetencer skal eleverne skal have inden for programmering? Dette leder os over i næste afsnit, hvor vi beskriver forskellen på begreberne viden, færdigheder og kompetencer.

3.4 VIDEN, FÆRDIGHEDER OG KOMPETENCER

For at kunne inddrage computational thinking, programmering og teknologisk forståelse i folkeskolen er det nødvendigt at sætte folkeskolens formål i relief til den dansk kvalifikationsramme for livslang læring (UFM, n.d.-d)*, som er en udløber af Bologna-deklarationen* (Declaration, 1999; UFM, n.d.-b)*.*

DEN DANSKE KVALIFIKATIONSRAMME

Den danske kvalifikationsramme for livslang læring bruges til at give overblik over uddannelsesniveauer (UFM, n.d.-d) i Danmark, så de kan sammenlignes med andre europæiske lande, der også har tilsluttet sig Bologna-deklarationen (Declaration, 1999; UFM, n.d.-b). Den danske kvalifikationsramme er ligesom den Europæiske kvalifikationsramme (UFM, n.d.-c) opdelt i 8 uddannelsesniveauer og folkeskolens afgangsprøve (grundskole) befinder sig på niveau 1 (UFM, n.d.-e, n.d.-f) (Figur 9).



Figur 9: Indplacering af uddannelses beviser på de 8 niveauer (UFM, n.d.-g)

Hvert niveau er beskrevet ud fra begreberne viden, færdigheder og kompetencer (UFM, n.d.-a) i brede formuleringer om læringsudbyttet for uddannelsesniveauet. De forventede læringsudbytter stiger for hvert niveau i kompleksitet og selvstændighed. De forventede læringsudbytter for niveau 1 kan ses i Figur 10.

NIVEAU 1		
VIDEN	**FÆRDIGHEDER**	**KOMPETENCER**
Skal have grundlæggende viden inden for almene fag. Skal have grundlæggende forståelse af naturgivne, kulturelle, sociale og politiske forhold.	Skal besidde grundlæggende sproglige, numeriske, praktiske og kreative færdigheder. Skal kunne anvende forskellige grundlæggende arbejdsmetoder. Skal kunne evaluere eget arbejde. Skal kunne præsentere resultaterne af eget arbejde.	Skal kunne tage personlig stilling og handle i simple og overskuelige situationer. Skal kunne arbejde selvstændigt med på forhånd definerede problemstillinger. Skal have lyst til at lære og kunne indgå i delvist åbne læringssituationer under supervision.

Figur 10: Beskrivelse af niveau 1 i den danske kvalifikationsramme for livslang læring (UFM, n.d.-e)

FÆLLES MÅL

Folkeskolens formål rummer niveau 1 og er fortolket gennem Fælles Mål. Fælles Mål er nationale og obligatoriske mål for, hvad eleverne i folkeskolen skal lære i skolens fag og emner (UVM, 2017a) og "består af fagformål, kompetencemål og underliggende færdigheds- og vidensmål" (UVM, 2017a). Hvert kompetencemål kan være inddelt i flere trin, hvor hvert trin refererer til et antal sammenhængende klassetrin (fx 1.-2. klasse). På hvert trin er kompetencemålet delt i forskellige kompetenceområder, hvorunder kompetenceområdet er opdelt i flere faser, som består at et målpar, der består af færdigheds- og vidensmål. Se nedenstående eksempel fra faget fysik/kemi på fagformål (Figur 11), kompetencemål (Figur 12 og Figur 13) samt færdigheds- og vidensmål (Figur 14), hvor programmering er nævnt i forbindelse med fase 3's målpar (svarende til 9. kl.) under kompetencemålet *undersøgelse* i kompetenceområdet *produktion og teknologi*.

Fagformål

Eleverne skal i faget fysik/kemi udvikle naturfaglige kompetencer og dermed opnå indblik i, hvordan fysik og kemi – og forskning i fysik og kemi – i samspil med de øvrige naturfag bidrager til vores forståelse af verden. Eleverne skal i fysik/kemi tilegne sig færdigheder og viden om grundlæggende fysiske og kemiske forhold i natur og teknologi med vægt på forståelse af grundlæggende fysiske og kemiske begreber og sammenhænge samt vigtige anvendelser af fysik og kemi.

Stk. 2. Elevernes læring skal baseres på varierede arbejdsformer, som i vidt omfang bygger på deres egne iagttagelser og undersøgelser, blandt andet ved laboratorie- og feltarbejde. Elevernes interesse og nysgerrighed over for fysik, kemi, naturvidenskab og teknologi skal udvikles, så de får lyst til at lære mere.

Stk. 3. Eleverne skal opnå erkendelse af, at naturvidenskab og teknologi er en del af vores kultur og verdensbillede. Elevernes ansvarlighed over for naturen og brugen af naturressourcer og teknologi skal videreudvikles, så de får tillid til egne muligheder for stillingtagen og handlen i forhold til en bæredygtig udvikling og menneskets samspil med naturen – lokalt og globalt.

Figur 11: Fagformål for fysik/kemi (EMU, n.d.-c)

7. - 9. klasse

Vis mere

Undersøgelse	Modellering	Perspektivering	Kommunikation
Eleven kan designe, gennemføre og evaluere undersøgelser i fysik/kemi	Eleven kan anvende og vurdere modeller i fysik/kemi	Eleven kan perspektivere fysik/kemi til omverdenen og relatere indholdet i faget til udvikling af naturvidenskabelig erkendelse	Eleven kan kommunikere om naturfaglige forhold med fysik/kemi

Figur 12: Kompetencemål for fysik/kemi (EMU, n.d.-b)

7. - 9. klasse

Vis mindre

Eleven kan designe, gennemføre og evaluere undersøgelser i fysik/kemi

Kompetenceområdet undersøgelse omfatter seks færdigheds- og vidensområder:

Naturfaglige undersøgelser er naturfaglige mål og er enslydende for naturfagene i udskolingen. Disse fokuserer på undersøgelsesmetoder, validering af resultater, konklusion og generalisering.

Stof og stofkredsløb fokuserer på undersøgelser af grundstoffer, kemiske reaktioner og processer i centrale stofkredsløb.

Partikler, bølger og stråling fokuserer på undersøgelser af lydbølger, farver, elektromagnetisk stråling og atomare processer.

Energiomsætning fokuserer på undersøgelser af energiomsætninger, transport og lagring af energi.

Jorden og universet fokuserer på undersøgelser af fysiske fænomener, atmosfæren og jordens ressourcer.

Produktion og teknologi fokuserer på undersøgelser af udnyttelsen af råstoffer, produktionsmetoder samt teknologier vedrørende elektronisk og digital styring.

Læs mere om undersøgelse i faget fysik/kemi

Obligatorisk ✓

Figur 13: Kompetenceområder for kompetencemålet undersøgelse (EMU, n.d.-e)

Produktion og teknologi Vis mindre

Trinforløbet tager udgangspunkt i elevernes undersøgelser af proteiner, fedt og kulhydrater. Herigennem udbygges elevernes kendskab fra natur/teknologi til fødevarers opbygning og energiindhold. Undervisningen fokuserer på fødevareproduktion, herunder konservering, emulgatorer og farvestoffer. Der arbejdes endvidere med gæringsprocesser og produktion af alkohol.

Senere skal eleverne med udgangspunkt i råstoffer som olie, kalk, salt og produkter fra landbrug og fiskeri, lave undersøgelser af industriens produktionsmetoder, herunder redoxprocesser, katalyse, elektrolyse og brug af enzymer. Eleverne skal bl.a. kunne undersøge, hvilke danske råstoffer der indgår i store industrielle produktioner samt selv kunne gennemføre dele af produktionsprocessen i mindre skala.

I sidste fase skal eleverne gennem undersøgelser af elektroniske og digitale apparater fra hverdagen opnå kendskab til, hvordan de fungerer og bliver reguleret. Gennem elevernes viden om opbygningen af elektriske kredsløb, simpel programmering og transmission af data, kan eleverne begynde selv at udføre eksperimenter vedrørende elektronisk og digital styring fx til styring af procesrobotter. Det er ikke et mål i sig selv, at eleverne lærer at programmere, men derimod at de arbejder systematisk, eksakt og reflekteret med at løse problemstillinger gennem inddragelse af it.

Fase 1	Fase 2	Fase 3
Færdighedsmål Eleven kan undersøge fødevareproduktion	**Færdighedsmål** Eleven kan undersøge udnyttelse af råstoffer og dele af produktionsmetoder	**Færdighedsmål** Eleven kan designe og gennemføre undersøgelser vedrørende elektronisk og digital styring
Vidensmål Eleven har viden om næringsstoffer og tilsætningsstoffer i fødevarer	**Vidensmål** Eleven har viden om råstoffer og produktionsprocesser	**Vidensmål** Eleven har viden om elektroniske kredsløb, simpel programmering og transmission af data

Figur 14: Viden- og færdighedsmål (målpar) inden for kompetenceområdet Produktion og teknologi (EMU, n.d.-e)

Målparret i fase 3 i Figur 14 er det eneste sted, hvor programmering er direkte nævnt som mål. Ifølge Brian Ravnborg Christensen (bilag 3) kan man finde en indirekte henvisning til programmering i fysik/kemi under kompetencemålet modellering i kompetenceområdet, samt i det vejledende eksempel i natur/teknologi på 5. - 6. klassetrin under kompetencemålet *undersøgelse* i kompetenceområdet *teknologi og ressourcer* i fase 2's målpar (Figur 15), som giver forslag til læringsmål, der omhandler styring og programmering af robotter. Desuden er programmering også nævnt i vejledningen for faget matematik (EMU, n.d.-f) i kapitel 3.4 *IT- og medier*. Vejledningen er ikke obligatorisk jf. Brian Ravnborg Christensen (bilag 3).

Figur 15: Vejledende eksempel på læringsmål i natur/teknologi (EMU, n.d.-a)

FÆRDIGHEDS- OG VIDENSMÅL BLIVER VEJLEDENDE

Rækken af færdigheds- og vidensmål har siden 2014 været bindende nationale mål. Den 19. maj 2017 udsendte UVM en pressemeddelelse, hvori de beretter, at forligskredsen har indgået en ny aftale, der gør færdigheds- og vidensmålene vejledende (UVM, 2017b). Dermed går det eneste færdigheds- og vidensmål, der helt konkret nævner programmering, fra at være bindende til at være vejledende. Det der er tilbage af bindende mål er kompetencemålene (Figur 12) og overskrifterne til færdigheds- og vidensmålene, som i fysik/kemi lyder således: ”Eleven kan designe, gennemføre og evaluere undersøgelser i fysik/kemi”(EMU, n.d.-e) (Figur 13).

Vicedirektøren for Styrelsen for IT og læring (STIL) Jakob Harder udtaler, at programmering ”indgår i fagene og lærerne fastsætter derfor den nødvendige fordeling af timer til realiseringen af fagenes mål” (bilag 2). Det vil derfor være op til den enkelte lærer at vurdere, hvor meget eleverne skal programmere i fagene. Hvorvidt eleverne møder programmering i fagene afhænger således af lærerens prioritering af programmering samt dennes viden, færdigheder og kompetencer inden for området.

VIDEN

Definitionen af viden er, ifølge den danske kvalifikationsramme, "viden om et emne samt forståelse" (UFM, n.d.-d) og rummer typer af viden, kompleksiteten af viden samt evnen til at forklare viden (UFM, n.d.-a). Vi ser nærmere på begrebet viden gennem professor Lars Qvortrup (1950) definition, som i sin bog *Det vidende samfund - mysteriet om viden, læring og dannelse* (Qvortrup, 2005) giver et bud på kategorier af viden samt dybden af viden.

Qvortrup bruger Gregori Batesons inddeling af viden, fra første til fjerde ordens læring, til at kategorisere viden med følgende overskrifter: *Videnskategori, Vidensform, Videnssystematik* og *Vidensbetegnelse. Videnskategori* handler at bestemme dybden af viden, og hvad det er for en slags viden, det handler om. *Vidensform* handler om, hvilken form for viden, der kan identificeres. *Videnssystematikken* beskriver typen af viden, mens *Vidensbetegnelse* sætter navn på vidensformen.

Vi har i nedenstående skema indsat Qvortrups identificeringer sammen og fortolket folkeskolens formål samt den danske kvalifikationsramme for livslang læring ind de fire videnskategorier.

Orden	Videns-kategori	Vidensform	Videns-systematik	Videns-betegnelse	Læring i folkeskolen
1. ordens viden	Viden	Viden om noget	Viden om omverden	Faktuel viden/ Kvalifikation	Viden og Færdigheder
2. ordens viden	Viden om viden	Viden om videns-situationen	Viden om viden	Situativ viden/ Kompetencer	Kompetencer
3. ordens viden	Viden om (viden om viden)	Viden om videns-betingelserne	Viden om videnssystemet	Systemisk viden/ Kreativitet	Kreativitet
4. ordens viden	Viden om (viden om (viden om viden))	Verden som videns-forudsætning	Kollektiv grundlagsviden	Verdensviden/ Kultur	Dannelse

Figur 16: Ovortrups vidensformer (Qvortrup, 2005) sat sammen med folkeskolens formål (Ministeriet for Børn Undervisning og Ligestilling, 2016) og livslang læring (UFM, n.d.-d)

Det er dog også væsentlig at tale om, hvordan videnstilegnelsen/læringen foregår. Illeris (2012, p. 92) taler om oprettelse af skemaer og forandringen af disse. Illeris er inspireret af Piagets skemateori (Illeris, 2015, p. 55) og deler videnstilegnelsen op i 4 niveauer som kumulativ, assimilativ, akkomodativ samt transformativ læring. Ifølge Illeris (2015) drejer kumulativ læring sig om oprettelse af nye skemaer, som står isoleret fra andre skemaer (fx en tillært pinkode) og denne form for viden er ikke blivende, da den forsvinder, når der ikke længere er brug for denne viden. Assimilativ handler om at tilføje ny læring, der tilføjer mere viden til allerede eksisterende skemaer. Ved akkomodativ læring ændres der på eksisterende skemaer, hvorved der opstår ny eller anderledes indsigt i et område og dermed opnås nye kompetencer. Endelig er der transformativ læring, som kun opstår i særlige tilfælde, når der opstår en forandring af, hvem vi er. Fx når en nyuddannet sygeplejerske opfatter sig selv som sygeplejerske.

Set ud fra en vidensvinkel kan man diskutere, om det giver indsigt og forståelse, at beskæftige sig med programmering få gange i løbet af ens folkeskoletid i form af enkelte forløb i matematik, natur/teknologi og fysik/kemi. Hvis viden om programmering skal kunne omsættes til teknologisk forståelse og begyndende computational thinking, skal der som minimum opstå assimilativ læring.

Det kan også anfægtes, om det er tilstrækkelig med assimilativ læring, hvis der skal opstå ny indsigt i området. Opnåelse af akkomodativ læring vil kræve, at programmering skal fylde væsentlig mere i fagene og emner, end det gør i dag. Om det så skal ske ved at integrere programmering mere i skolens nuværende fag eller ved oprettelse af et selvstændigt fag, kommer helt an på, hvor vigtigt man anser teknologisk forståelse og computational thinking for. En opgradering vil dog også kræve, at der sker en væsentlig efteruddannelse af underviserne, da det stadig vil være meget få, der har kompetencerne til at kunne påtage sig at undervise i området.

FÆRDIGHEDER

Ifølge Illeris er de fleste ret enige i, hvad færdighed betyder (Illeris, 2012, p. 19). Slår man op i Ordbogen.com, får man følgende forklaring på begrebet ”talent for noget bestemt, som enten er medfødt eller opøvet” (“Færdighed,” n.d.). I den danske kvalifikationsramme er begrebet forklaret med at ”Færdigheder angiver, hvad en person kan gøre eller udføre.” (UFM, n.d.-a). En færdighed er altså evnen til at kunne gøre bestemte handlinger.

En opøvet færdighed er med til at skabe indsigt i et område og er dermed med til at øge muligheden for akkomodativ læring. Derfor er det heller ikke tilstrækkeligt at få viden om det datalogiske område, hvis man skal få en forståelse for området. Det betyder, at det at programmere, må være en færdighed, der skal øves for at kunne bruge computational thinking på andre områder.

KOMPETENCER

Kompetence er et lidt diffust og bredt begreb, som der er mange definitioner af. Nordenbo beskriver begrebet, som rummende "en henvisning til noget, som individet gør eller kan gøre" (Nordenbo, 2011, pp. 52–53). Knud Illeris går i bogen *Kompetence. Hvad – Hvorfor – Hvordan,* tættere på begrebet. Han skriver, at de fleste definitioner af kompetence peger på "at det…drejer sig om at være i stand til at handle i relation til bestemte kendte, ukendte og uforudsigelige situationer" (Illeris, 2012, p. 35). Kompetencer kan dermed forstås, som de handlemuligheder den enkelte har.

En mere klar og anerkendt definition kommer fra OECDs (Organisationen for Economic Co-operation and Development) DeSeCo-projekt (Definition and Selection of Competencies), hvor Rychen og Salganik skriver at

> en kompetence defineres som evnen til på en vellykket måde at forholde sig til komplekse krav i en bestemt sammenhæng gennem en mobilisering af psykosociale forudsætninger (herunder både kognitive og ikke-kognitive aspekter). Dette indebærer en krav-orienteret eller funktionel tilgang til definitionen af kompetencer. Der er især fokus på de resultater, individet opnår gennem en handling, et valg eller en adfærd med hensyn til de krav, der f.eks. er knyttet til en bestemt professionel position, social rolle, eller et personligt projekt.(Illeris, 2012, p. 33; Rychen & Salganik, 2003, p. 43)

Det handler altså ikke kun om at kunne mestre og vide, men også om evnen til at kunne handle ud fra de forudsætninger den enkelte har opnået af viden og færdigheder. At opnå kompetence inden for programmering vil derfor give eleverne mulighed for at kunne handle i forhold til it som et grundvilkår, og på sigt gøre eleverne i stand til at både være kreative og opnå en tidssvarende dannelse, hvor det digitale får en stadig større rolle.

Kompetence kan opfattes som noget positivt, som de færreste vil sige nej til at få, uden at ordet har en egentlig betydning. Pia Bramming, afdelingsleder og lektor på DPU samt Ph.d. kalder kompetencer for *"et plusord!"* (Illeris, 2012, p. 31) og professor på SDU og dr.phil. Katrin Hjort betegner kompetenceudvikling "som et udtryk, hvis vigtigste betydningsindhold er, at det ikke betyder noget" (Illeris, 2012, p. 32). Når begrebet er utydeligt, skyldes det, at mange definitioner og forståelser er uklare og ofte modsatrettede (Illeris, 2012, p. 32).

I *Et stærkere og mere trygt digitalt samfund: den fællesoffentlige digitaliseringsstrategi 2016-2020* (Regeringen/KL/Danske-Regioner, 2016) tales der om digitale kompetencer og digital dannelse for børn og unge. Begreberne digital kompetence og dannelse bliver også diskuteret i forhold til 21st century skills, som vi kommer nærmere ind på i næste kapitel.

3.5 21ST CENTURY SKILLS

Siden årtusindeskiftet er 21st century skills kommet til at spille en større rolle i den uddannelsespolitiske debat (Berthelsen, 2017, p. 2)*. Vi ønsker derfor at kigge nærmere på begrebets oprindelse samt de forskellige organisationers bud på fremtidens kompetencer for dernæst at koble det til en dansk folkeskolekontekst og diskutere det ift. kompetence- og dannelsesbegrebet.*

21st century skills kan oversættes til det 21. århundredes kompetencer. Det engelsk ord skills betyder direkte oversat færdigheder, men da definitionerne på 21st century skills rummer mere end færdigheder og er tættere på definitionen af begrebet kompetence giver det bedst mening at oversætte til det 21. århundredes kompetencer.

Ifølge OECD defineres 21st century skills som "those skills and competencies young people will be required to have in order to be effective workers and citizens in the knowledge society of the 21st century" (Claro & Ananiadou, 2009, p. 8). I litteraturen møder vi også betegnelsen 21st century learning. Vi anvender og forstår både 21st century skills, 21st century learning og det 21. århundredes kompetencer ud fra OECD's definition.

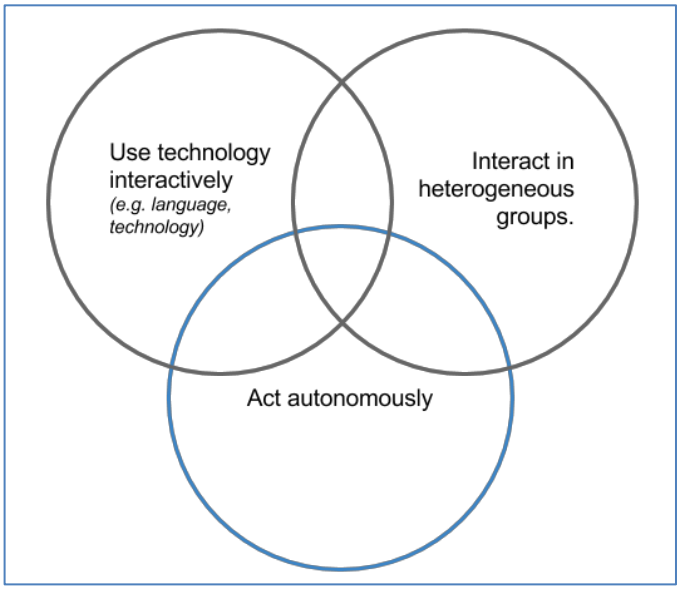
HVOR STAMMER 21ST CENTURY SKILLS FRA?

Tilbage i 80'erne i USA opstod der en erkendelse af, at det 20. århundredes kompetencer ikke længere var tilstrækkelige, hvis de amerikanske virksomheder skulle klare sig i den globale konkurrence i det 21. århundrede. Optimeret masseproduktion og specialiserede medarbejdere var ikke længere nok, hvis man ønskede økonomisk succes, og dette kaldte på handling (Berthelsen, 2017, pp. 3–5). En reform af uddannelsessystemet i 1991 beskrev således, hvordan omstillingsparathed, løbende kompetenceudvikling og livslang læring ikke kun var et krav til de få, men til alle uddannede borgere (Berthelsen, 2017, p. 5).

Derefter tog flere organisationer og private virksomheder initiativ til at identificere fremtidens nøglekompetencer (Berthelsen, 2017, p. 5). Deriblandt OECD, som satte gang i det internationale DeSeCo-projektet i 1997 (OECD, 2003), og den amerikanske interesseorganisation *Partnership for 21st Century Learning* (P21), der er et konsortium bestående af det amerikanske undervisningsministerium, Apple, Microsoft m.fl. (P21, n.d.-a). P21 har fokus på børn og unge på K-12 området, hvilket svarer til folkeskoleområdet i Danmark, samt implementeringen af 21st century skills i det amerikanske uddannelsessystem og er derfor relevant ift. vores projekt. ITL Research, der er et globalt forskningsprogram, som Microsoft Education står bag, kommer i 2011 også med et bud på fremtidens kompetencer i form af 21st Century Learning Design Rubricks (ITL Research, 2014). I nedenstående uddyber vi disse tre konsortiers definitioner og modeller for 21st century skills.

DESECOS BEGREBSRAMME FOR NØGLEKOMPETENCER

For at få en succesfuldt liv og et velfungerende samfund, hvor teknologien hele tiden ændrer sig og stiller krav til omstillingsparathed, har DeSeCo identificeret en række nøglekompetencer, som de klassificerer i tre brede kategorier (OECD, 2003)(Figur 17): *1) Use technology interactively, 2) Interact in heterogeneous groups og 3) Act autonomously.*



Figur 17: The DeSeCo Project's conceptual framework for key competencies (OECD, 2003, p. 5)

Kategori 1) *Use technology interactively,* handler ikke kun om at have adgang til teknologi og have færdigheder til at kunne betjene den. For at kunne bruge teknologien aktivt, kræver det både en fortrolighed samt en forståelse af, hvordan teknologi kan ændre den måde, hvorpå vi kommunikerer og interagerer (OECD, 2003, pp. 10–11). Kategori 2) *Interact in heterogeneous groups* handler om kompetencen til samarbejde, konfliktløsning og til at skabe relationer (OECD, 2003, pp. 12–13). Den sidste kategori 3). *Act autonomously* beskrives som kompetencen til at handle autonomt og selvstændigt (OECD, 2003, pp. 14–15).

DeSoCo's tre kategorier virker brede og generelle og kan virke svære at operationalisere i folkeskolen. Dette kan skyldes, at deres fokus er på borgere generelt og ikke på elever i grundskolen. Derfor vil vi se nærmere på Partnership for 21st Century Learnings Framework (P21) og derefter ITL Researchs seks rubrics, der er målrettet K-12.

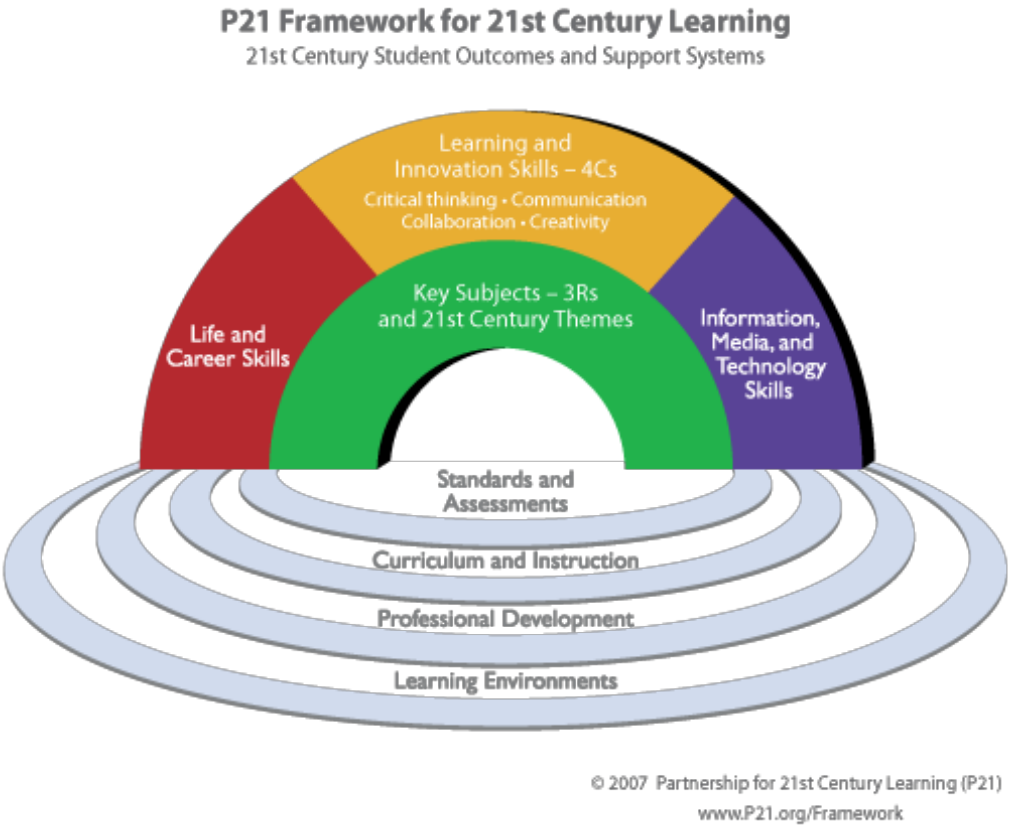
P21 FRAMEWORK FOR 21ST CENTURY LEARNING

P21 blev grundlagt i 2002 og hed oprindeligt *P21 Partnership for 21st Century Skills* og har siden skiftet navn til *Partnership for 21st Century Learning*. Deres mission er at være katalysator for 21st Century Learning på grundskoleområdet gennem kollaborative partnerskaber på tværs af uddannelse, samfundet, erhvervslivet og politikere "so that all learners acquire the knowledge and skills they need to thrive in a world where change is constant and learning never stops" (P21, n.d.-b).

P21s bud på 21st century skills forklares med udgangspunkt i deres regnbuemodel (Figur 18). Regnbuen med de fire farver repræsenterer læringsudbyttet for den studerende i det 21. århundrede og er den viden, færdigheder og ekspertise, eleverne skal beherske for at lykkes i det

21. århundrede. Modellen har beskrevet de forskellige områder særskilt, men de skal tænkes som tæt forbundet i den 21. århundredes undervisning og læring (P21, 2002).

P21 har opdelt deres bud på 21st century skills i temaerne: 1) *Life and Career, 2) Learning and Innovations samt 3) Informations, Media and Technology,* der illustreres som regnbuens yderste ring. Regnbuens grønne ring: *Key Subjects and 21st Century Themes,* handler om at mestre kernefag som fx sprog, matematik, økonomi, science samt *21st century themes* som global bevidsthed, entreprenørskab, sundhed m.fl. (P21, 2002).



Figur 18: P21 framework (P21, 2015)

Life and Career Skills beskriver kompetencer som fx fleksibilitet, tilpasningsevne, initiativ, selvstyring og ansvarlighed, der alle er vigtige for at kunne navigere i komplekse liv og arbejdsmiljøer.

Learning and innovations skills har fokus på kreativitet og innovation, kritisk tænkning og problemløsning, kommunikation og samarbejde (P21, 2015, p. 3). I 2015 udgiver P21 i samarbejde med forskere fra University of Connecticut desuden serien *4C's,* som udfolder kompetencerne kreativitet, kritisk tænkning, kollaboration og kommunikation (P21, n.d.-c)

Informations, Media and Technology learning skills er kompetencer, som gør eleverne kompetente til at navigere i et teknologi- og mediedrevet samfund, der er karakteriseret af uanede mængder af informationer og hurtige skift i udvalget og brugen af digitale værktøjer. *Informations literacy* stiller krav til, at man kritisk kan administrere og vurdere informationen samt forholde sig etisk og juridisk til brugen af disse informationer (P21, 2015, p. 5). *Media literacy* handler om at kunne analysere og forstå forskellige mediebudskaber samt at kunne skabe og bidrage med egne medieproduktioner. *ICT (informations, communications & technology) literacy* beskriver kompetencer, som er nødvendige for at kunne anvende teknologi effektivt som et redskab til at organisere, evaluere og kommunikere med (P21, 2015, p. 6).

De grå ringe i modellen (Figur 18) repræsenterer de supportsystemer, der er nødvendige, for at eleverne kan udvikle 21st century skills. Supportsystemerne består af *Standards and Assessments, Curriculum and Instruction, Professional Development og Learning Environment*. For at opnå resultater, skal der være fokus på bl.a. effektive summative og formative vurderinger i form af fx portfolio, innovative læringsmetoder, samarbejde med det omkringliggende samfund samt undersøgelses- og problembaseret læring, således at eleverne kan engagere sig i meningsfulde problemer (P21, 2015, p. 8).

21ST CENTURY LEARNING DESIGN RUBRICS

ITL Research har identificeret seks 21st century skills (Figur 19) og udviklet seks tilhørende rubrics eller nøgler, som skal hjælpe læreren med at identificere og forstå de muligheder, der er for at tilrettelægge læringsaktiviteter, således at eleverne opnår 21st century skills (ITL Research, 2014, p. 2). Hver nøgle repræsenterer en vigtig skill/kompetence, som eleven skal udvikle. Center for Undervisningsmidler (CFU) har udviklet websitet 21skills.dk, som har til formål, at *"vejlede undervisere, elever og studerende inden for den pædagogiske sektor, i arbejdet med det 21.århundredes kompetencer* (CFU, n.d.). Indholdet på 21skill.dk tager afsæt i 21st Century Learning Design Rubrics, som CFU mere eller mindre har oversat direkte til dansk (CFU, n.d.).

21st Century Learning Design Rubrics	21skill.dk
Collaboration	Kollaboration
Knowledge construction	Videnskonstruktion
Self-regulation	Selvevaluering
Real-world problem-solving and innovation	Problemløsning og innovation
The use of ICT for learning	It og læring
Skilled communication	Kompetent kommunikation

Figur 19: 21st Century Learning Design Rubricks og 21skills.dk

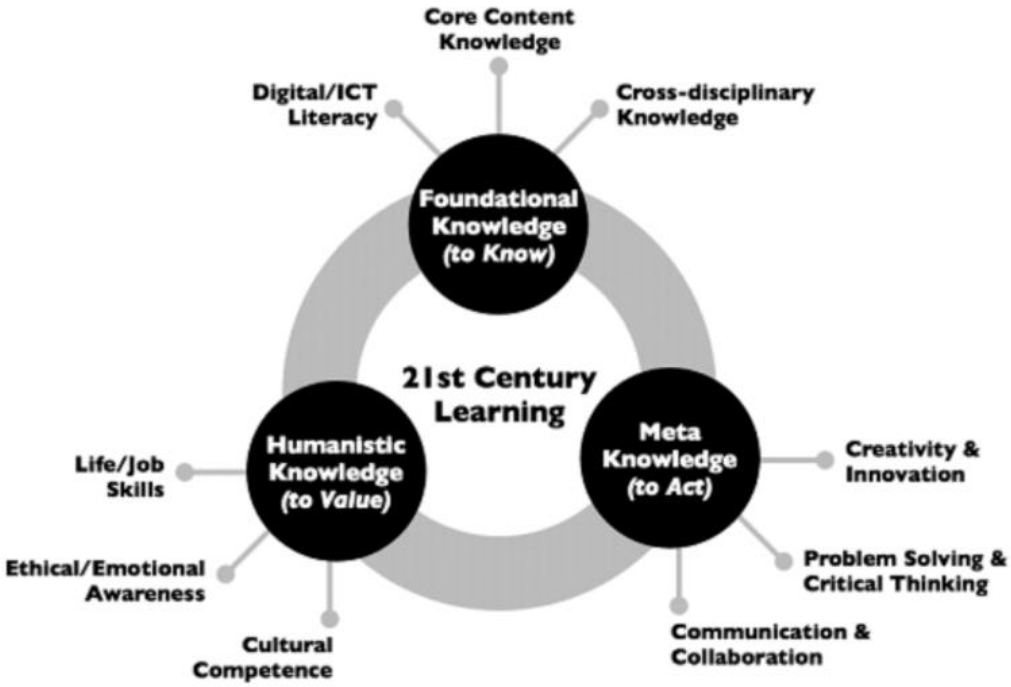
21ST CENTURY SKILLS I EN DANSK FOLKESKOLEKONTEKST

Udover Fælles Mål for fagene skal de tre tværgående temaer: *Innovation og entreprenørskab*, *Sproglig udvikling* og *It og medier* indtænkes i alle fag og er derfor beskrevet i introduktionen i fagenes Fælles Mål. It og medier som tværgående emne beskriver de fire elevpositioner *1) Eleven som kritisk undersøger 2) Eleven som analyserende modtager 3) Eleven som målrettet og kreativ producent* og *4) Eleven som ansvarlig deltager* (EMU, 2014b).

De fire elevpositioner har flere sammenfald med P21's definition af 21st century skills især inden for *Informations, Media and Technology* learning men også *Life and Career Skills*. P21's kategori af *Learning and Innovations Skills* er mere tydelig i det tværgående emne *Innovation og entreprenørskab* (EMU, 2014a). Der er også sammenfald mellem DeSeCo's kategori *Use technology interactively* og de fire elevpositioner, idet DeSeCo også vægter den producerende rolle i brugen af teknologi. ITL Research og dermed også 21skills.dk adskiller sig en smule med kompetencen self-regulation, men har på samme måde som P21 og DeSeCo flere overlap med de tværgående temaer.

FÆLLES MODEL

I den ovenstående gennemgang bliver det tydeligt, at de forskellige konsortiers definitioner af det 21. århundredes kompetencer både er forskellige og til dels uklare, men på samme tid er det også muligt at identificere en række gennemgående kompetencer. Det samme ser vi i andre artikler, som sammenligner de forskellige modeller og definitioner af 21st century skills. Kirsten Kereluik, forsker ved Michigan Virtual Research Institute at Michigan University, m.fl. har sammenlignet 15 forskellige modeller af 21st century skills herunder fx P21 og DeSeCo's modeller og er kommet frem til en syntese, der samler disse 15 modeller i en ny model (Kereluik et al., 2013, p. 130)(Figur 20)



Figur 20: Synthesis of 15 different 21st century learning frameworks into one visual image (Kereluik et al., 2013)

Kereluik m.fl. (2013) diskuterer også, hvorvidt der overhovedet er noget nyt og unikt ved 21st century skills. Vi skulle jo også være kreative, løse problemer, kommunikere og samarbejde i det 20. århundrede. På den anden side argumenterer de også for, at alt har ændret sig i forhold til undervisning og læring pga. teknologien. Digital/ICT literacy er derfor placeret under kategorien *Foundational Knowledge* på niveau med kernefagene, da det er vigtigt med en grundlæggende viden omkring anvendelsen af it, som har ændret måden, hvorpå vi interagerer, kommunikerer, samarbejder mm. i alle skolens fag (Kereluik et al., 2013, p. 132). Teknologien og den lette adgang til enorme mængder information har også ændret *Meta Knowledge (*Figur 20), og måden hvorpå vi fx tænker kritisk, løser problemer, kommunikerer og samarbejder er transformeret på grund af teknologi (Kereluik et al., 2013, p. 132). Til sidst har teknologien også indflydelse på *Humanistic Knowledge* og stiller krav om en højere grad af selvregulering og stillingtagen til etiske områder ifm. privatliv og ophavsrettigheder (Kereluik et al., 2013, p. 132).

DISKUSSION

21st century skills bliver kritiseret for at tage sit afsæt i arbejdsmarkedets behov i en amerikansk kontekst, hvor konsortier af private it-virksomheder og offentlige instanser har defineret modeller med forskellige sæt af kompetencer. Det er dog ikke nyt med uddannelsesfilosofiske bidrag, der forsøger at forstå fremtidens behov (Elf & Koed, 2017, p. 3). Dannelsesteorierne forsøger jo også at forstå og definere fremtidens behov, herunder hvilken viden vi skal besidde, samt hvilke kompetencer og arbejdsmetoder vi skal udvikle for at blive udviklings- og handlingsduelige både

nu og i fremtiden (Jank & Meyer, 2006, pp. 172–173). Forskellen er blot, at her er det private virksomheder, som er med til at definere visioner om fremtiden og sætte dagsordenen for børn og unges uddannelser.

Ifølge folkeskolens formål er skolens opgave ikke at tjene arbejdsmarkedets behov (Ministeriet for Børn Undervisning og Ligestilling, 2016) og producere "soldater til konkurrencestaten" (Berthelsen, 2017, p. 17). Frygten for at implementeringen af 21st century skills har dette sigte, er ifølge Berthelsen ubegrundet, hvis kompetencebegrebet betragtes mere bredt, som fx Kereluik m.fl. (2013) gør med kategorien *humanistic knowledge,* der har en række fællestræk med det, vi forstår som dannelse (Berthelsen, 2017, p. 13). P21's bud på *Life and Career Skills* og DeSeCos kategori *Acting Autonomously* har på samme måde fællestræk med dannelsesbegrebet. 21st century skills og dannelse har også det til fælles, at det er et mål for alle og ikke kun en afgrænset samfundsgruppe eller elite. Dannelse bliver udviklet i en kontekst af samfundsmæssige, kulturelle og politiske begivenheder (Jank & Meyer, 2006, p. 169), hvilke 21st century skills også er ifølge Berthelsen (Berthelsen, 2017).

21st century skills har endnu ikke samme politiske bevågenhed i Danmark, som det har i USA. Dette begrunder Berthelsen med, at anvendelsen af 21st century skills i folkeskolen primært er drevet af interesse fra undervisernes side af (Berthelsen, 2017, p. 11). Hvis vi ser bort fra Siri-kommissionens rapport (Siri-kommissionen, 2016), er 21st century skills ikke i samme grad koblet på den uddannelsespolitiske debat i Danmark, som det er tilfældet i USA, hvor begreber fra 21st century skills har været centrale ift. reformeringen af et tungt og på nogle områder ineffektivt skolesystem (Berthelsen, 2017, p. 11).

4 TEORIAFSNITTET

Ifølge Resnick giver programmeringsværktøjet Scratch gode muligheder for at eksperimentere med nye ideer, bryde komplekse ideer op i simplere dele samt finde og løse fejl. Scratch lærer også børn og unge at samarbejde og klare de frustrationer, der opstår, når der er noget, der ikke går efter planen (Resnick, 2012). Derfor vil vi inddrage teorier om erfaringsbaserede læreprocesser, hvor oplevelser og erfaringer spiller en central rolle i læringsprocessen.

ERFARINGSBASEREDE LÆREPROCESSER

Erfaringsbaserede læreprocesser kendetegnes ved, at den lærende er i direkte kontakt med den virkelighed, der studeres, modsat traditionel undervisning i klasselokalet, hvor den lærende primært læser om, taler om, skriver om denne virkelighed men aldrig kommer direkte i kontakt med den undervejs i processen (Kolb, 2015, p. xviii).

Den erfaringsbaserede læringsteori har til hensigt "at fremlægge et holistisk, integreret perspektiv på læring, som kombinerer erfaring, perception, kognition og adfærd" (Illeris, 2000, p. 48), og erfaringen spiller en central rolle i læreprocessen. Erfaringsbaseret læring tager dermed afstand fra de kognitive læringsteorier, der lægger hovedvægten på manipulering, tilegnelse og genkaldelse af abstrakte symboler, og de behavioristiske læringsteorier, der "afviser enhver betydning af bevidsthed og subjektiv erfaring i læreprocessen" (Illeris, 2000, p. 48).

4.1 ERFARINGSBASEREDE LÆRINGSTEORIER

Læringsteoretikeren David A. Kolb (1939) udgiver i 1984 bogen *Experiential learning - Experience as the Source of Learning and Development*, som bygger videre på tidligere teorier om erfaringsbaseret læring (Kolb, 2015). Inden vi præsenterer Kolbs læringscirkel, vil vi derfor kort præsentere tre af de læringsteorier, der har inspireret Kolb til sin teoriudvikling.

Inden for de erfaringsbaserede læringsteorier møder vi psykolog Jean Piaget (1898 - 1980), der er kendt for identificeringen af barnets fire hovedfaser af kognitiv vækst samt samspillet mellem akkommodation og assimilation i forhold til barnets læring (Illeris, 2000, p. 51). Dette samspil mellem akkommodation og assimilation er ikke altid symmetrisk (Jerlang, 1998, p. 238). Hvis assimilation dominerer læreprocessen, kan dette vise sig som leg. Hvis akkommodation dominerer, ses der tegn på imitation, hvor barnet forsøger at tilpasse sig til omgivelserne. Ifølge Piagets stadieteori befinder eleverne fra medieholdet sig i det 4. stadie: *det formelt-operationelle stadie* (ca. 11/12 – 15 år) (Jerlang, 1998, p. 246). Elever i denne alder kan nu i højere grad tænke abstrakt og reflekterende, hvilke gør det muligt at gennemføre hypotetisk-deduktive tankeoperationer (Illeris, 2000, pp. 52–53). Eleverne begynder derfor at kunne tænke i flere dimensioner på samme tid og arbejde med teorier og muligheder fx "hvis nu… så…"(Jerlang, 1998, p. 266).

Professor Kurt Lewin (1880 - 1947), kendt fra aktionsforskning og læring i laboratorieforsøg, fremhæver, hvordan konkrete her-og-nu oplevelser danner grundlag for den videre afprøvning og testning af de abstrakte begrebers gyldighed (Illeris, 2000, p. 49). Fokuspunktet for læring er dermed den umiddelbare personlige oplevelse. For Lewin er det centralt at integrere de personlige oplevelser med hensigtsmæssige feedback-processer i en effektiv og målrettet læreproces, hvor der er balance mellem observation og handling (Illeris, 2000, p. 49). Feedbacken skal fremadrettet være med til at ændre valg af nye oplevelser og adfærd.

Filosoffen John Dewey (1859 - 1952) beskriver læring "som en dialektisk proces der integrerer erfaring og begreber, observationer og handling" (Illeris, 2000, p. 50). I processen mod at nå sofistikerede og modne mål fremhæver Dewey vigtigheden af, at den lærende formår at udsætte de umiddelbare handlinger/blinde impulser for i stedet først at iagttage de omgivende vilkår og derefter sammenholde disse iagttagelser med tidligere erfaringer (Illeris, 2000, pp. 49–50).

4.2 HVAD KARAKTERISERER ERFARINGSBASERET LÆRING?

LÆRINGS SOM PROCES OG IKKE SOM RESULTATER

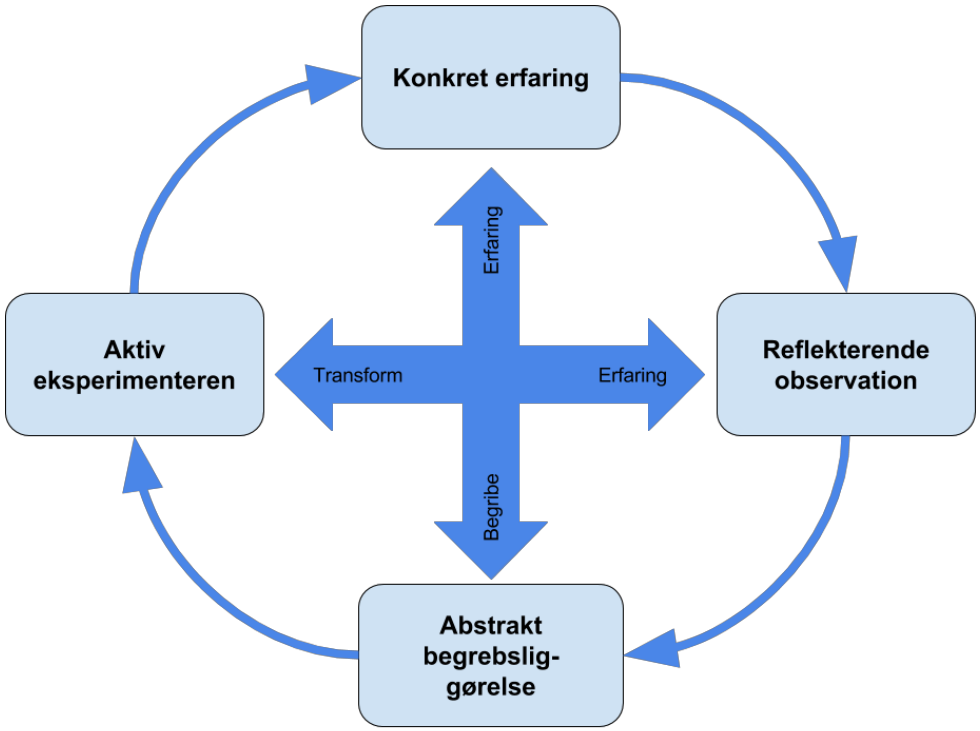
Kolb definerer læring således: ”Learning is the process whereby knowledge is created through the transformation of experience” (Kolb, 2015, p. 49). Erfaringsbaseret læring adskiller sig fra adfærdsorienterede læringsteorier og den traditionelle uddannelsestænkning ved at definere læring ud fra processen og ikke ud fra resultaterne (Illeris, 2000, p. 53). Tanken om at læring kan defineres ud fra resultater, der har form af akkumuleret fakta og vaner eller en mængde færdige idéer, ligger fjernt fra den erfaringsbaseret læring (Illeris, 2000, p. 53). Ifølge empiriske filosoffer skal idéer forstås som elementer af bevidsthed eller mentale atomer, der altid forbliver det samme (Kolb, 2015, p. 37). Det erfaringsbaserede læringsperspektiv tager afsæt i, at idéer hele tiden formes og omformes gennem en tilpasningsproces, hvor erfaringerne hele tiden spiller ind (Illeris, 2000, p. 54). ifølge psykologen Jerome Bruner (1915 – 2016) er formålet med uddannelse derfor, ”at stimulere til undersøgelser og færdigheder i at udvikle nye erkendelser, ikke at lære en mængde erkendelser udenad” (Illeris, 2000, p. 54).

LÆRING SOM EN SPÆNDINGS- OG KONFLIKTFULD PROCES

Både Lewin, Dewey og Piaget beskriver i deres teorier, hvordan læring sker gennem løsningen af konflikter mellem dialektisk modsatrettede måder at håndtere verden på. Piaget fremhæver konflikten mellem akkommodation og assimilation, og ifølge Dewey, opstår konflikten mellem impulsen, som giver drivkraft til idéerne, og fornuften, som skal give denne drivkraft mening. I Lewins erfaringsbaserede læringsmodel ses der to konflikter: konflikten mellem den konkrete oplevelse og de abstrakte begreber samt konflikten mellem handling og observation (Illeris, 2000, p. 56). Deres teorier og modeller peger dermed i retningen af, ”at læring i sin natur er en spændings- og konfliktfyldt proces” (Illeris, 2000, p. 57).

4.3 KOLBS LÆRINGSCIRKEL

Tydeligt inspireret af Lewin, Piaget og Deweys modeller, har Kolb udviklet nedenstående læringscirkel (Figur 21). Læringscirklen skal ikke forstås som en ”flad” cirkel, men som en læringsspiral, ”where… we return again to the experience and know it anew in a continuous recursive spiral of learning” (Kolb, 2015, p. 61).



Figur 21: Vores oversættelse af fig. 2.5 The Experiential Learning Cycle (Kolb, 2015, p. 51)

Kolbs forklarer med læringscirklen, hvilke fire forskellige læringsevner, den lærende skal gøre brug af for at opnå nye færdigheder, erkendelser og holdninger

1. **Konkrete erfaringer:** Den lærende skal – uden fordomme – kunne involvere sig åbent og fuldt ud i nye oplevelser/erfaringer (Illeris, 2000, p. 57).

2. **Reflekterende observation:** Den lærende "skal være i stand til at reflektere over og observere deres oplevelser fra mange perspektiver" (Illeris, 2000, p. 57). Dette kan give sig til udtryk som diskussioner og reflekterende spørgsmål, som får eleverne til at reflektere over deres hands-on oplevelser (Konak, Clark, & Nasereddin, 2014, p. 14).

3. **Abstrakt begrebsliggørelse**: Den lærende "skal være i stand til at skabe begreber, som integrerer deres observationer i logisk holdbare teorier" (Illeris, 2000, p. 57). Abstrakt begrebsliggørelse kan, ifølge Konk m.fl. (2014, p. 14) være svær at opnå i korte undervisningsforløb. Peer- og klassediskussioner kan være brugbare måder, hvorpå læringserfaringen kan kobles til holdbare teorier. Her spiller underviseren en vigtig rolle og kan hjælpe eleverne på vej ved fx at stille generaliserende spørgsmål (Konak et al., 2014, p. 14)

4. **Aktiv eksperimenteren**: Den lærende er i stand til at anvende teorierne fra *Abstrakt begrebsliggørelse*, når de skal planlægge den næste konkrete erfaring (Konak et al., 2014, p. 14) samt træffe beslutninger og løse problemer (Illeris, 2000, p. 57).

Kolbs læringscirkel (Figur 21) viser de to grundlæggende dimensioner i læringsprocessen. Den ene dimension har *Konkret erfaring* i den ene pol, og *Abstrakt begrebsliggørelse* i den anden pol. Den anden dimension har *Aktiv eksperimenteren* i den ene pol og *Reflekterende observation* i den anden. Udfordringen er, at læringsevnerne er modsatrettede og at det i praksis vil være svært fx at gøre sig konkrete erfaringer og på samme tid skabe begreber og holdbare teorier. Derfor skal den lærende være i stand til at vælge, hvilken læringsevne der passer i enhver given læringssituation. Den lærende placerer sig i spændet mellem to poler og bevæger sig i dette spænd, alt efter hvad læringssituationen kræver. Hvis en læringsevne dominerer læringsprocessen, kan læringen blive begrænset til det område, som kendetegner denne læringsevne (Illeris, 2000, p. 58). Dette kan forklares med, at vi udvikler vores læringsevner, eller som Kolb også beskriver dem: læringsstile, på forskellig vis, alt efter hvilke erfaringer vi har gjort os tidligere samt hvilke krav, der stilles i vores nuværende miljø (Konak et al., 2014, p. 37).

DISKUSSION

Kolbs læringscirkel bliver kritiseret for, at fremstille læring som en meget forenklet proces, der mekanisk sker trin for trin (Kolb, 2015, pp. 55–56). Kolbs læringscirkel kan ud fra denne kritik betragtes ud fra metaforen: en bustur, hvor turen bringer den lærende frem til nye erkendelser. Den lærende "står på" ved *Konkrete erfaringer* og rejser derefter videre og stopper ved de forskellige læringsevner og turen fortsætter derefter videre til nye erfaringer. Kolb forklarer, at han i starten også anvendte læringscirklen som en enkel og pragmatisk tilgang til at forklare de forskellige hændelser i erfaringsprocessen, hvormed erfaringer transformeres til læring (Kolb, 2015, pp. 55–56). Efter 50 års studier inden for erfaringsbaseret læring, har han udviklet sine forståelser omkring erfaringsbaseret læring og har videreudviklet sin teori på denne baggrund (Kolb, 2015, p. xviii). I ovenstående gennemgang af læringscirklen kommer der også en anden vinkel, hvor – hvis vi skal blive i bus-metaforen – kan se det som en rejse mellem forskellige poler, og hvor eleven rejser hen til den læringsevne, som passer bedst i den givne læringssituation. Bussen vil derfor også kunne køre på de blå pile i midten af Kolbs læringscirkel (Figur 21), og dermed virker hans læringscirkel mindre mekanisk.

Ifølge Piagets faseteori udvikles barnets evne til at tænke abstrakt i løbet af puberteten, hvor barnet er ca. 11/12 – 15 år (Jerlang, 1998). Nogle af medieholdets elever i 6. klasse vil muligvis ligge tættere på det forrige stadie: *den konkret-operationelle periode* (ca. 6/7 – 11/12 år), som kendetegnes ved, "at barnets måde at erkende på stadig afhænger af de konkrete handlinger med tingene" (Jerlang, 1998, p. 263). Ifølge Piaget skal barnet udvikles gennem de samme stadier dvs. barnet kan ikke springe et stadie over (Jerlang, 1998, p. 245). Piaget afviser dog, at man kan træne barnet, således at det kommer hurtigere gennem stadierne, med peger i stedet på "at man pædagogisk og socialt sikrer, at barnet får et stort 'udbytte' af hvert stadie, ved at give det mange handlemuligheder" (Jerlang, 1998, p. 245). I undervisningen er det altså lærerens rolle at give

barnet mange handlemuligheder samt at bygge videre på elevens egen praksis modsat at bygge videre på antagelser og ikke-observerbare ting (Jerlang, 1998, p. 271).

Når vi sammenholder Kolb's læringscirkel med Piagets faseteori, bliver det tydligt, at læreren spiller en vigtig rolle i forløbet, hvor eleverne skal programmere i Scratch. Læreren skal give eleverne mange handlemuligheder med udgangspunkt i deres egen praksis, således at de får et stort udbytte af hvert stadie. I vores tilfælde, hvor nogle elever befinder sig mellem to stadier, vil læreren opleve et stort spænd ift. denne opgave. Dette spænd give sig også til udtryk ift. læringsevnen *Abstrakt begrebsliggørelse,* der netop stiller krav til, at eleverne kan koble deres erfaringer med begreber og holdbare teorier, hvilket stiller krav til, at eleverne i højere grad kan tænke abstrakt og reflekterende (Illeris, 2000). Her kan læreren understøtte denne proces ved fx at stille generaliserende spørgsmål og gennem peer- og klassediskussioner, således at eleverne har et nyt afsæt til læringsevnen *Aktiv eksperimenteren*.

5 METODE OG EMPIRI

Til indsamling af empiri vil vi anvende kvalitative dataindsamlingsmetoder, da de egner sig til at beskrive erfaringsprocesser, oplevelser og det sociale liv (Brinkmann & Tanggaard, 2015), hvilket er centralt for vores projekt. Vi har indsamlet vores data via metoderne deltagerobservation, videoobservationer og interviews, og som forskningsmetodologi anvender vi grounded theory.

5.1 EMPIRI

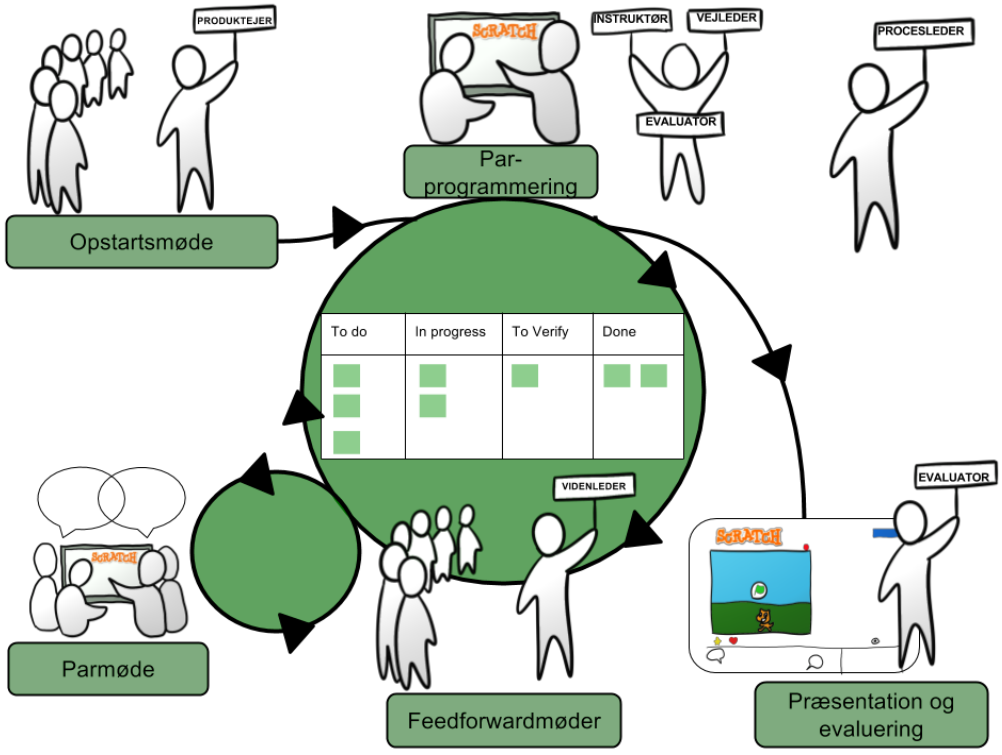
Vi bygger vores masterprojekt på empiri fra projekt #ProgrammeringNaturligvis, som er et fællesprojekt i Aalborg Kommune med 12 deltagende lærere fordelt på otte skoler (Aalborg Kommune, n.d.). I projektet gennemgår lærerne et opkvalificeringsforløb på UCN bestående af modulerne: 1) *Grundlæggende forståelse for programmering*, 2) *Arbejde med alternative grænseflader* 3) *Understøttelse af digitale kompetencer i Folkeskolen* og 4) *Design og produktion*. Derudover samles alle lærerne tre dage pr. halvår til opfølgning, gruppevejledning og videndeling.

Til gruppevejledning og videndelingsdagen i december 2016 præsenterede vi vores IKT-didaktiske design fra kernemodul 4 *IKT og didaktisk design* (Figur 22) for lærerne, med henblik på feedback og senere afprøvning i praksis, da vi på dette tidspunkt havde planer om, at vores masterprojekt skulle have fokus på et re-design af det IKT-didaktiske design i samarbejde med en lærer og klasse. Siden har vores projekt og problemformulering udviklet sig i en anden retning, og vi har derfor ikke længere det samme fokus.

Læreren, i den 6. klasse vi indsamler empiri fra, er på nogle punkter inspireret af det IKT-didaktiske design, herunder parprogrammering og brugen af task board. Derfor vil vi kort beskrive tankerne bag vores IKT-didaktiske design samt parprogrammering og task boardet i det næste afsnit, da det har indflydelse på vores indsamling og analyse af data.

IKT-DIDAKTISK DESIGN

Vores IKT-didaktiske design (Figur 22) tager udgangspunkt i programmering i Scratch, og til udviklingen af modellen fik vi inspiration af andre modeller fx Gynther & Christiansens undervisningsloop (Gynther, 2010, p. 84) samt den iterative HCI Lifecycle model (Preece, Rogers, & Sharp, 2002, p. 186), som vi fik præsenteret i kernemodul 2 *IKT og interaktionsdesign*. Derudover er vi også inspireret af agil softwareudvikling, hvor principper som iterative processer med løbende og hyppige leveringer er centrale (Fowler & Highsmith, 2001) samt *eduScrum* (Delhij, van Solingen, & Wijnands, 2015), der er en undervisningstilgang, der bl.a. benyttes på skoler i Holland. EduScrum er kraftigt inspireret af den agile tilgang Scrum. Scrum er udbredt inden for softwareudvikling og er en metode og et værktøj, som bl.a. har til formål at gøre det lettere at samarbejde og løse komplekse opgaver (Delhij et al., 2015).



Figur 22: IKT-didaktisk design

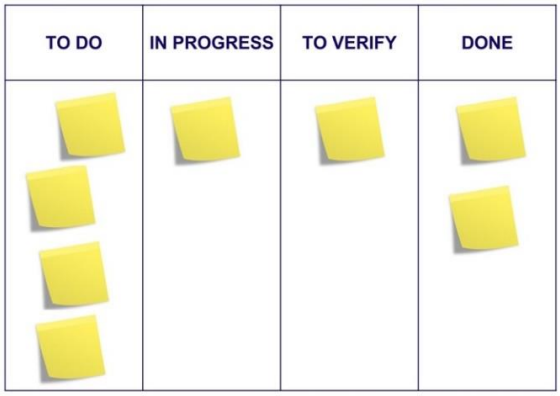
PARPROGRAMMERING

Parprogrammering har været anvendt indenfor softwareudvikling siden 70'erne, og inden for de sidste 10 år har parprogrammering også vundet indpas på uddannelsesområdet (Hanks, Fitzgerald, McCauley, Murphy, & Zander, 2011). Ved parprogrammering sidder programmørerne sammen to og to ved én computer og samarbejder om at løse forskellige problemer. Mens den ene har rollen som *driver* og dermed styrer tastatur og mus, har den anden rollen som *navigator*, der sidder ved siden af og kommer med forslag til forbedringer og identificerer eventuelle fejl (Lewis, 2011, p. 105). Parprogrammering har en række fordele som fx øget kollaboration, det at hjælpe og få hjælp af makkeren samt beskrive og forklare løsninger til makkeren (Lewis, 2011, p. 107). Derudover viser studier også, at de studerende i højere grad er glade for parprogrammering end soloprogrammering og giver udtryk for, at parprogrammering er sjovt og brugbart til forskellige opgaver (Hanks et al., 2011, p. 144).

TASK BOARD

Task boardet er et stykke karton, som er opdelt i fire kolonne (Figur 23): *To Do:* Opgaver som skal løses. *In progress*: Opgaver de er i gang med at løse. *To verify:* Løste opgaver, der skal testes af andre. *Done:* Opgaver der er løst og godkendt gennem test.

Hvert par har deres eget task board, som skal sikre transparens (Delhij et al., 2015) og fungere som et kommunikativt redskab for lærer og elever i hele perioden. Parret opsplitter udviklingen af spillet i små delopgaver, hvilket understøtter konceptet *decomposition* i computational thinking (Barefoot Computing, 2014). Delopgaverne beskrives på hver deres post-it og sættes på task boardet under *to do*. Fx: lave/finde en baggrund, finde en Sprite, få Spriten til at flytte sig på baggrunden og optælling af point. Læreren kan på forhånd definere en række forslag til delopgaver, som eleverne kan bruge som inspiration til deres egne delopgaver. Parret beslutter selv, hvornår en delopgave skal flyttes fra én til en anden kolonne, om en delopgave skal fjernes eller en ny tilføjes.



Figur 23: Task board

HVORDAN UNDERSØGER VI PRAKSIS?

Forud for indsamlingen af empirien havde vi et møde med læreren, hvor vi i fællesskab lavede aftaler om indhold, antal gange og tidspunkter for vores observationer samt afstemme vores roller. Vi gennemgik også de tekniske detaljer omkring indsamlingen af empiri, og læreren sørgede for indsamling af forældrenes tilladelser, således at disse formaliteter var på plads, inden vi påbegyndte indsamlingen af empiri.

Empirien blev indsamlet over en periode på 14 dage fordelt på fem undervisningsgange á 1,5 times varighed. Den første gang præsenterede læreren forløbet for eleverne herunder indhold, læringsmål og andre praktiske detaljer. Derefter gik eleverne på sitet Hour of Code (Hour of Code, n.d.), hvor de oprettede en profil og valgte et blokprogrammeringskursus, som de arbejdede med individuelt. Anden gang arbejde eleverne med en række trin-for-trin opgaver, hvor de fik viden og færdigheder om forskellige funktioner, som de senere kunne bruge til udviklingen af deres eget spil. De resterende tre gange arbejdede eleverne med at programmere deres spil.

UDVÆLGELSE AF GRUPPER

På medieholdet var der i alt 17 elever, som blev sammensat i grupper á to elever dvs. otte grupper i alt. Vi havde ingen præferencer til sammensætningen af grupperne til parprogrammering, så læreren besluttede selv, hvordan eleverne skulle sættes sammen. Eleverne fik dog mulighed for at ønske, hvem de ville være makker med.

Undervejs i dataindsamlingsprocessen var der fire grupper, der skilte sig ud af forskellige årsager.

GRUPPE E: bestod af en pige og en dreng, der arbejdede stille og roligt med opgaverne og anvender task boardet undervejs i processen.

GRUPPE G: bestod af to drenge, der arbejdede hurtigt og effektivt med opgaverne, men som ikke altid parprogrammerede eller anvendte task boardet.

GRUPPE F: bestod af to drenge, der næsten ikke anvendte task boardet og som til tider mistede overblikket.

Til sidst fik vi også fået øje på GRUPPE H: der bestod af to piger, hvor især den ene pige udviklede sig meget undervejs i forløbet.

Vi sætter derfor et særligt fokus på de fire grupper i det videre arbejde med data, idet grupperne spænder bredt ift. deres kompetencer inden for programmering samt ved, at de repræsenterer forskellige gruppesammensætninger på tværs af køn.

Under vores videre bearbejdelse af videoobservationerne sprang to af de udvalgte grupper os særligt i øjnene, som to grupper der var i hver sin ende af det læringsmæssige spekter. Det var

gruppe F, som entusiastisk og energisk arbejdede med deres spil på en ustruktureret måde, og som løb ind i flere og flere problemer undervejs og virkede til at tabe pusten lidt til sidst. Gruppe F brugte desuden meget få ord, når de samarbejdede om spillet. I modsætning hertil var gruppe E, to dygtige og stabilt arbejdende elever, der brugte mange ord og først kom i problemer, da de kastede sig ud i at bruge komplicerede funktioner i Scratch. Vi valgte, på baggrund af deres forskellige brug af begreber og arbejdsmetoder, at tage et nærmere kig på disse to grupper.

5.2 ETISKE OVERVEJELSER

I forbindelse med indsamling og behandling af empiri er etiske og juridiske aspekter et vigtigt punkt, som vi skal tage hensyn til "lige fra begyndelsen af en undersøgelse til den endelige rapport" (Kvale & Brinkmann, 2014, p. 107). Alle deltagere skal underskrive en kontrakt, hvor det nøje er beskrevet og præciseret, hvordan empirien bliver indsamlet og anvendt (Brinkmann & Tanggaard, 2015, pp. 95, 104). I vores case var deltagerne ikke myndige, derfor skulle forældrene underskrive samtykkeerklæringen (bilag 4). Da vi undervejs i indsamlingen af empirien besluttede os for at gennemføre et supplerende interview med fire udvalgte grupper, har disse gruppers forældre underskrevet en ekstra samtykkeerklæring (bilag 5).

I samtykkeerklæringen beskriver vi bl.a., at videooptagelserne vil blive slettet efter vores eksamen, samt at skolens og elevernes navne bliver anonymiseret i projektet. På denne måde beskyttes deltagernes fortrolighed (Kvale & Brinkmann, 2014, pp. 105–123). Vi informerer også om, at deltagelsen er frivillig, samt at deltagerne til hver en tid kan trække sig ud af undersøgelsen.

Som forskere er vi også opmærksomme på vores etiske ansvar i forhold til at være tro over for de valgte metode til indsamling og behandling af data (Corbin & Strauss, 2015, p. 13). Vi skylder deltagerne at være forberedte, systematiske og engagerede gennem hele projektet og dermed undlade fx at tage genveje og sjuske undervejs i processen (Corbin & Strauss, 2015, pp. 13–14). Som forskere har vi også et ansvar for at videregive resultaterne i vores projekt i relevante forum (Corbin & Strauss, 2015, p. 14) fx Skoleforvaltningen, UCN og gruppen af lærere i projekt #ProgrammeringNaturligvis. Vi tilbyder også at dele resultaterne med læreren, elevernes forældre og deltagerne i vores eliteinterview.

5.3 VIDEOOBSERVATION

Når man ønsker at studere menneskers handlinger og praksisser i forskellige sammenhænge, er videoobservation en central forskningsmetode (Brinkmann & Tanggaard, 2015). Videoobservationer foregår *in situ* og har "fokus på de direkte aflæselige træk i situationen, dvs. deltagerens interaktion med det materielle og sociale miljø" (Brinkmann & Tanggaard, 2015, p. 98) og dermed ikke på deltagernes indre. Vi er interesseret i, hvad eleverne siger, men vi er også

opmærksomme på, "hvordan de konkret brugte deres krop og de materielle omgivelser, når de taler eller foretager sig andre 'ordløse handlinger'" (Brinkmann & Tanggaard, 2015, p. 103). Videooptagelserne gør det muligt for andre, som ikke var til stede i selve situationen, at deltage i observationen efterfølgende (Brinkmann & Tanggaard, 2015), hvilket var en fordel for os, da vi skiftevis havde mulighed for at observere selve undervisningen.

Inden eleverne gik i gang med at programmere i Scratch på deres computer, aktiverede de programmet Screencastify (Screencastify, n.d.) med følgende indstillinger: Optag aktiviteter på skærmen, anvend indbygget mikrofon og indlejr billede fra integreret webcam. Dette muliggjorde, at vi efterfølgende kunne se, hvad de foretog sig i Scratch, samtidig med at vi både kunne se deres interaktioner med hinanden og høre deres tale. Af hensyn til videooptagelsernes lydkvalitet, blev grupperne fordelt i klasselokalet og tilstødende grupperum således, at de ikke sad for tæt på hinanden. En fordel ved dette stationære setup til videoobservationerne er, at vi ikke kommer "for sent" til en scene (Brinkmann & Tanggaard, 2015, p. 101), hvilket kunne være tilfældet, hvis vi havde optaget med håndholdte kamera og var i gang med at observere en anden scene. På den anden side er skærmoptagelsen også mindre fleksibel og kan ikke følge med rundt i lokalet, som et håndholdt kamera ville kunne. Skærmoptagelsen virkede langt hen ad vejen som en naturlig del af begivenhederne (Brinkmann & Tanggaard, 2015, p. 105). Vi observerede, at både eleverne, læreren og vi selv glemte, at optagelserne var i gang, da selve optagelsen stort set var usynlig efter igangsættelsen. Efter optagelsen uploadede grupperne optagelsen i en Google-mappe, som var delt med os. Dette muliggjorde, at vi både løbende og afsluttende havde adgang til deres videooptagelser.

5.4 DELTAGEROBSERVATION

Ud over videoobservationerne har vi også foretaget deltagerobservationer. Deltagerobservationer giver mulighed for at få en intuitiv forståelse af det indsamlede materiale, hvilket kan minimere forhastede og kontekstomme konklusioner (Brinkmann & Tanggard, 2015, s. 86). Deltagerobservationer gør det også muligt at indfange hændelserne, mens de sker og kan dermed åbne op for interaktioner mellem aktørerne, det situerede og det ikke italesatte (Brinkmann & Tanggard, 2015, s. 86). Ordet deltagerobservation består af to modsatrettede begreber. *At deltage*, hvor man aktivt involverer sig og deltager i aktiviteterne kontra *at observere*, hvor man betragter praksis på afstand. Forskeren skal derfor være i stand til at udføre begge dele og bevæge sig i spændet mellem at deltage og observere (Brinkmann & Tanggaard, 2015, p. 83).

Vi deltog på et niveau, hvor vi bevægede os rundt i klasselokalet, observerede og tog noter, og vi interagerede naturligt med eleverne, når de henvendte sig til os. På den måde forsøgte vi at undgå, at eleverne følte sig som forskningsobjekter (Brinkmann & Tanggaard, 2015, p. 95). Som forberedelse til deltagerobservationerne havde vi udviklet et skema til vores feltnoter (bilag 6),

som vi løbende noterede i, mens vi observerede. Ulempen ved at tage noterne undervejs er, at det kan opleves som en hæmsko for forskeren, samt give deltagerne en oplevelse af at blive overvåget (Brinkmann & Tanggaard, 2015, p. 92). Omvendt er fordelen ved at tage noterne undervejs, at de ikke bliver en "rekonstrueret beretning af en situeret oplevelse, hvor det kan være vanskeligt at få øjeblikkets detaljerigdom med" (Brinkmann & Tanggaard, 2015, p. 92).

Fordelen ved deltagerobservationerne er, at de giver "forskeren et indblik i de normative og kulturelle koder, som præger de fleste fællesskaber" (Brinkmann & Tanggaard, 2015, p. 86), hvilket kan forberede forskeren til at stille de "rigtige" spørgsmål i fx et interview. Dette er relevant for os, idet vi efterfølgende vil supplere vores video- og deltagerobservationer med interviews.

5.5 INTERVIEW

På baggrund af vores deltager- og videoobservationer ønsker vi at supplere vores data med semistrukturerede interviews med de fire grupper, for at få en bedre forståelse af deres handlinger, samspil og oplevelser. Vi har forberedt vores interviewundersøgelse med udgangspunkt i Kvale og Brinkmanns syv faser, hvoraf vi kort gennemgår de tre første (Kvale & Brinkmann, 2014, pp. 151–157). Fase fire – syv bliver en del af vores afsnit om grounded theory.

FASE 1: TEMATISERING

Formålet med vores interview med de fire grupper var at supplere og berige data fra vores deltager- og videoobservationer med elevernes egne refleksioner omkring parprogrammering, brugen af task board og deres tanker omkring programmering generelt.

FASE 2: DESIGN

Forud for interviewet har vi udarbejdet et skema til de semistrukturerede interviews, som vi gennemførte med grupperne enkeltvis. Vi har inddelt spørgeguiden i emnerne *Kendskab til Scratch, Task board og Parprogrammering* med tilhørende spørgsmål (bilag 7). Vi har holdt vores spørgsmål korte og enkle og tilpasset dem til elevernes ordforråd og opfattelsesevne (Kvale & Brinkmann, 2014, p. 189)

FASE 3: INTERVIEW

Interviewet blev gennemført med hver enkelt gruppe i et tilstødende grupperum og blev videooptaget. Den ene af os var den primære interviewer, der anvendte spørgerguiden, og den anden supplerede med uddybende spørgsmål, som dukkede op undervejs i interviewet.

5.6 ELITEINTERVIEW

Vi ønsker at foretage skriftlige eliteinterviews med en række eksperter på området omkring programmering og computational thinking i forhold til folkeskolen og fælles mål til at supplere tilgængelig litteratur på området. Vi har valgt at spørge Brian Ravnborg Christensen, CFU-konsulent og med i arbejdsgruppen, der skrev Fælles Mål til naturfagene i 2014 og undervisningsministeren Merete Riisager samt Jakob Harder, vicedirektør for Styrelsen for It og læring (STIL).

Ved eliteinterview er det især vigtigt, at intervieweren har et godt kendskab til emnet for at få respekt og opnå en vis symmetri i interviewrelationen (Kvale & Brinkmann, 2014, p. 201). Qua vores masterstudie, vores uddannelser, nuværende jobfunktioner samt involvering i projekt #ProgrammeringNaturligvis, mener vi at have et grundigt kendskab til området og derfor kan stille relevante og kvalificerede spørgsmål til de udvalgte eksperter. Eksperter kan, ifølge Kvale og Brinkmann (2014), være vant til at blive interviewet og dermed have forberedte svar, derfor har vi været opmærksomme på at stille åbne spørgsmål og spørgsmål, hvortil de skal komme med deres egen holdning og kigge ind i fremtiden. Vi har sendt en præsentation af os selv samt en kort beskrivelse af vores projektet, hvori vi beskriver formålet med det skriftlige interview (bilag 8). Vi har medsendt en tilpasset interviewguide til hver enkelt ekspert (bilag 9 & 10).

5.7 OPSAMLING PÅ METODERNE

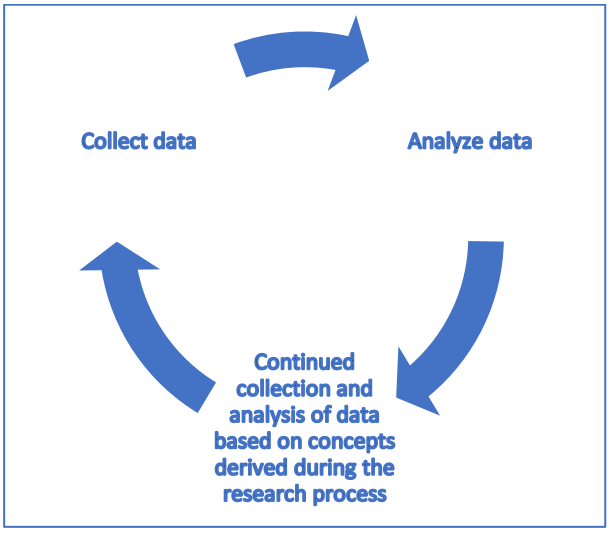
Deltagerobservationerne repræsenterer vores umiddelbare indtryk, som vi har noteret, mens det fandt sted i undervisningssituationen. Disse har været værdifulde til at få et indtryk af, hvad der sker i undervisningen og dermed et godt afsæt til vores videre analyse. Videoobservationerne har den styrke, at de giver et indblik i, hvad eleverne foretager sig på skærmen, og hvordan de taler med hinanden, mens de programmerer. Data fra vores interviews består af elevernes refleksioner, og de har til formål at berige og supplere data fra både deltager- og videoobservationer.

Vi er opmærksomme på, at den indsamlede data har forskellige styrker og dermed skal behandles og kombineres med omtanke i vores analyse. I analyseafsnittet vil vi derfor markere tydeligt, hvilke datakilder vi benytter hvornår og med hvilket formål.

5.8 GROUNDED THEORY

Grounded theory er en kvalitativ forskningstilgang, der blev udviklet af de amerikanske sociologer Barney G. Glaser og Anselm Leonard Strauss tilbage i 1967. Grounded theory har til formål at udvikle en teori eller elementer til en teori, med udgangspunkt i empirisk data dvs. at projektet som udgangspunkt ikke indledes med en teori (Brinkmann & Tanggaard, 2015, p. 242).

I grounded theory er det centralt, at data bliver indsamlet og analyseret i en forbundet og iterativ proces, hvilket kan illustreres med nedenstående Figur 24:



Figur 24: "Interrelationship Between Data Collection and Analysis" (Corbin & Strauss, 2015, p. 8)

Analysen starter, når indsamlingen af den første data er afsluttet. I analysen udforskes data og relevante begreber identificeres, og i forlængelse af denne analyse indsamles ny data. Denne proces betegnes som teoretisk sampling (Corbin & Strauss, 2015, p. 69). Den teoretiske sampling gentages, indtil forskeren har opnået en "mæthed", hvilket i praksis kan afhænge af fx den tid, der er til rådighed eller af en eventuel begrænset periode, hvor det er muligt at indsamle empiri (Corbin & Strauss, 2015, p. 136). Fordelene ved teoretisk sampling er, at det bliver muligt for forskeren løbende at opdage begreber, udfylde huller og gå i dybden med dem (Corbin & Strauss, 2015, pp. 136–137).

Centrale analysestrategier i grounded theory er at stille spørgsmål til data, sammenligne data og finde ligheder og forskelle i data (Corbin & Strauss, 2015, p. 90). Undervejs i processen kan forskeren gøre forskellige notater i form af memoer, der kan indeholde refleksioner, idéer og andre indtryk som lyde, lugt mm. (Bryman, 2008, p. 547)(Brinkmann & Tanggaard, 2015, p. 262). Ifølge Corbin og Strauss interagerer forskeren med data og analyserer, mens memoerne bliver lavet (Corbin & Strauss, 2015, pp. 106–107).

KODNING

Den mest centrale proces i grounded theory er kodningen, som kan inddeles i disse tre trin: 1) Åben kodning, 2) Aksekodning og 3) Selektiv kodning (Bryman, 2008, p. 543).

ÅBEN KODNING

Den første kodning har til formål at besvare spørgsmålet "What's going on?" (Brinkmann & Tanggaard, 2015, p. 246). Den første kodning er derfor åben og fri og minder om en brainstorm (Corbin & Strauss, 2015, p. 69). Data nedbrydes i mindre dele fx sætninger, som navngives/kodes med forskellige begreber. Disse begreber kan derefter samles i kategorier, for at gøre det mere overskueligt (Brinkmann & Tanggaard, 2015, p. 249). Fx kan begreberne *idéer, hjælper og stiller spørgsmål* samles i kategorien *samarbejde*.

Vi har samlet og behandlet vores data fra både deltagerobservationer, videoobservationer og interviews i programmet Nvivo (QSR International, n.d.). Nvivo egner sig til at samle komplekse datasæt og gør det muligt at kode på tværs af forskellige datatyper (AAU, n.d.), hvilket vi har behov for i vores projekt. Ifølge Corbin & Strauss (2015) er der mange fordele ved at bruge programmer som Nvivo i analyseprocessen. Udover at det er muligt at arbejde med og organisere flere forskellige datakilder, gør Nvivo det også muligt at arbejde kreativt med forskellige visninger af data (Corbin & Strauss, 2015, pp. 204–205). Nvivo kan nemlig let fremstille forskellige visuelle diagrammer i processen, som gør det muligt for forskeren at holde styr på begreber, kategorier og de indbyrdes forbindelser (Corbin & Strauss, 2015, p. 123). Ved brugen af disse programmer i analysen, har Corbin & Strauss (2015) følgende råd: "Analysis is about thinking, and thinking is the one thing the computer can't do yet" (Corbin & Strauss, 2015, p. 205).

AKSEKODNING

Åben kodning og aksekodning går hånd i hånd i analyseprocessen. Hvor den åbne kodning nedbryder data til små håndterbare dele, samler aksekodningen dem igen og skaber forbindelser på tværs (Corbin & Strauss, 2012, p. 5). Corbin og Strauss beskriver dog, at denne opdeling kan virke kunstig og kun har til formål at forklare processen, hvori data først bliver opdelt for derefter at blive samlet igen. I analyseprocessen vil forskeren nemlig helt naturligt skabe disse forbindelser (Corbin & Strauss, 2012, p. 5). Vi anvender dog alligevel betegnelsen aksekodning til at beskrive den del af kodeprocessen, hvor vi samler og kategoriserer begreberne, da vi i Nvivo har tilstræbt, at den åbne kodning skal være så åben og fri som muligt, for dernæst at samle begreberne i kategorier under aksekodningen.

SELEKTIV KODNING

Under den selektive kodning udvælges kernekategorier, også kaldet centrale kategorier, som derefter undersøges nærmere. En kernekategori kendetegnes ved at optræde hyppigt i data og relatere sig til de andre større kategorier (Corbin & Strauss, 2015, p. 189).

Memoer kan hjælpe med visualiserer de forskellige kategoriers fylde samt vise, hvordan kategorierne relaterer sig til hinanden (Corbin & Strauss, 2015, p. 189). Til udvælgelsen af vores

kernekategorier, har vi kombineret vores memoer med visuelle diagrammer fra Nvivo samt vores undersøgelsesspørgsmål.

DATAINDSAMLINGS- OG ANALYSEPROCESSEN TRIN FOR TRIN

For overskuelighedens skyld har vi beskrevet hele vores dataindsamlings- og analyseproces trin for trin i nedenstående skema.

TRIN	AKTIVITET	BESKRIVELSE
1	Deltagerobservation og videoobservationer	• Tager feltnoter og gennemser videoobservationer • Analysen påbegyndes • Teoretisk sampling
2	Interviews	• Interview gennemføres med udgangspunkt i spørgeguiden.
3	Samle de forskellige datakilder i Nvivo	• Opretter et Nvivoprojekt til hver gruppe • Samler data fra deltager- og videoobservationer samt interviews
4	Åben kodning – 1. runde	• Gennemser alle videoobservationer og skriver detaljerede noter dertil • Opretter memoer til vores noter • Transskriberer interviews • Koder noterne fra videoobservationer og de transskriberede interviews • Koder feltnoter fra deltagerobservationen
5	Åben kodning – 2. runde	• Transskriberer udvalgte områder på baggrund af koderne fra 1. runde af åben kodning • Koder transskriberingerne
6	Aksekodning	• Samle alle gruppernes projekter i ét Nvivo projekt • Kategoriserer begreberne fra den åben kodning • Opretter visuelle diagrammer i Nvivo ud fra forskellige kriterier
7	Selektiv kodning	• Kernekategorier identificeres og udvælges vha. memoer, visuelle diagrammer og undersøgelsesspørgsmål
8	Udvikling af teori	• Der udvikles en teori eller dele af en teori ud fra kernekategorierne.

Figur 25: Oversigt over vores dataindsamlings- og analyseproces trin for trin

TRANSSKRIBERING

Vi har selv foretaget kodningen og transskriberingen af vores data i programmet Nvivo dvs. ikke anvendt automatisk kodning i Nvivo eller benyttet os af andre løsninger ift. transskriberingen, da dette arbejde og de fund vi gør os undervejs er værdifulde og en del af analysen.

I transskriberingen har vi benyttet os af følgende tegn:

[tekst] = forklaring af ord, som er underforstået i sætningen.
tekst… = eleven gør ikke sætningen færdig, holder en pause eller siger noget, som ikke er relevant for konteksten.
(tekst) = beskriver hændelser på skærmen og elevernes handlinger foran webcam.

Derudover navngives eleverne med gruppebetegnelse, køn og nummer, fx Fd3 (F for gruppe, d for dreng og 3 for dreng nummer tre).

5.9 LITTERATURSØGNING

For at få et overblik over mængden af litteratur, der er tilgængelig omkring emnerne programmering i folkeskolen, computational thinking, 21st century skills og digital dannelse, har vi gennemført en systematisk og gennemsigtig litteratursøgningsproces. Denne proces startede på MIL-seminariet januar 2017, hvor vi fik et kursus i litteratursøgning. Her fik vi færdigheder inden for bloksøgning, trunkeringer og boolske operatorer (og, eller, ikke, and, or, not) samt indsigt i relevante søgedatabaser. Vi arbejdede systematisk i processen vha. en søgeprotokol (bilag 11), som vi udfyldte med først danske og derefter engelske søgeord, valg og fravalg af databaser samt antal resultater. Derefter har vi udvalgt og indsat links til de relevante søgeresultater. Efterfølgende bookede vi en ny vejledning med en bibliotekar på Aalborg Universitetsbibliotek, som bidrog med flere søgeteknikker- og metoder samt forslag til nye søgeord og databaser. Dette resulterede i udarbejdelse af nye søgeprotokoller (bilag 12, 13, 14 & 15), og den samlede mængde søgeresultater har været afsættet til den videre læse- og skriveproces og vores literature review i kapitel 1.4.

5.10 METODEKRITIK

Som vi skrev i kapitel 5.7 *Opsamling på metoderne*, er vi opmærksomme på, at vi har forskellig typer af data og derfor også må være omhyggelige med, hvornår vi bruger de forskellige kilder, og hvad kilderne bliver brugt til.

Vi har taget noter til og transskriberet forskellige kilder. Da vi har forskellige forforståelser og indsigter qua vores baggrund og erfaringer, er vi klar over, at vi hver især kan komme til at påvirke

resultatet gennem vores eget bias (Kvale & Brinkmann, 2014, p. 314). Vi har gennem opmærksomhed og dialog diskuteret forståelsen og indholdet af koder og kategorier for at formindske den forudindtagethed, vi hver især har præget bearbejdelsen og analysen med. Vi er dog også opmærksomme på, at vi undervejs i processen, hvor vi har bearbejdet og analyseret data, får flere fælles holdepunkter, som kan påvirke os. De detaljerede noter og den åbne kodning bliver bevidst og ubevidst præget af vores forskellige fokusser, da den bliver foretaget over en længere periode, hvor faktorer som vejledning, gruppemøder, litteraturlæsning og projektskrivning har indflydelse på de "briller" vi ser på data med.

Vi er ikke interesseret i at generalisere ud fra vores empiri (Kvale & Brinkmann, 2014, p. 332), men er interesseret i at kvalificere debatten om programmering i folkeskolen. Vores empiri består af en stor mænge data af forskellig typer, som det ses af Figur 26. Figur 26 har ikke til formål at bidrage med en kvantitativ vinkel, men for at dokumentere at validiteten gennem diversiteten og omfanget er stor, ikke mindst da data er indsamlet på forskellige måder og over et tidsrum på 14 dage og med observationer samt interviews fra i alt 5 forskellige dage (Kvale & Brinkmann, 2014, p. 328).

Empiri/Kilder/Ressourcer	Antal	Tid i minutter	Tid i timer, minutter, sekunder
Deltagerobservationer (5 forskellig dage)	5	375 minutter	6 timer + 15 minutter + 0 sekunder
Anvendte videoobservationer (4 forskellige dage)	14	493 minutter	8 timer + 13 minutter + 5 sekunder
Alle videoobservationer (4 forskellige dage)	31	906 minutter	15 timer + 29 minutter + 24 sekunder
Interview (1 dag)	5	35 minutter	0 timer + 35 minuttet + 10 sekunder
I alt	41	1316 minutter	21 timer + 56 minuttet + 34 sekunder

Figur 26: Omfanget af kilder

Ved at gennemse alle videoer og gennemgå vores deltagerobservationer har vi udvalgt, hvilke videoobservationer vi ville se nærmere på, og hvem vi ville interviewe, selvfølgelig med den risiko at vi kan have overset vigtige detaljer. I Figur 27 ses den bearbejdede data. Dog mener vi, at vi ved at have udvalgt 14 videoobservationer, som repræsenterer otte elever og over halvdelen af tiden af videoobservationsoptagelser, at validiteten og reliabiliteten af vores empiri fastholdes (Kvale & Brinkmann, 2014, p. 318).

Name	Nodes	Reference	Created On	Created By
04.04.17deltagerobservationer	7	8	11-05-2017 09:13	ALCL
21.03.17 deltagerobservationer	10	14	11-05-2017 09:13	ALCL
28.03.17deltagerobservationer	22	26	11-05-2017 09:13	ALCL
30.03.17deltagerobservationer	38	70	11-05-2017 09:13	ALCL
31.03.17deltagerobservationer	31	68	11-05-2017 09:13	ALCL
E Apr 4 2017	77	382	11-05-2017 09:13	ALCL
E Mar 28 2017	34	114	11-05-2017 09:13	ALCL
E Mar 30 2017a	51	312	11-05-2017 09:13	ALCL
E Mar 30 2017b	57	350	11-05-2017 09:13	ALCL
E Mar 31 2017	85	862	11-05-2017 09:13	ALCL
F Apr 4 2017	40	180	11-05-2017 09:14	AEMA
F Mar 28 2017	80	953	11-05-2017 09:14	AEMA
F Mar 30 2017	39	180	11-05-2017 09:14	AEMA
F Mar 31 2017	67	788	11-05-2017 09:14	AEMA
G Mar 28 2017	41	242	11-05-2017 09:15	AEMA
H Apr 4 2017	78	534	13-05-2017 16:03	ALCL
H Mar 28 2017	89	544	13-05-2017 16:03	ALCL
H Mar 30 2017	52	296	13-05-2017 16:03	ALCL
H Mar 31 2017	82	832	13-05-2017 16:03	ALCL
Interview E	10	24	11-05-2017 09:13	ALCL
Interview Fa	31	102	11-05-2017 09:14	AEMA
Interview Fb	17	36	11-05-2017 09:14	AEMA
Interview G	33	92	11-05-2017 09:15	AEMA
Interview H	32	120	13-05-2017 16:03	ALCL

Figur 27: Benyttede kilder

Vi har ikke transskriberet alle godt og vel otte timers videoobservationer, da vi 1) ikke har haft tiden til det og 2) mener, at vi gennem omfattende notater, memoer og kodninger har kunne udvælge vigtige passager til transskribering. Af nedenstående Figur 28 ses omfanget af noter og transskriberinger.

Data til analyse	Antal
Antal rækker af noter til videoobservationer	106 rækker
Antal rækker af transskriberinger i videoobservationer	487 rækker
Antal af referencer i videoobservationer	2748 reference
Antal ord i noter og transskriberinger	Ca 17.000 ord
Antal rækker af transskriberinger i interviews	391 rækker
Antal af åbne koder (nodes)	173 koder
Antal af akse koder(nodes) - kategoriseringer	15 kategorier

Figur 28: Omfanget af bearbejdet data til analyse

Vi har altså gennem diversiteten af vores data og med blik for vores egne forforståelser og forudindtagethed forsøgt at analysere gennem dialog, med indsigt og med udgangspunkt i vores videnskabsteoretiske ståsted, som vi vil redegøre for i det næste kapitel.

5.11 VIDENSKABSTEORI

Vi har et socialkonstruktivistisk ståsted og forstår den samfundsmæssige og menneskelige virkelighed gennem konstrueret videnskabelig viden (Collin, 2004, p. 23), der er skabt af mennesket og samfundet i kollektive menneskelige processer (Collin, 2011, p. 250). Det vil sige at de videnskabelige teoriers indhold er helt eller overvejende bestemt af den sociale proces, hvorigennem de er opstået. De er dermed afspejlinger af det samfund, i hvilket de er fremkommet, snarere end af den virkelighed, der er deres genstand (Collin, 2011, p. 251).

Vores erkendelse af verden er konstrueret, og vi er kollektivt med til at lave konstruktioner, hvor igennem vi forstår verden. Vi har konstrueret vores empiri og data gennem vores design og metoder til dataindsamling og gennem de fælles processer med os som forskere, eleverne på valgholdet i 6. kl. og elevernes lærer.

Vi fortolker vores empiri og data hermeneutisk. Hermeneutik er en humanistisk videnskabsteoretisk tilgang, der betyder meningsfortolkning. I modsætning til naturvidenskaben, hvor man søger at forstå naturfænomener, søger man i hermeneutikken "en dyberegående forståelse af menneskelig aktivitet og produkterne af sådanne aktiviteter, og sådanne fænomener har mening, thi de har deres udspring i mennesker, der mener og vil noget" (Pahuus, 2011, p. 140).

Så når vi fortolker gennem en hermeneutisk tilgang, er formålet ”at nå frem til en gyldig og fælles forståelse af en teksts mening” (Kvale & Brinkmann, 2014, p. 80). Tekst skal forstås i bred forstand, som skreven ”tekst, tale (sproglige fænomener), andre udtryksformer, handlinger samt evt. begivenheder og sagforhold” (Jacobsen, Schnack, Wahlgren, & Madsen, 2009, p. 165).

Vi har som forskere og mennesker en forforståelse af verden, som vil være med til at påvirke vores fortolkning og forståelse af hver en tekst. Vi kan ikke isolere os vores undersøgelsesfænomen og må inddrage vores forforståelse for at opnå indsigt i vores empiri. Menneskets forforståelser og i visse sammenhænge fordomme kommer af, at vi er historiske væsner, der er betinget af de erfaringer, vi har gjort, de traditioner vi har levet i og det samfund og den kultur, vi kommer af (Kvale & Brinkmann, 2014, p. 80).

Inden for den hermeneutiske tradition for fortolkning af en tekst, kan man benytte sig af flere principper, som ikke er trin-for-trin-metoder, men i stedet en række ”generelle principper, som har vist sig brugbare i en lang tradition for fortolkning af tekster”(Kvale & Brinkmann, 2014, p. 276). I Kvale & Brinkmanns bog *Interview* (2014) opstilles 7 principper, hvor særligt første princip, der omhandler den hermeneutiske cirkel, er central:

1. ”Det første princip gælder den kontinuerlige proces frem og tilbage mellem dele og helhed” (Kvale & Brinkmann, 2014, p. 275). For at forstå og fortolke en teksts helhed må man fortolke delene, som igen fortolkes ud fra helheden. Forståelsen og fortolkningen af delene er afhængig af forståelsen og fortolkningen af delene. Denne vekselvirkning mellem dele og helhed anses i den hermeneutiske tradition som positiv, da det giver en stadig dybere meningsforståelse gennem en kontinuerlig spiral eller cirkularitet.
2. ”Et andet princip er, at meningsfortolkning slutter, når man er nået frem til…en indre enhed i teksten”(Kvale & Brinkmann, 2014, p. 275). Det vil sige at forståelsen af teksten kan stå uden logiske modsigelser.
3. Et tredje princip er, at delfortolkninger testes ”i forhold til tekstens globale mening og eventuelt også i forhold til lignende tekster af samme forfatter” (Kvale & Brinkmann, 2014, p. 275).
4. Et fjerde princip er, at ”teksten skal forstås ud fra sin egen referenceramme” (Kvale & Brinkmann, 2014, p. 275). Det betyder, hvad teksten selv siger om temaet.
5. Et femte princip er, at teksten forstås fra viden om temaet (Kvale & Brinkmann, 2014, p. 275).
6. Et sjette princip er, at tekstens fortolkning kun kan ske med udgangspunkt i fortolkerens historie eller forståelsestradition. En fortolker kan ikke fjerne sig fra sin historie eller forståelsestradition, da fortolkeren lever midt i dette. Man kan dog som fortolker forsøge at være så eksplicit om sin forståelsestradition, som det er muligt, ved at være bevidst om at formuleringer af spørgsmål til en tekst er medbestemmende for formerne for svarene (Kvale & Brinkmann, 2014, p. 275).

7. ”Et syvende princip siger, at enhver fortolkning medfører fornyelse og kreativitet” (Kvale & Brinkmann, 2014, p. 275). Meningen udvides ved hver fortolkning, da enhver fortolkning frembringer ”nye differentieringer og indbyrdes relationer i teksten” (Kvale & Brinkmann, 2014, p. 275) – ”Jedes Verstehen ist ein Besserverstehen” (Kvale & Brinkmann, 2014, p. 275).

Vi er opmærksom på vores forskellige forforståelser og forståelsestraditioner. Vi forsøger at udnytte de forskellige forståelsestraditioner som en styrke, når vi gennem dialog komplementerer hinanden, fx når vi på vores empiri kan fortolke fx koden *slette* ind i både kategorien *Programmering* og *Problemløsning*. Dermed bevæger vi os rundt i den hermeneutiske cirkel, hvor vi forsøger at sætte os ind i og forstå hinandens forståelseshorisonter, for derigennem at udfordre vores egne forståelser og i fællesskab skabe en fælles forståelseshorisont (Jacobsen et al., 2009, p. 170), og vi udvider meningen, når vi kan pege på indbyrdes relationer imellem observationerne.

Gennem vores arbejde med dette projekt har vi desuden fået en fælles forståelsesramme, som vi også har bragt med som en fælles forforståelse. Vi fortolker os frem til en indre enhed og forstår teksten ud fra sin egen referenceramme bl.a. ved at fortolke videoobservationer gennem fortolkningen af deltagerobservationer og interviews.

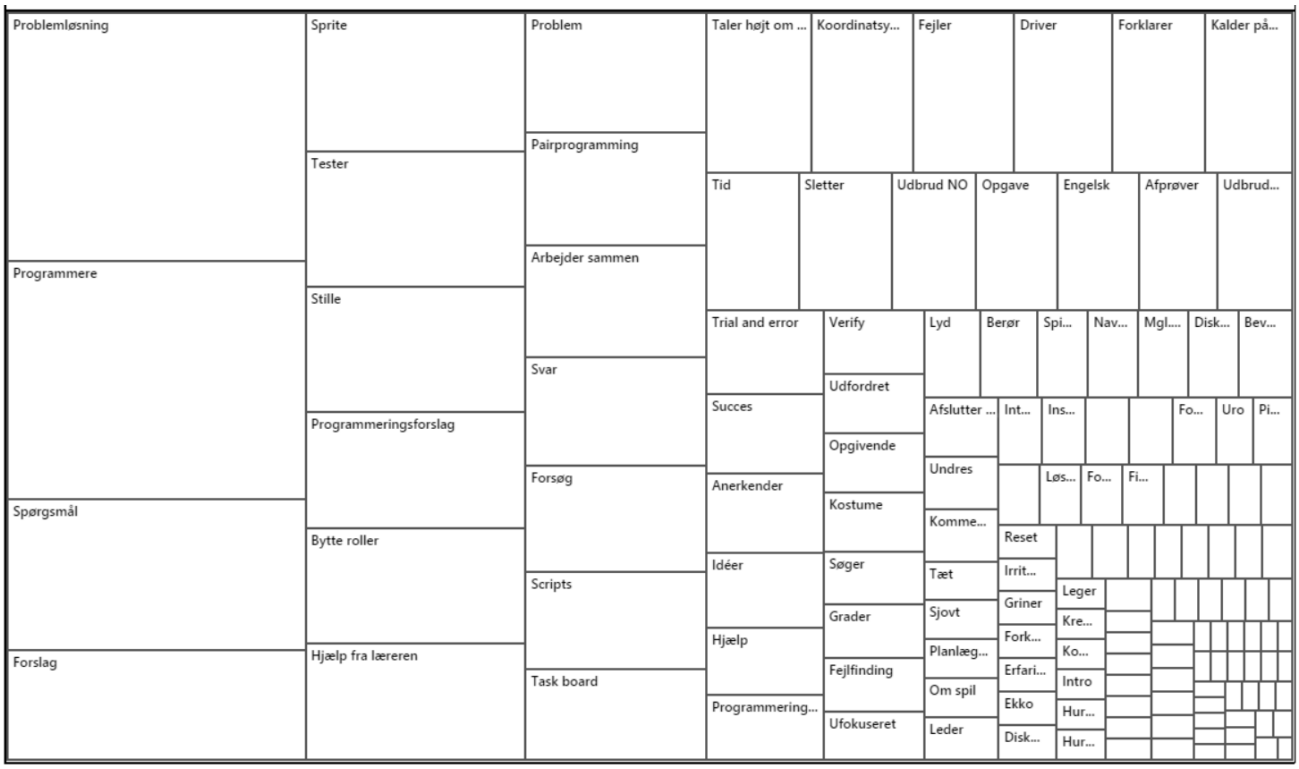
Vores videnskabsteoretiske ståsted har indflydelse på vores valg af kvalitative metoder til indsamling af data. Grounded theory har også direkte sammenhæng med den hermeneutiske cirkel, da vi i en kontinuerlig proces bevæger os frem og tilbage mellem begreber og kategorier, for at forstå og fortolke data og dermed opnå en dybere meningsforståelse.

6 ANALYSE

Vores analyse består af fire dele, hvor vi i den første del giver en oversigt over vores fund under den åbne kodning og aksekodningen fra de fire grupper. I den anden del af analysen besvarer vi undersøgelsesspørgsmål ***1. I hvilket omfang udvikler eleverne 21st century skills, når de programmerer i Scratch****? med udgangspunkt i resultatet fra aksekodningen. Derefter besvarer vi undersøgelsesspørgsmål* ***2. I hvilket omfang udvikler eleverne computational thinking, når de programmerer i Scratch?*** *Til sidst besvarer vi undersøgelsesspørgsmål* ***3. I hvilket omfang udvikler eleverne digital dannelse, når de programmerer i Scratch?*** *med udgangspunkt i elevernes refleksioner fra de gennemførte interviews.*

6.1 WHAT'S GOING ON?

Efter den første åbne kodning af data fra de fire grupper, har vi samlet alle koder på tværs af alle datakilder og lavet nedenstående visuelle diagram (Figur 29). Størrelsen på firkanterne viser, hvor mange gange koden er anvendt, dvs. jo flere gange en kode er anvendt, desto større er firkanterne. Diagrammet giver os et indblik i *What's going on?*, som den åbne kodning egner sig godt til (Brinkmann & Tanggaard, 2015, p. 246).



Figur 29: Åben kodning alle grupper

Diagrammet viser tydeligt, at det, der er blevet kodet flest gange, er flg. begreber: problemløsning[4], programmere, spørgsmål, forslag, Sprite, tester, stille, bytte roller, programmeringsforslag, hjælp fra læreren, problem, parprogrammering, arbejder sammen, svar og programmeringsforslag. Det skal bemærkes, at vi i den åbne kodning efterhånden stoppede

[4] Problemløsning: "det at løse et væsentlig, uafklaret anliggende ved at forholde sig velovervejet til det gennem analyse og tankevirksomhed" ("Problemløsning," n.d.)

med at kode samarbejde[5] og kommunikation[6], da dette skete hele tiden, mens eleverne programmerede. Alternativet var, at vi skulle dobbeltkode langt de fleste sætninger med samarbejde og kommunikation, hver gang de fx kom med en idé, stillede et spørgsmål eller svarede hinanden.

Ovenstående diagram er et godt afsæt til den følgende aksekodning, hvor vi samler alle begreberne i forskellige kategorier.

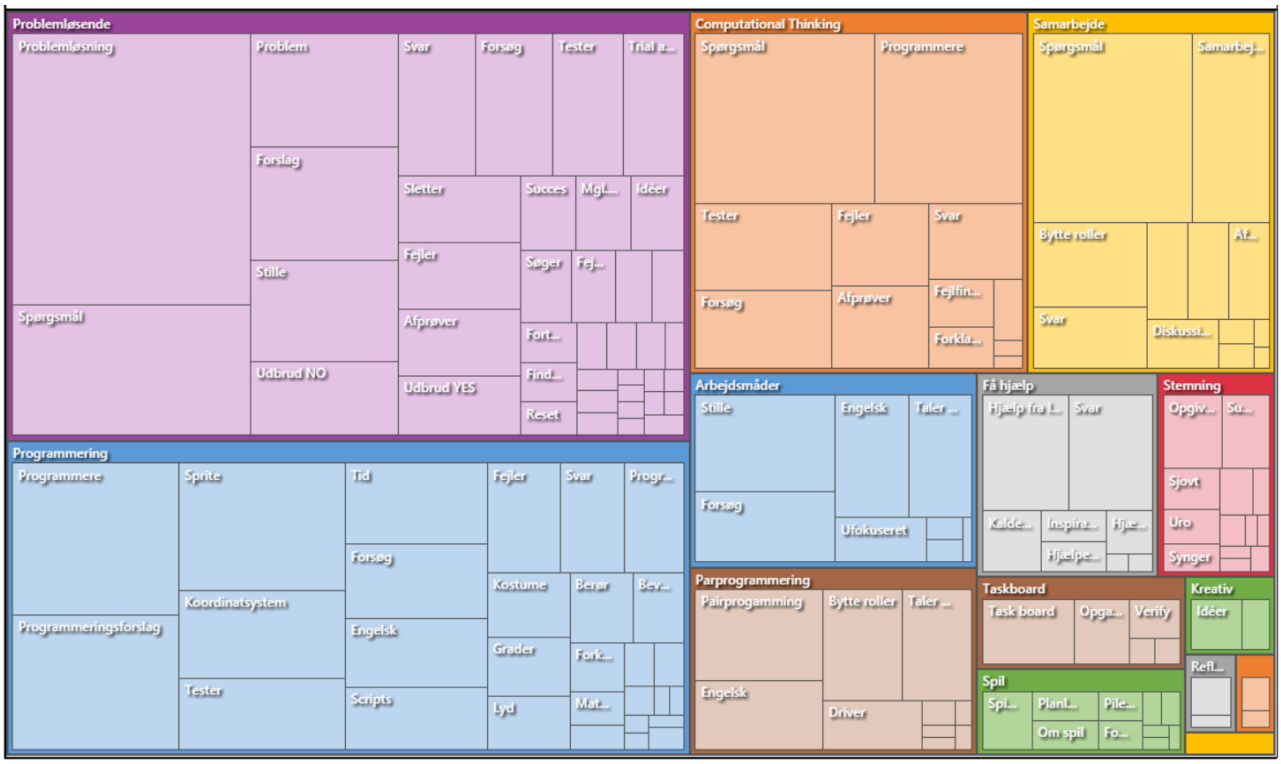
AKSEKODNING

I dette afsnit arbejder vi kun med koder fra videoobservationerne, dvs. vi har frasorteret koder fra deltagerobservationer, som er vores egne observationer og data fra interviews, som er elevernes refleksioner.

Hvor den åbne kodning deler data op i mindre håndterbare dele, samle aksekodningen disse begreber på ny og skaber forbindelser på tværs (Corbin & Strauss, 2012, p. 5). Efter aksekodningen har vi genereret nedenstående diagram (Figur 30). Hver kategori får automatisk tildelt en farve, og der er ikke sammenhæng mellem fx kategorien *Programmering* og *Arbejdsmåder*, selv om de begge har farven blå.

[5] Samarbejde: ”arbejde som flere personer, organisationer el.lign. udfører i fællesskab også om det at indgå i et sådant arbejde” (“Samarbejde,” n.d.)

[6] Kommunikation: ”Ved K forstås en meddelelse, en information af idéer eller følelser fra én person eller gruppe til en anden. I K-processen er der altid tale om en afsender, et budskab og en modtager” (Wiingaard, 2007).



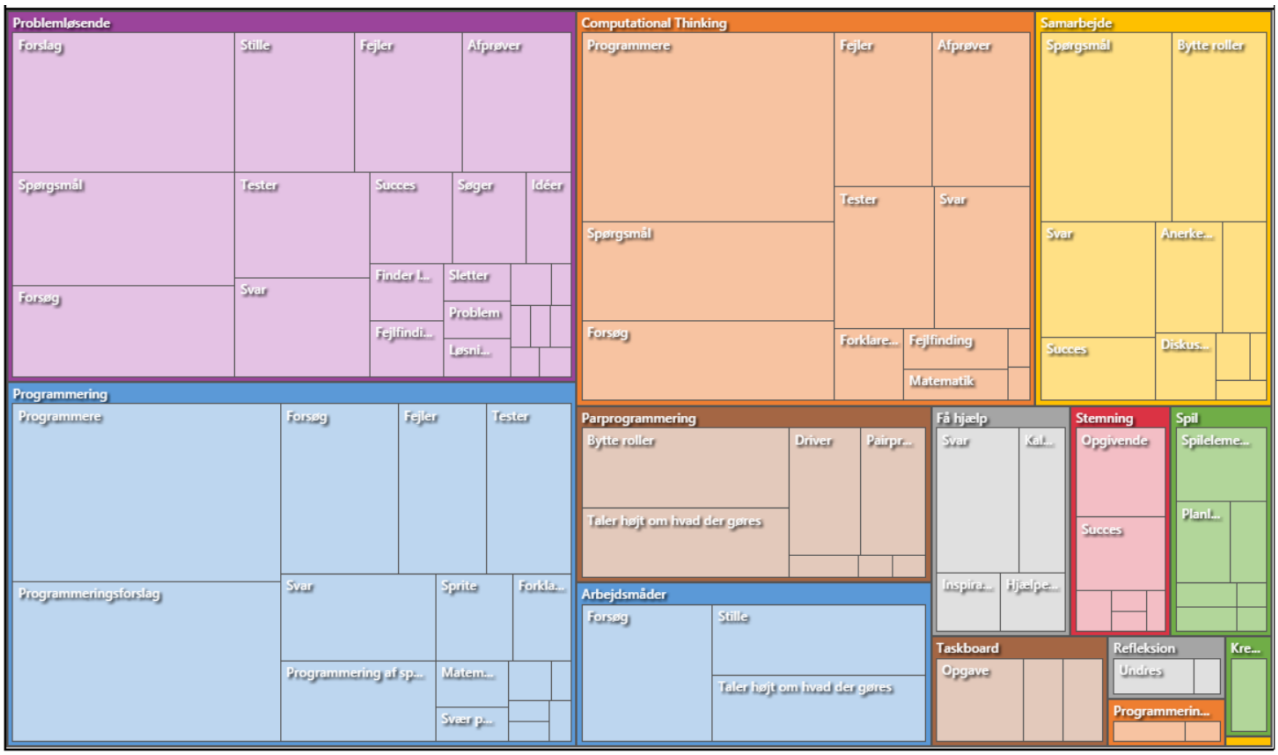
Figur 30: Aksekodning af videoobservationer de fire grupper

Problemløsende er den største kategori og består af begreberne: problemløsning, spørgsmål, svar, forslag, tester, problem, forsøg, fejler, forklarer m.

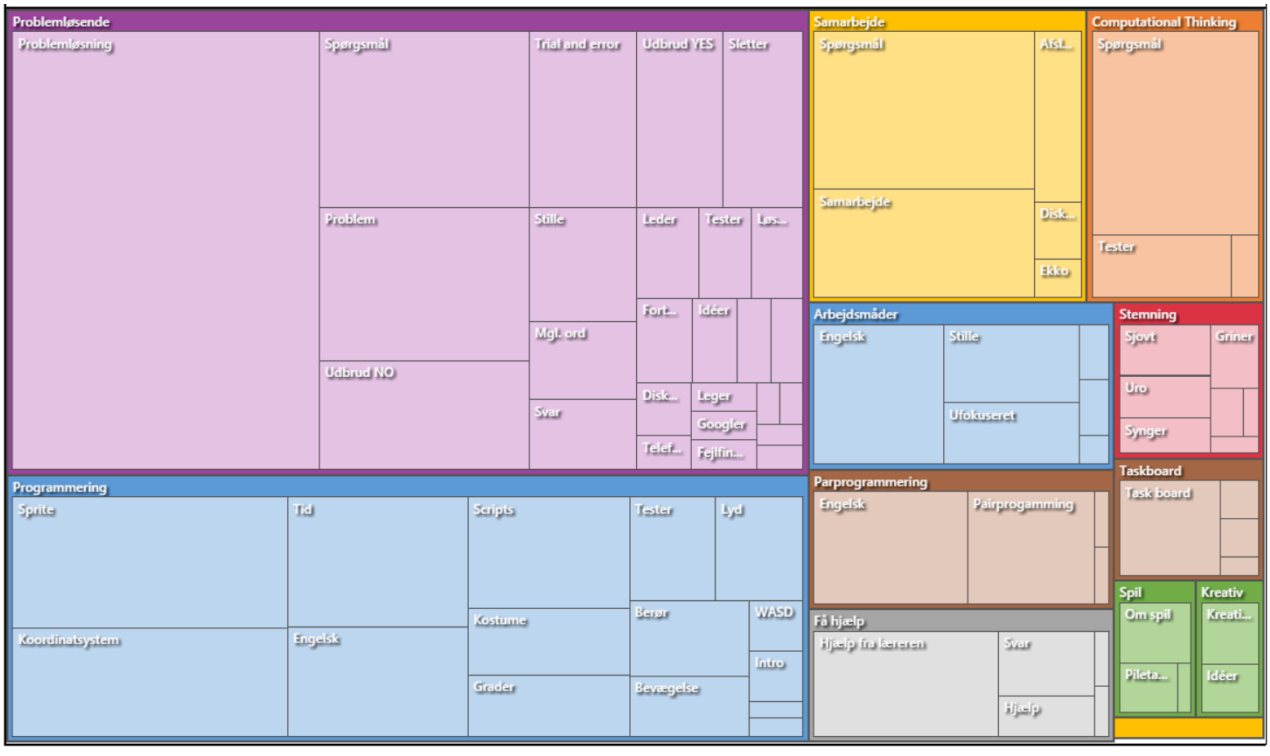
Derefter kommer kategorien *Programmering*, som består af begreberne: programmere, Sprite, programmeringsforslag, forsøg, fejler, tester, svar, koordinatsystem, tid m.fl. Den tredjestørste kategori er *Computational Thinking*, som består af begreberne spørgsmål, programmere, tester, fejler, svar, forsøg, afprøver, fejlfinding, forklare programmering m.fl.

Vi ser en del forbindelser mellem kategorierne. Fx har *Problemløsende* forbindelse til kategorien *Computational thinking* ved at de begge indeholder begreberne spørgsmål, svar, forsøg og fejler. *Programmering* har også forbindelser til *Computational Thinking* fx programmere og forsøg.

Vi har lavet yderligere to udtræk fra ovenstående aksekodning på gruppe E og F, idet de adskilte sig fra de andre grupper under vores deltagerobservationer og i den åbne kodning. I Figur 31 ser vi, hvordan gruppe E arbejder stort set lige meget med *Problemløsende* og *Programmering*. Kategorierne *Computational Thinking, Samarbejde* og *Parprogrammering* fylder også en del. I Figur 32 viser diagrammet, hvordan gruppe F's kategori *Problemløsende* fylder en del mere end i gruppe E. Derefter kommer kategorierne *Programmering* og *Samarbejde.*



Figur 31: Aksekodning gruppe E



Figur 32: Aksekodning gruppe F

UDVÆLGELSE AF KERNEKATEGORIER

Den næste del af analyseprocessen bliver at udvælge kernekategorier, som kendetegnes ved at optræde hyppigt i data og relatere sig til de andre større kategorier (Corbin & Strauss, 2015, p. 189). Ud fra ovenstående diagrammer, vores memoer og undersøgelsesspørgsmål, udvælger vi kernekategorierne *Problemløsende* og *Programmering*, som vi vil uddybe i de næste afsnit med udgangspunkt i undersøgelsesspørgsmålene *1. I hvilket omfang udvikler eleverne 21st century skills, når de programmerer i Scratch?* og *2. I hvilket omfang udvikler eleverne computational thinking, når de programmere i Scratch?*

6.2 I HVILKET OMFANG UDVIKLER ELEVERNE 21ST CENTURY SKILLS?

I dette afsnit vil vi supplere med data fra vores deltagerobservationer, da disse forklarer, hvordan eleverne udvikler grundlæggende viden, færdigheder og kompetencer inden for programmeringssproget Scratch.

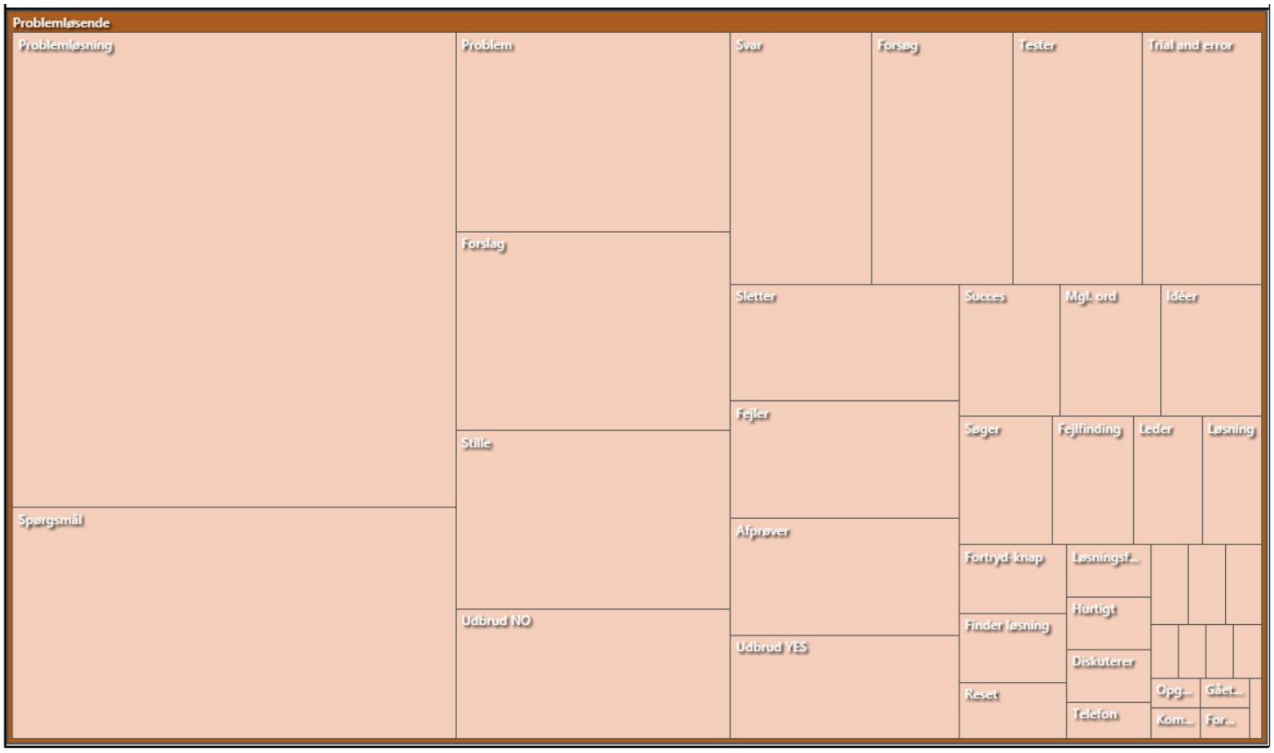
Den første gang medieholdet skal arbejde med programmering, skal de programmere uden strøm, dvs. læreren har lavet forskellige stykker karton som farve-, form- og programmeringsmæssigt svarer til de blokke, de senere skal anvende I Scratch (bilag 16). De står rundt om et stort bord og byder på skift ind med løsninger på de opgaver, læreren stiller dem. Her guider læreren dem i retning af læringsevnen *Abstrakt Begrebsliggørelse*, ved at støtte dem i at udvikle begreber og holdbare teorier med udgangspunkt i de *Konkrete erfaringer,* de gør sig i fællesskab ved at programmere uden strøm. Igennem en fælles dialog har eleverne også mulighed for at gøre sig *Reflekterende observationer*. Til sammen er disse læringsevner et godt afsæt til den næste aktivitet, hvor eleverne skal i gang med et individuelt minikursus i blokprogrammering på Hour of Code (Hour of Code, n.d.), hvor de trin for trin bliver guidet til løsninger af opgaverne (bilag 16). Den næste gang skal de sammen to og to lave nogle trin-for-trin opgaver, som læreren først gennemgår fælles på klassen (bilag 17). Efter disse to gange har eleverne opnået viden og færdigheder inden for blokprogrammering, hvilket Qvortrup betegner som 1. ordens viden (Qvortrup, 2005). Ifølge Illeris (2012) kan overstående aktiviteter betragtes som assimilativ læring, hvormed der tilføjes ny viden til allerede eksisterende skemaer.

Den tredje gang skal de i gang med at programmere deres eget spil i Scratch (bilag 18) og dermed anvende deres viden og færdigheder fra tidligere i en ny og mere åben kontekst og "handle i relation til bestemte kendte, ukendte og uforudsigelige situationer" (Illeris, 2012, p. 35). Intentionen er, at eleverne ændrer eksisterende skemaer gennem akkomodativ læring, når de skal programmere deres eget spil i Scratch, og dermed opnår de, ifølge Illeris (2012), nye kompetencer. Disse kompetencer betragter Qvortrup som 2. ordens viden, der kendetegnes ved, at man har viden om viden (Qvortrup, 2005).

I det næste afsnit vil vi uddybe hvilke kompetencer, eleverne udvikler, når de programmerer i Scratch. Dette vil vi gøre med udgangspunkt i aksekodningen og kernekategorien *Problemløsende*.

PROBLEMLØSENDE

I nedenstående diagram (Figur 33) uddyber vi kategorien *Problemløsende* for både gruppe E og F. Det der kendetegner deres strategier til at løse problemer, ud over problemløsning, er fx at stille spørgsmål, definere et problem, komme med forslag, afprøve forskellige løsninger og slette.



Figur 33: Problemløsende gruppe E og F

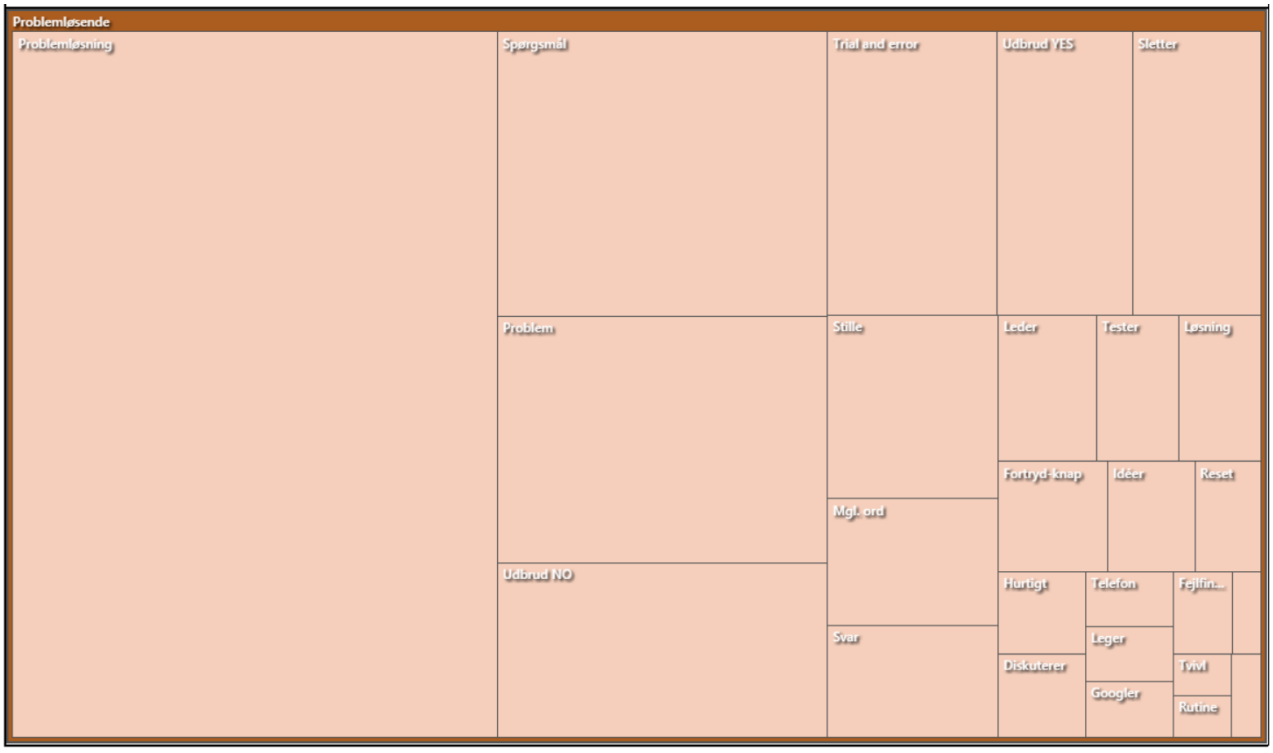
Herunder er et eksempel på problemløsning fra gruppe E, og her kan vi se, at de benytter sig af flg. strategier: stiller spørgsmål, kommer med forslag, finder fejl, finder løsninger og succes.

Ep1: ”Der er kun dertil, vi har sat den. Det er derfor. Du kan måske stoppe spillet igen?”
Ed1: ”Så det her...”
Ep1: ”Der er fordi...Det skal jo herover? Ikke? Og det er jo helt andre koordinater end dem, der er der. Det er derfor, den ikke glider længere.”
Ed1: ”Jamen, det er…jo det, jeg har skrevet herovre.”
Ep1: ”Hva'?”
Ed1: ”Det er jo det, jeg har skrevet.”
Ep1: ”Det er da ikke. Du har da ikke skrevet minus.”
Ed1: ”Nåhh, jeg har ikke skrevet minus i den første” [om X-koordinatpunktet]
Ep1: ”Nej”

Ed1: (Stilhed - taster minus foran tallet i X, trykker på start, Dinosprite siger 'Hvor er jeg henne?', bananasprite klones og klon glider mod Dinosprite.)
Ep1: ”Det dur da!” (bilag 19, l. 37 - 47)

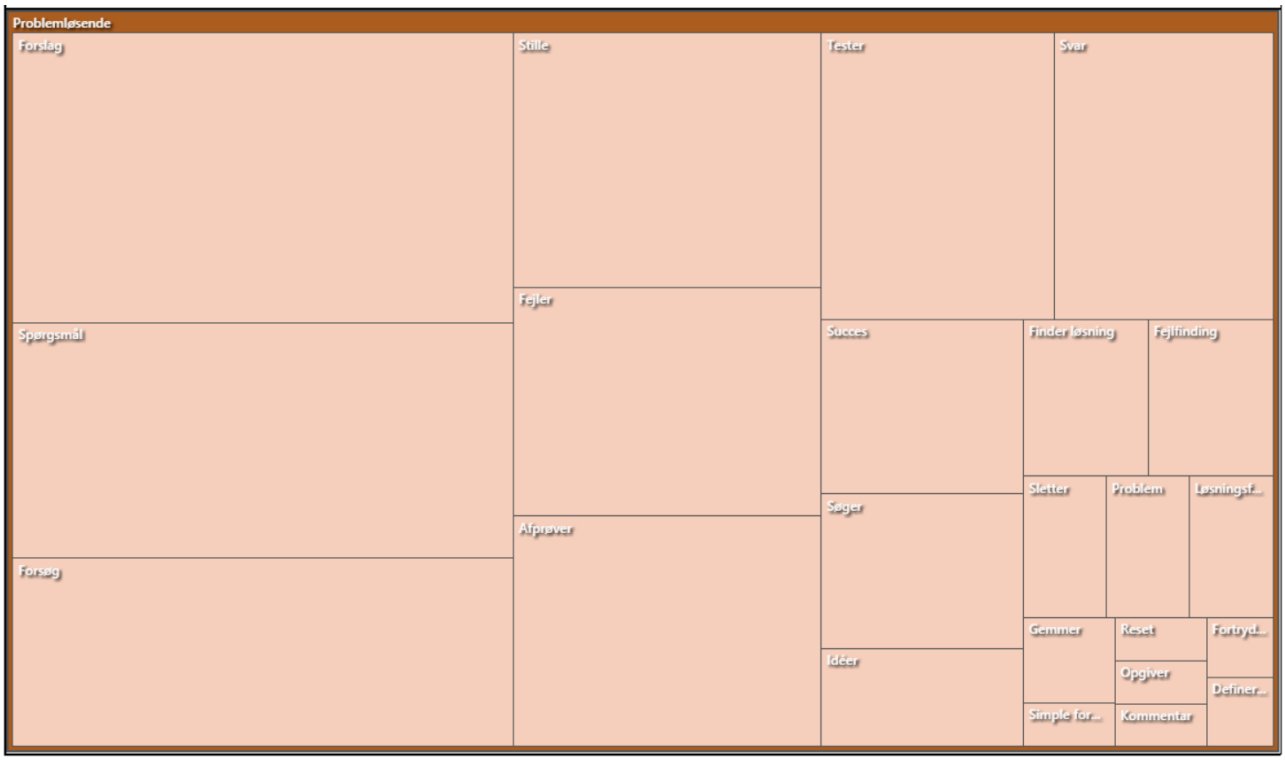
I eksemplet har vi også kodet parprogrammering, driver og navigator, da det er tydeligt, at navigatoren har overblikket, tjekker for fejl og kommer med forslag (Lewis, 2011), som gør det muligt for gruppen at løse problemet i fællesskab.

Når vi sammenligner gruppe E og F, er det tydeligt, at kategorien *Problemløsende* fylder mere i gruppe F. Når vi kigger nærmere på kategorien *Problemløsende* i gruppe F, bemærker vi i nedenstående diagram (Figur 34), at gruppen adskiller sig fra gruppe E (Figur 35) ved at have en del hændelser af *Trial and error*[7], som kendetegnes ved, at de forsøger sig frem med forskellige løsninger. Trial and error kan sammenlignes med læringsevnen *Konkret erfaring* fra Kolbs læringscirkel (Kolb, 2015).



Figur 34: Problemløsning gruppe F

[7] ”Eleverne lærer ved hjælp af forsøg-fejl-metoden” (“Trial and error,” n.d.)



Figur 35: Problemløsende gruppe E

Nedenstående eksempel er kodet som trial and error. Her forsøger gruppe F sig frem med forskellige løsninger:

Fd3: ”Hvordan fjerner jeg denne streg her?” (fortsætter med at trække centreringsmarkeringen rundt på Spriten) ”Hvad er det, der sker?”
Fd2: (makker svarer, men står for langt væk)
Fd3: ”Jeg ved ik’. Prøv at se” (trækker stadig rundt med centreringsmarkøren). ”Okay, vi sætter en ny mand” [ind]
Fd2: ”Jaa”
Fd3: ”Prøv se, så skal jeg tryk’ på ham, men det kan jeg ikke, fordi jeg tegner”
Fd2: ”Åhhhh, prøv bare at klik et eller andet sted”
Fd3: ”Men det kan jeg ikke”
Fd2: (mumler) ”Ja, ik’ også”
Fd3: ”Det kan man heller ikke” (bilag 20, l. 24 - 39)

I ovenstående eksempel dominerer læringsevnen *Konkret erfaring* (Illeris, 2000) og vi ser ikke tegn på *Reflekterende observationer*. Gruppen prøver sig frem med forskellige løsninger og viser ikke, at de trækker på deres viden og færdigheder fra tidligere programmeringsaktiviteter. Den tredje gang de programmerede, ser vi dog en udvikling i form af tegn på, at de også begynder at koble refleksioner på deres *Konkrete erfaringer*.

Fd3: ”Minus 20, det er så tæt på, vi skal lige lidt mere op”
Fd2: ”What? Vent lige. Prøv lige at se herovre. Den er…”

Fd3: ”Katten den er – den skal bare ku’”
Fd2: ”Ok, hvis det her er. Skal vi prøve -30?”
Fd3: ”Jeg ved ikke helt, om det hjælper”
Fd2: ”Prøv”
Fd3: ”Så går den bare længere skridt, tror jeg. For hvis vi ændre den måde, - 1, så går den” [klikker så katten går rundt] ”Så det har ikke noget at gøre med, hvor den går hen. Det er skridtene.” (bilag 21, l. 57 - 63)

Piagets faseteori kan være med til at forklare, hvorfor gruppe F primært gør sig konkrete erfaringer. I løbet af puberteten udvikles elevernes evne til at tænke abstrakt, og hvis nogle af eleverne modnes senere, kan de stadig befinde sig i slutningen af det forrige stadie *den konkret-operationelle,* hvor de stadig er afhængige af de konkrete handlinger med tingene (Jerlang, 1998).

Vi har også kategoriseret begrebet *stille* som problemløsende*,* dvs. når eleverne programmerer uden at tale sammen. I én af vores transskriberinger, hvor vi har kodet gruppe F som stille, har vi skrevet følgende:

Fd2: ”Okay, skal jeg...” (utydeligt hvad han siger)
Fd3: ”Nej, hvordan kan jeg trykke på ham - minimere ham?” (trykker rundt, fortryder handling)
Fd3: (Stilhed 18 sek. - Er i 'Kostumer', klikker 'fortryd', 'gør om' 'fortryd'. Sletter kostume. Klikker på 'Ny Sprite' Vælger sprite kostume, som lige blev slettet. Flytter pilen til 'Operatorer', til baggrund. Klikker på den nye Sprite) (bilag 20, l. 11 - 13)

Driveren i gruppe F ser ud til primært at afprøve forskellige muligheder og i et tempo, hvor det ikke samtidig er muligt at reflektere. Det tolker vi som et tegn på, at eleven primært gør sig *Konkrete erfaringer* (Kolb, 2015). *Stille* kan også være et tegn på, at gruppen på forhånd har reflekteret over løsningen, og derefter er stille, fordi driveren udfører det, de har aftalt. Det vil sige, at eleverne først placerer sig op ad den *Reflekterende observation* for dernæst at bevæge sig over til *Aktiv eksperimenteren* i Kolbs læringscirkel (Corbin & Strauss, 2015; Illeris, 2000).

Ep1: ”Altså - på et tidspunkt, engang da jeg lavede det her. Så lavede vi en eller anden streg her agtig. Som man lige kunne se også. Sådan at når den rørte den, så... Prøv den der ’skab en variabel’ eller sådan noget. Og så prøver vi at arbejde med det. Navn på variabel. Prøv ’på denne Sprite’.”
Ed1: (Stilhed 13 sek. - klikker på 'data', klikker på 'lav ny variabel', klikker på 'kun denne Sprite', skriver navnet Score til variablen, klikker 'OK', Score dukker op i øverste venstre hjørne på baggrunden med værdien 0, peger med musen på Score) (bilag 19, l. 55 - 57)

KOMMUNIKATION OG SAMARBEJDE

I den åbne kodning blev det tydeligt, at eleverne kommunikerede og samarbejdede hele tiden, mens de programmerede i Scratch. Når vi transskriberede, fremgik det tydeligt, at eleverne kommunikerede hele tiden ved på skift at sige noget fx ved at stille spørgsmål, svare og komme med idéer til løsninger.

Inden for kategorien *Samarbejde*, har vi kodet spørgsmål, bytte roller, svar, forklarer m.fl. I nedenstående eksempel, ser vi hvordan gruppe F samarbejder.

> Fd2: ”Okay, slukker lige det her...Prøv se - han griber jo bolden. Han må ikke gribe den - den skal være sådan der eller sådan noget.”
> Fd3: ”Okay, så skal vi lige trykke på bolden.”
> Fd2: ”Okay, hvad er hans koordinat. Tjek lige hvad hans koordinat er. Ja, Amon [Sprite] Hans koordinat er...”
> Fd3: ”What, prøv...når du trykker på...okay. Han er på 165…69. Prøv lige at se bolden, den er 100 - den er 44”
> Fd2: ”Det er når, den går dertil.”
> Fd2: ”Når du trykker på K”
> Fd2: ”Nu er den....”
> Fd3: ”Den går til Amon” (bilag 21, l. 33 - 40)

Den hyppige forekomst af samarbejde har selvfølgelig også sammenhæng med den præmis, at eleverne skulle arbejde sammen to og to ved samme computer under parprogrammeringen.

KREATIVITET

Kreativitet[8] er ikke kodet mere 22 gange på tværs af alle datakilder, hvilket har undret os. Dette kan muligvis forklares med, at Qvortrup (2005) placerer kreativitet som 3. ordens viden, og kreativitet dermed ligger over kompetencer, som er 2. ordens viden. Kreativitet betragtes derfor som en mere dyb form for viden, som kræver at eleverne har flere programmeringskompetencer ift. Scratch, før de kan begynde at opfinde, være idérig og tænke ud af boksen. Dette kan skyldes, at de kun har programmeret i alt fem gange, hvoraf de to først gange gik med minikursus i blokprogrammering og trin-for-trin opgaver. Læreren bemærker, at både gruppe E og F oplever at være begrænset i at mestre alle funktioner i Scratch. Gruppe E begynder at klone deres Sprites, hvilket er en kompleks form for programmering (bilag 22, l. 14). Gruppe F får også hjælp fra læreren på et tidspunkt, hvor de har gang i flere Sprites, end de kan overskue (bilag 20, l. 247).

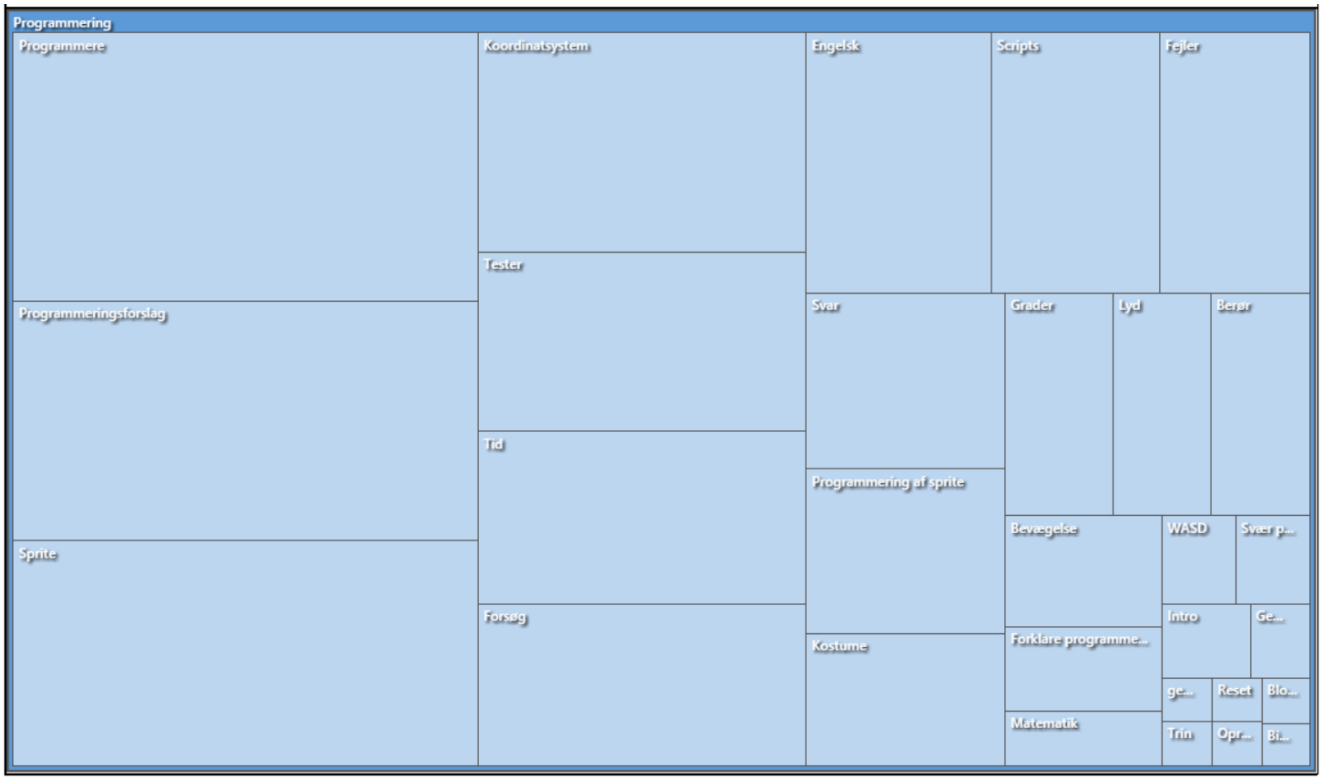
[8] Kreativitet: 1. det at være idérig, original og tænke uden for boksen, 2. evne til at være nytænkende og finde på nye idéer, 3. det at arbejde eller udfolde sig kunstnerisk ved fx at male eller tegne” (“Kreativitet,” n.d.)

OPSUMMERING

I empirien ser vi tydelige tegn på, at eleverne i stort omfang udviklede kompetencer til problemløsning, samarbejde og kommunikation, når de programmerer i Scratch.

6.3 I HVILKET OMFANG UDVIKLER ELEVERNE COMPUTATIONAL THINKING?

Programmering er den andenstørste kategori, hvilket hænger naturligt sammen med, at eleverne bruger det meste af deres tid på at programmere i vores videoobservationer. *Programmering* indeholder begreber som fx programmerer, programmeringsforslag, Sprite, koordinatsystem, tester, tid og forsøg. *Programmering* handler dermed om selve processen, hvori eleverne udvikler deres spil. Kategorierne *Programmering* og *Computational thinking* er tæt forbundne og har en række forbindelser, der viser, at eleverne anvender koncepter og tilgange fra computational thinking, når de programmerer (Barefoot Computing, 2014).

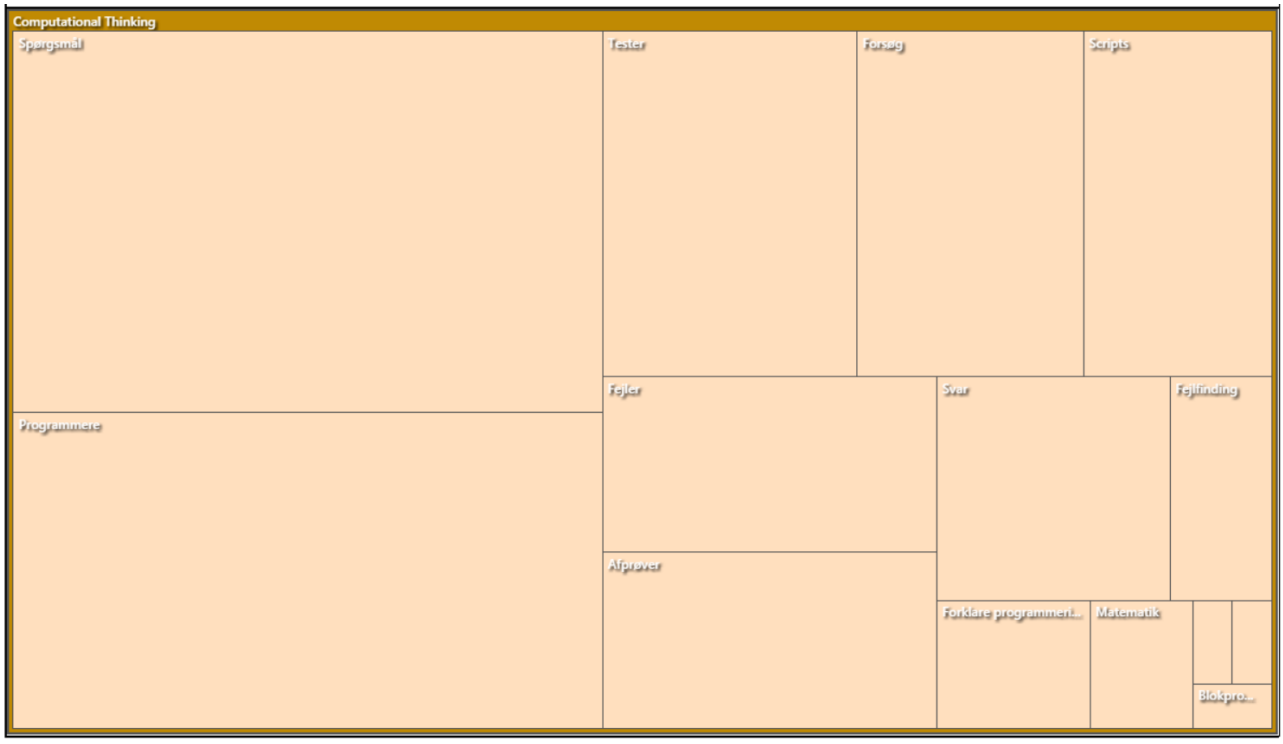


Figur 36: Programmering gruppe E og F

COMPUTATIONAL THINKING

Kategorien *Computational thinking* indeholder begreber som at stille spørgsmål, programmere, tester, forsøg, fejlfinding, afprøve og svar. Disse begreber er også centrale ift. Wings definition af kompetencen computational thinking, som udvikles ved både at formulere problemer ved hjælp af

spørgsmål og svar samt identificere fejl gennem logisk tænkning og afprøvning (Wing, 2010). Det er vigtigt at huske på, at eleverne er i en proces, hvor de udvikler sig til computational thinkers. Så vi kigger på, om eleverne, når de programmerer i Scratch, benytter sig af koncepter eller tilgange indenfor computational thinking (Barefoot Computing, 2014). Vi vil i dette afsnit dykke ned i nogle af et par af tilgangene og koncepterne.



Figur 37: Computational thinking gruppe E og F

TILGANGEN KOLLABORATION

Ifølge Barefoot Computing (2014) er der forskellige tilgange til computational thinking. En tilgang kan være *Kollaboration*, som vi ser ske i stort omfang, når eleverne programmerer i Scratch. Eksemplet herunder viser, hvordan gruppe F kollaborerer om at få en Sprite i deres spil til at flytte sig med piletasterne. Opgaven er simpel, da de tidligere har prøvet flere øvelser der ligner. Alligevel virker de usikre på hvad de skal. Det tyder på at både viden og færdigheder stadig skal trænes, så de kan nå op på 1. ordens viden (Qvortrup, 2005).

Fd3: ”Ændre Y - det er y vi skal have”
Fd2: ”jo, det er opad. Jeg ved ikke om det er minus eller plus.”
Fd3: ”Ja, så skal den jo bare”
Fd2: ”Det er minus”
Fd3: ”okay, så det er bare minus - 1 eller 2?”
Fd3: ”Ja, så skal den jo bare”
Fd2: ”prøv lige med 3”

Fd3: ”Okay, what? Minus 10”
Fd3: ”Jeg tror det er direkte nedad” (bilag 21, l. 64-72)

Som nævnt i kapitel 6.2 i *Kommunikation og Samarbejde* hænger samarbejde og dermed også *Kollaboration*, sammen med at de skal parprogrammere og at de arbejder på den samme computer. Så selvom gruppe F i nedenstående eksempel bevæger sig mellem *Konkret erfaring* og *Aktiv eksperimenteren* i et forsøg på at opnå assimilativ læring og 1. orden viden, ser vi at konceptet Kollaboration, hjælper dem til at komme tættere på, da de fællesskab får de samme *Konkrete erfaringer* og kan hjælpe hinanden i den *Aktive eksperimenteren*.

TILGANGEN DEBUGGING

Debugging er også en tilgang, som eleverne benytter sig af, hvor de bevæger sig mellem *Konkret erfaring* og *Aktiv eksperimenteren* i Kolbs (2015) læringscirkel, mens de identificerer en fejl, som de skal finde en løsning på. Det ser vi, gruppe E gør i nedenstående transskribering, hvor de vil placere en Sprite et bestemt sted ved tryk på mellemrumstasten. Selvom Ep1 og Ed1 gennem denne vekslen mellem *Aktiv eksperimenteren* og *Konkret erfaring* er i stand til at bruge konceptet *Debugging*, er det ikke sikkert, at de er bevidste om, at det er det, de gør. Der er stadig et stykke vej til *Abstrakt begrebsliggørelse* (Kolb, 2015) af *Debugging*, som kræver, at læreren sammen med eleverne sætter ord på konceptet.

Ed1: ”Prøv bare at skrive noget. Vi prøver bare lige.”
Ep1: ”Okay.” (Afprøver i testområdet, ved at trykke på start) ”Så går den der ned.”
Ed1: ”Såååååh prøv.. 50!”
Ep1: (Spriten forsvinder ud af billedet) ”Hvordan får vi den tilbage igen?”
Ed1: ”Y, den skal stå et andet sted bare...”
Ep1: ”Så det er y?” (sætter y til 50) ”Nej, ups..” (trykker på stop og derefter start, Spriten rykker lidt op i billedet).
Ed1: ”Så prøv med 100.”
Ep1: (Taster 100 i y. Trykker på stop og derefter start, Spriten rykker lidt op i billedet.) ”Så røg den længere opad. Den rykker opad nu.”
Ed1: ”Jamen det var også starten.”
Ep1: ”150... Nåh ja, det var jo rigtig. Så sætter jeg den [y] på 150.” (Taster 150 i y. Trykker på start. Spriten flytter højere op, så dele af den forsvinder) ”Nej, okay...”
Ed1: ”Okay...”
Ep1: ”Så det 145....”
Ed1: ”Prøv så med 135...”
Ep1: ”135...” (taster 135 i y. Trykker start. Spriten flytter lidt nedad.) ”Sår'n der. Så når vi gør sådan der.” (Trykker mange gange på pil opad. Spriten flytter sig mod højre) ”Nu

prøver jeg bare lige...” (Trykker på mellemrum. Spriten flytter tilbage til startpunktet.) (bilag 23, l. 11-24)

KONCEPTET ALGORITME

Når Ed1 og Ep1 viser at de mestre tilgangen *Debugging og Kollaboration*, kan man også forestille sig, at de kunne mestre andre koncepter inden for *computational thinking* (Barefoot Computing, 2014). Vi kiggede derfor ind i koden matematik, for se om de benyttede sig af konceptet *Algoritme*. Vi fandt nedenstående stykke, hvor Ep1 og Ed1 vil programmere en Sprite til at bevæge sig fra side til side vha. piletasterne.

Ed1: ”Når man trykker på...Vi starter bare med at lave den til højre.”
Ep1: ”Højre, så skal den gå til...Er det Y? Hvaffor en akse er det?”
Ed1: ”Det er X”
Ep1: ”Er det X-aksen? Nej det er da Y-aksen det der” [om retningen Spriten skal bevæges med]
Ed1: (Trækker kodeblokken 'ændre x med' ind i Scriptet) ”Det får vi at se.” (Trykke på start og på højre pil. Spriten bevæger sig mod højre)
Ep1: ”Nååååh. Okay.” (bilag 24, l. 21-26)

Her bevæger Ep1 sig mellem *Reflekterende observation* og *Konkret erfaring* i Kolbs lærings cirkel, når hun udtrykker sin usikkerhed på, hvilken akse i koordinatsystemet, der skal flyttes langs. Hverken Ep1 eller Ed1 er i tvivl om hvilken type kodeblok, der skal bruges, så de er i stand til at oprette algoritmen, da de begge ved, hvilke trin der skal til for at oprette algoritmen. Der er heller ikke i tvivl om, at når Spriten skal flyttes i en bestemt retning, skal der ændres på koordinatet for X eller Y, så de er helt klare på, hvilke regler der skal benyttes i algoritmen. Der er derfor gode tegn på, at gruppe E benytter sig af konceptet *Algoritme*. Ed1 synes dog at være helt sikker på, hvilken akse det handler om. Det kunne tyde på, at han bevæger sig mellem *Abstrakt begrebsliggørelse*, og *Aktiv eksperimenteren,* og at han på dette område har opnået akkomodativ læring, da han ved, hvilken retning Spriten skal bevæges i, selvom han ikke kan se et konkret koordinatsystem i deres Scratch-projekt. Om Ep1 bevæger sig mod akkomodativ læring, da hun i sidste linje, gør en konkret erfaring, ser vi ingen tegn på her. Dagen efter ser det dog ud til at hun har godt styr på koordinatsystemet (se eksempel i kapitel problemløsende), så der kan være tegn på, at hun har fået ændret i sit skema (Illeris, 2015) og opnået akkomodativ læring.

TILGANGEN ABSTRAKTION

Abstraktion (Barefoot Computing, 2014) i computational thinking handler om at kunne se bort fra unødvendige detaljer. At kunne se bort fra detaljer betyder ofte, at man kan forklare og løse en opgave ved brug af overordnede begreber. I denne sammenhæng er det interessant at drage læringsevnen *Abstrakt begrebsliggørelse* i Kolbs læringscirkel, når vi ser på elevernes brug af

begreber til at integrere deres refleksioner og observationer i holdbare teorier med (Illeris, 2000). Det kan diskuteres i hvilken grad de befinder sig ift. denne læringsevne, men brugen af forskellige programmeringsbegreber kan være tegn på, at de nærmer sig.

I empirien er det tydeligt, at gruppe E og F arbejder forskelligt, når det gælder brugen af abstrakte begreber. Gruppe E benytter sig ofte af udtryk og begreber fra Scratch, som vi herfra vil benævne som Scratch-begreber og fremhæve med fed skrift. især Ep1, som er navigator, bruger begreberne, og Ed2, som er driver, forstår begreberne, da man kan se, at der hurtigt springes til rette kategori eller brik.

Ed1: ”Hmm.. Jeg ved ikke rigtig lige, hvad jeg skal lægge det ind under.”
Ep1: ”Hva'?”
Ed1: ”Hvad skal vi lægge det ind under?”
Ep1: ”Øhm. Den der **berører** og så skal vi have...Så **hvis** den **berører**, **så** skal den vel give **minus** point. Altså **hvis** den **berører..**”
Ed1: ”Mmmm”
Ep1: ”**Hvis** den **berører** en banan, **så** skal den **minus** et point.”
Ed1: (Stilhed - trækker de sammenstatte kodeblokkene 'hvis 'berøre musepil' så' ned, klikker på Kostume, klikker på Scripts, klikker på hændelser, vælger 'Når du klikker på start')
Ep1: ”Så skal den vist....Ja det må vist være sådan der...**Når der klikkes på**...”
Ed1: (Stilhed - Klikker på start mange gange, Dino-Sprite siger 'Hvor er jeg henne' mange gange)
Ep1: ”Jamen det er jo også...**Hvis** den **berører museklik**, det gør den jo. Det skal jo være **når jeg berører bananas**. Og **Apple**, det vil sige vi skal have to af dem...”
(bilag 19, l. 67 - 76)

Ved gruppe F har vi ofte kodet *ufærdige sætninger*, *afbryder*, *afslutter hinandens sætninger* og *mangler ord* som en del af deres strategier til problemløsning. Her kommer et eksempel på, hvordan de næsten ikke bruger Scratch-begreber.

Fd2: ”Ja, men han skal først lige… Når… (afbrydes)”
Fd3: ”Når den rammer den [baseball]”
Fd2: ”Når **Sprite 1** eller sådan noget rammer baseball…kom derhen”
Fd3: ”Så kommer den derhen”
Fd2: ”Kom derhen…gå 10 skridt eller…10 skridt til højre eller sådan noget. Du ved godt, hvad jeg mener”
Fd3: ”Når den rammer den der, så…”
Fd2: ”Så den ligeså stille triller hen ”
(bilag 20, l. 128 – 135)

Denne forskel kan igen forklares med, at gruppe F er sene til at indtræde i Piagets 4. stadie, og derfor er udfordret på at udvikle og anvende abstrakte Scratch-begreber.

OPSUMMERING

Der er betydelig forskel på, hvor de to grupper er i forhold til computational thinking, dog viser begge grupper, at der arbejdes med flere koncepter og tilgange fra kompetencen.

6.4 I HVILKET OMFANG UDVIKLER ELEVERNE DIGITAL DANNELSE?

I vores interviews med de fire gruppe, spurgte vi ind til, hvad de synes om at arbejde med parprogrammering og task boardet, samt hvad de kunne bruge det, de havde lært i forløbet andre steder. Gennem deres erfaringer med at udvikle eget spil i Scratch, havde flere elever fået en indsigt i, hvad det vil sige at programmere de større spil, som de selv spiller derhjemme.

> Anja: ”Hvordan er det her med, at man både kan spille nogle spil, og nu har I fået et indblik i hvordan, man programmerer et spil?”
> Fd3: ”Det er fedt, så kan vi se, hvor svært det egentlig er.”
> Anja: ”Ja”
> Fd2: ”I forhold til de helt store spil…hvor meget [tid] det tager” (bilag 25, l. 26 - 29)

Gruppe H har også fået indsigt i, hvad det vil sige, at programmere de store spil som fx GTA:

> Hp3: ”Altså jeg har lært…sådan hvad der egentlig ligger bag at lave spil. Selvom det bare er sådan et lille spil…med en bold, som skal forbi nogle forhindringer. Det tager sådan virkelig lang tid. Så prøv at tænke på dem, som sådan skal programmere de der store spil fx GTA eller sådan noget.” (bilag 26, l. 62)

Gruppe E fortæller, at det også var interessant at lære om computerspillets historie og computerens udvikling:

> Ep1: ”Øhm, jeg synes det har været meget interessant at høre om om computerspillets historie. Det med, hvor det først blev programmeret på de der kæmpe maskiner og sådan noget…at det nu har udviklet sig ved, at nu har man sådan…en virtuel verden, hvor man bare får sådan et par briller på. Hvor [man] så styrer det hele. Så hvordan det har udviklet sig, synes jeg er meget fedt.” (bilag 27, l. 112)

Flere af eleverne giver i ovenstående interviews udtryk for, at de har fået større indsigt i, hvad der ligger bag de store og avancerede spil, som de spiller i deres fritid, efter at de selv har programmeret et simpelt spil i Scratch. Dette viser, at de har fået en større forståelse for teknologien og ”dens rolle i vores fælles liv” (Bundsgaard, 2017, p. 12). Ved at udvikle deres eget spil, får de også kompetencerne til selv at udvikle og styre teknologierne (Bundsgaard, 2017), hvilket er centralt for elevernes digitale dannelse. Kompetencer til problemløsning, samarbejde og kommunikation, som eleverne udvikler, når de programmerer i Scratch, kan ifølge Schnack (2011) også have et dannelsesperspektiv for den enkelte elev. Digital dannelse handler nemlig også om,

at løse problemer med it (Regeringen/KL/Danske-Regioner, 2016, p. 55), hvilket vi i høj grad ser tegn på i kernekategorien *Problemløsende*.

7 KONKLUSION

Før vi svarer på vores problemformulering: *Hvilke kompetencer udvikler eleverne, når de programmerer i Scratch?,* vil vi først besvare de tre undersøgelsesspørgsmål.

I HVILKET OMFANG UDVIKLER ELEVERNE 21ST CENTURY SKILLS?

Eleverne udvikler viden og færdigheder inden for programmering gennem online programmeringskurser i blokprogrammering og ved at løse trin-for-trin opgaver, som læreren har forberedt. Dette betragter vi som 1. ordens viden qua Qvortrups forskellige kategorier af viden (Illeris, 2012). Eleverne anvender deres nye viden og færdigheder, når de skal udvikle deres eget spil i Scratch.

I analysen ser vi tydelige tegn på, at eleverne især udvikler kompetencer til problemløsning, kommunikation og samarbejde, når de programmerer i Scratch. Problemløsning er helt central, når eleverne skal designe og udvikle deres spil i Scratch. Gennem iterative loops, hvor *erfaringsbaserede* lærerprocesser er centrale, definerer eleverne et problem, finder en løsning, tester om det virker, tilpasser og afprøver igen. Eleverne arbejder sammen to og to, og her ser vi også tydelige tegn på, at de udvikler deres kommunikative kompetencer ved på skift at stille spørgsmål, svare, forklare sine idéer mm. Det er også meget tydeligt, at eleverne udførte de forskellige opgaver i fællesskab og dermed samarbejdede.

Disse tre kompetencer er helt centrale i Kereluik m.fl.'s (2013) model, hvor problemløsning, kommunikation og samarbejde betegnes som vigtige kompetencer i det 21. århundrede.

I HVILKET OMFANG UDVIKLER ELEVERNE COMPUTATIONAL THINKING?

Computational thinking, som er en kompleks kompetence og har tætte bånd til 21st century skills, ser vi flere tegn på, at eleverne er på vej i mod. Vi har kun kigget på to koncepter og to tilgange, og selvom eleverne er på forskelligt niveau og har opnået forskellige læringsstadier jf. Illeris (2015), ser vi tegn på, at de alle får oparbejdet færdigheder inden for computational thinking, og dermed er på vej til at kunne benytte computational thinking i andre sammenhænge.

Et af de centrale begreber inden for computational thinking er *Abstraktion*. Ved nogle elever ser vi tegn hos på, at de kan tænke og handle abstrakt, dog er det væsentligt at være opmærksom på, at eleverne kan befinde sig på forskellige stadier i deres kognitive udvikling.

Analysen viser, at det er forskelligt, hvor de befinder sig i forhold til Kolbs læringscirkel, alt efter hvor godt de mestrer de enkelte tilgange eller koncepter, samt hvor stor viden hver elev har. Dog ser de alle ud til at bevæge sig mellem flere læringsevner, hvilket antyder, at de arbejder med at udvikle sig hele tiden gennem forløbet. Dermed når alle elever større færdigheder inden for de forskellige koncepter og tilgange i computational thinking, og vi ser tegn på, at alle er på vej til at blive computational thinkers.

I HVILKET OMFANG UDVIKLER ELEVERNE DIGITAL DANNELSE?

I vores delkonklusion efter afsnittet omkring dannelse, konkluderer vi ud fra en didaktisk analyse, at programmering har dannelsespotentiale – både her og nu men også ift. fremtiden.

I analysen ser vi, at eleverne har fået en større indsigt i, hvad der ligger bag de store og avancerede spil, som de spiller i deres fritid, efter at de selv har programmeret et simpelt spil i Scratch. Dette er tegn på, at de har fået en større forståelse for teknologien og "dens rolle i vores fælles liv" (Bundsgaard, 2017, p. 12). Ved at udvikle deres eget spil, får de også kompetencerne til selv at udvikle og styre teknologierne (Bundsgaard, 2017), hvilket er centralt for elevernes digitale dannelse.

HVILKE KOMPETENCER UDVIKLER ELEVERNE, NÅR DE PROGRAMMERER I SCRATCH?

Vi kan konkludere, at eleverne især udvikler kompetencer til problemløsning, kommunikation og samarbejde, som er centrale kompetencer i 21st century skills. Eleverne viser også flere tegn på at kunne benytte sig af koncepter og tilgange fra kompetencen computational thinking, når de programmerer i Scratch. Eleverne udvikler desuden digital dannelse, idet de har fået en større indsigt i, hvad der ligger bag de store og avancerede spil, som de spiller i deres fritid, ved selv at programmere et simpelt spil i Scratch.

8 PERSPEKTIVERING

Konklusionen fra dette masterprojekt kan forhåbentlig bruges som et indspark i debatten om programmering i folkeskolen og være med til at svare på spørgsmålet *Hvorfor skal eleverne programmere i folkeskolen?*

Vi har også gjort en række andre fund, som vi vil udfolde i denne perspektivering, herunder *Hvor meget skal programmering fylde?, Lærerens kompetencer* og *Piger og programmering*.

HVOR MEGET SKAL PROGRAMMERING FYLDE?

I empirien, hvor eleverne programmerer i et forløb á fem gange 1,5 time, ser vi ikke mange tegn på, at eleverne når at udvikle kreative kompetencer. Hvis sigtet er, at folkeskolen skal uddanne elever med stærke og kreative it-kompetencer, skal de deltage i undervisningsforløb, hvor der er mere tid til at udvikle viden, færdigheder og kompetencer. Dermed er korte forløb, som er spredt tilfældigt ud over elevernes samlede skoleforløb, ikke nok. Et valgfag for interesserede elever i udskolingen er heller ikke nok, hvis vi vil have ALLE elever med og dermed indfri Iversens ambition om, at vi i fællesskab uddanner danskerne til at blive de mest "IT-kompetente og IT-kritiske borgere i verden" (Iversen, 2017a) og indfrier undervisningsministerens mål om, at "udvikle Danmarks styrkeposition i forhold til anvendelse af it i undervisningen" (UVM, 2017c).

LÆRERENS KOMPETENCER

Der er ingen tvivl om, at 6. klasse i vores empiri har en dygtig og meget kompetent lærer, der, udover at have gode relationer til eleverne og stærke kompetencer inden for klasseledelse, også har generelle stærke it-kompetencer og derudover færdigheder og kompetencer inden for programmering, der må betragtes som værende over gennemsnittet. Disse kompetencer er bl.a. opnået og styrket gennem et opkvalificeringsforløb i projekt #ProgrammeringNaturligvis, hvor de deltagende lærere arbejder med programmering gennem fire moduler á fire hele dage. Derudover deltager lærerne i ni dage med opfølgning, videndeling og gruppevejledning. Denne lærers opkvalificeringsforløb når samlet set op på næsten 25 hele dage, hvor der også har været gode muligheder for hands-on, refleksioner, afprøvning af forløb i undervisning, videndeling og evaluering i et fællesskab med andre lærere.

Hvis eleverne i folkeskolen skal møde lærere, som har kompetencer på samme niveau, som læreren fra vores empiri, vil det formodentlig kræve, at lærerne gennemgår et opkvalificeringsforløb i lignende omfang. Elementerne fra opkvalificeringsforløbet bør også indtænkes på læreruddannelsen, således at de nyuddannede lærere allerede besidder kompetencerne, når de kommer ud i folkeskolen.

PIGER OG PROGRAMMERING

I vores empiri har vi ikke haft et særligt fokus på pigerne, men alligevel ser vi, hvordan pigerne i 6. klasse, på lige fod med drengene, er meget motiverede og udvikler kompetencer til at programmere. En pige fortæller i interviewet:

> Altså jeg kunne godt finde på at kode i fritiden. Jeg har tænkt på det. Men det er bare sådan, jeg har ikke en computer eller noget. Jeg har ikke lyst til at gøre det på iPaden, så jeg venter bare lige til min fødselsdag. (bilag 26, l. 74)

Alligevel falder antallet af kvinder, der søger ind på it-uddannelserne, og regeringen er opmærksomme på dette frafald i deres regeringsgrundlag for 2016 og vil derfor igangsætte en

række aktiviteter, ”som skal fremme piger og kvinders interesse for it og programmering – blandt andet gennem rollemodeller, praktik og aktiviteter” (Regeringen, 2016, p. 82).

Vi fortsætter derfor vores samarbejde og bygger videre på vores erfaringer fra #ProgrammeringNaturligvis og dette masterprojekt i et kommende pilotprojekt for skoler i Aalborg Kommune, hvor piger i udskolingen tilbydes valgholdet *Kreativ kodning for piger*.

LITTERATURLISTE

Allsopp, B. B., & Ejsing-Duun, S. (2016). Programming Concepts in Playful Programming Products. In *10th European Conference on Games Based Learning: ECGBL 2016* (p. 1).

Antal optagne fordelt på køn. (2016). Retrieved May 26, 2017, from http://studier.ku.dk/bachelor/ansoegning-og-optagelse/optagelsesstatistik/2016/anta-optagne-fordelt-paa-koen/

Apple. (n.d.). Swift Playgrounds. Retrieved April 9, 2017, from http://www.apple.com/dk/swift/playgrounds/

Bach, K. (2017). Vækstråd anbefaler øget digitalisering af undervisning, selvom effekten er tvivlsom. Retrieved May 10, 2017, from https://www.information.dk/indland/2017/05/vaekstraad-anbefaler-oeget-digitalisering-undervisning-selvom-effekten-tvivlsom

Barefoot Computing. (n.d.). About Barefoot - Barefoot Computing Barefoot Computing. Retrieved May 23, 2017, from https://barefootcas.org.uk/about-barefoot/

Barefoot Computing. (2014). What is computational thinking?, 1–6. https://doi.org/June 2009

BBC. (2016). BBC micro:bit launches to a generation of UK students - Media Centre. Retrieved February 19, 2017, from http://www.bbc.co.uk/mediacentre/latestnews/2016/bbc-micro-bit-schools-launch

Beck, E. E. (2011). *Computers in Education: What for? Nordic Journal of Digital Literacy* (Vol. 6). Universitetsforlaget. Retrieved from https://www.idunn.no/dk/2011/special_issue/art03

Behrendt, M. (2017). It-camp for piger: "Det er et demokratisk problem, hvis ét køn designer vores samfund." Retrieved May 25, 2017, from http://www.dr.dk/nyheder/viden/tech/it-camp-piger-det-er-et-demokratisk-problem-hvis-et-koen-designer-vores-samfund

Bensav. (2016). ELI5: Difference between Coding, Programming and Computer science? [Forum Post]. Retrieved April 27, 2017, from https://www.reddit.com/r/explainlikeimfive/comments/41zw2l/eli5_difference_between_coding_programming_and/

Berthelsen, U. D. (2017). 21st century skills om det 21. århundredes kompetencer - fra arbejdsmarkdspolitik til allemandseje. Retrieved from http://www.videnomlaesning.dk/media/2080/21st-century-skills-ulf-dalvad_berthelsen.pdf

Brinkmann, S., & Tanggaard, L. (2015). *Kvalitative metoder: en grundbog* (2. udgave). Kbh.: Hans Reitzel.

Bryman, A. (2008). Qualitative data analysis. In *Social research methods* (pp. 538–563). Oxford University Press.

Bundsgaard, J. (2017). *Digital dannelse*. Aarhus; [Kbh.]; [Risskov: Aarhus Universitetsforlag ; I

samarbejde med Danmarks Lærerforening ; Frie Skolers Lærerforening.

Burke, Q., & Kafai, Y. B. (2011). Digital Storytelling with Scratch in Middle School Classrooms. *SIGCSE'12*, 1–6. https://doi.org/10.1145/2157136.2157264

Caspersen, M. E., & Kölling, M. (2009). STREAM: A First Programming Process. *ACM Transactions on Computing Education*, *9*(1), 1–29. https://doi.org/10.1145/1513593.1513597

Center for Computational Thinking. (2017). About CCT. Retrieved March 30, 2017, from http://cct.au.dk/about-cct/

CFU. (n.d.). 21Skills. Retrieved March 24, 2017, from http://info.21skills.dk/

Christensen, O. (2017). "Digital dannelse" er det modsatte af dannelse [Blog Post]. Retrieved April 22, 2017, from http://laererbloggen.dk/digital-dannelse-modsatte-dannelse/

Claro, M., & Ananiadou, K. (2009). *21st Century Skills and Competences for New Millennium Learners in OECD Countries*. Retrieved from http://www.oecd-ilibrary.org/education/21st-century-skills-and-competences-for-new-millennium-learners-in-oecd-countries_218525261154

Clausen, J. (2009). Turingmaskine. In *Den Store Danske, Gyldendal*. Retrieved from http://denstoredanske.dk/index.php?sideId=175230

Coding Pirates. (n.d.). Coding Pirates - Programmering for børn og unge. Retrieved March 11, 2017, from https://codingpirates.dk/

Collin, F. (2004). *Konstruktivisme*. Frederiksberg: Samfundslitteratur.

Collin, F. (2011). Socialkonstruktivisme i humaniora. In *Humanistisk videnskabsteori* (pp. 247–276). Søborg: DR Multimedie.

Corbin, J., & Strauss, A. (2012). Elaborating the Analysis. In *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (pp. 195–229). https://doi.org/10.4135/9781452230153.n9

Corbin, J., & Strauss, A. (2015). *Basics of qualitative research : techniques and procedures for developing grounded theory*. Retrieved from https://us.sagepub.com/en-us/nam/basics-of-qualitative-research/book235578

Danmarks Vækstråd. (2016). Rapport om kvalificeret arbejdskraft. Retrieved from http://danmarksvaekstraad.dk/file/634221/Rapport_om_kvalificeret_arbejdskraft.pdf

Danyl. (2011). What Are The Differences Between Programming And Coding [Forum Post]. Retrieved April 27, 2017, from http://www.nairaland.com/755214/what-differences-between-programming-coding

Declaration, B. (1999). The european higher education area. *Joint Declaration of the European Ministers of Education*, *19*. Retrieved from http://www.magna-charta.org/resources/files/BOLOGNA_DECLARATION.pdf

Dede, C. (2009). Comparing Frameworks for "21 st Century Skills." Retrieved from http://www.watertown.k12.ma.us/dept/ed_tech/research/pdf/ChrisDede.pdf

Delhij, A., van Solingen, R., & Wijnands, W. (2015). eduScrum Guiden - "Spillets regler." Retrieved from http://eduscrum.nl/en/file/CKFiles/The_eduScrum_Guide_DK_1.2.pdf

Department for Education. (2013). National curriculum in England: computing programmes of study - GOV.UK. Retrieved May 23, 2017, from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

Destron. (2006). difference between programming and coding? [Forum Post]. Retrieved April 27, 2017, from https://www.gamedev.net/topic/392174-difference-between-programming-and-coding/

Digital Vækstpanel. (2017). Danmark som digital frontløber.

Dorling, M. (2015). CAS computational thinking - A Guide for teachers Computing At School. Retrieved February 19, 2017, from https://community.computingatschool.org.uk/resources/2324

EDB. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?word=edb&dict=auto

Elf, N., & Koed, K. (2017). To kommentarer til Ulf Dalvad Berthelsen: 21st Century Skills. Retrieved from http://www.videnomlaesning.dk/media/2081/21st-century-skills-kommentarer.pdf

EMU. (n.d.-a). Fase 2 - Teknologi og ressourcer - Undersøgelse - 5. - 6. klasse - Natur/teknologi. Retrieved May 20, 2017, from http://www.emu.dk/omraade/gsk-lærer/ffm/naturteknologi/5-6-klasse/undersøgelse/teknologi-og-ressourcer/fase-2

EMU. (n.d.-b). Fysik/kemi. Retrieved May 20, 2017, from http://www.emu.dk/omraade/gsk-lærer/ffm/fysikkemi

EMU. (n.d.-c). Fysik/kemi - Fælles Mål, læseplan og vejledning. Retrieved May 20, 2017, from http://www.emu.dk/modul/fysikkemi-fælles-mål-læseplan-og-vejledning

EMU. (n.d.-d). Lærere og pædagogisk personale - Grundskole. Retrieved March 12, 2017, from http://www.emu.dk/omraade/gsk-lærer/

EMU. (n.d.-e). Undersøgelse - 7. - 9. klasse - Fysik/kemi. Retrieved May 20, 2017, from http://www.emu.dk/omraade/gsk-lærer/ffm/fysikkemi/7-9-klasse/undersøgelse

EMU. (n.d.-f). Vejledning for faget matematik. Retrieved May 20, 2017, from http://www.emu.dk/modul/vejledning-faget-matematik

EMU. (2014a). Innovation og entreprenørskab. Retrieved April 23, 2017, from http://www.emu.dk/modul/innovation-og-entreprenørskab

EMU. (2014b). It og medier - vejledning. Retrieved April 23, 2017, from

http://www.emu.dk/modul/vejledning-det-tværgående-emne-it-og-medier

Exner, M. (2012). I Estland lærer de at programmere fra 1. klasse. Retrieved March 11, 2017, from https://www.folkeskolen.dk/519757/i-estland-laerer-de-at-programmere-fra-1-klasse

Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. Retrieved from http://andrey.hristov.com/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf

Future People. (n.d.). Retrieved May 26, 2017, from http://www.futurepeople.dk/

Færdighed. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?word=færdighed&dict=auto#ddno

Google for Education. (n.d.). Google for Education: Computational Thinking. Retrieved May 23, 2017, from https://edu.google.com/resources/programs/exploring-computational-thinking/#!ct-overview

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Guzdial, M. (2008). Education Paving the way for computational thinking. *Communications of the ACM*, *51*(8), 25. https://doi.org/10.1145/1378704.1378713

Gynther, K. (2010). *Didaktik 2.0: læremiddelkultur mellem tradition og innovation*. Kbh.: Akademisk.

Hagen, W. (2006). The Style of Sources: Remarks on the Theory and History of Programming Languages. *New Media, Old Media. A History and Theory Reader*, 157–174. Retrieved from http://www.whagen.de/publications/2005/DigitalMediaRoutledge/RT2241_C0091.pdf

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: a literature review. *Computer Science Education*, *21*(2), 135–173. https://doi.org/10.1080/08993408.2011.579808

Hansen, H. B. (n.d.). Program (Syntaks og semantik). In *Den Store Danske, Gyldendal*. Retrieved from http://denstoredanske.dk/It,_teknik_og_naturvidenskab/Informatik/Software,_programmering,_internet_og_webkommunikation/program/program_(Syntaks_og_semantik)

Hazewinkel, M. (n.d.). Turing machine. Retrieved April 28, 2017, from https://www.encyclopediaofmath.org/index.php/Turing_machine

Hitsa. (n.d.). ProgeTiger Programme. Retrieved April 9, 2017, from http://www.hitsa.ee/it-education/educational-programmes/progetiger

Hour of Code. (n.d.). Hourofcode.org. Retrieved April 9, 2017, from https://hourofcode.com/dk

Hvid, M. (2001). Datalære - faget, som forsvandt. Retrieved March 11, 2017, from https://www.folkeskolen.dk/10187/datalaere---faget-som-forsvandt

Illeris, K. (2000). Den erfaringsbaserede lærerproces. In *Tekster om Læring* (pp. 47–66). Roskilde Universitetsforlag.

Illeris, K. (2012). *Kompetence: hvad, hvorfor, hvordan?* (2. udgave). Frederiksberg C.: Samfundslitteratur.

Illeris, K. (2015). Forskellige læringstyper. In *Læring* (pp. 51–72). Frederiksberg: Samfundslitteratur.

Informationsteknologi. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?word=informationsteknologi&dict=auto

IT-Branchen. (n.d.). Hvad er Coding Class? Retrieved March 11, 2017, from https://itb.dk/articles/fremtidens-kompetencer/hvad-er-coding-class

IT-Branchen. (2017a). Hvordan skal kreativ it på skoleskemaet? [News Post]. Retrieved March 14, 2017, from https://itb.dk/news/fremtidens-kompetencer/hvordan-skal-kreativ-it-på-skoleskemaet

IT-Branchen. (2017b). Høring: Nyt it-fag i folkeskolen [Event Post]. Retrieved March 12, 2017, from https://itb.dk/event/høring-nyt-it-fag-i-folkeskolen

ITL Research. (2014). 21CLD Learning Activity Rubrics. *21st Century Learning Design*, (December), 1–44.

ITU. (2014). ITU releases annual global ICT data and ICT Development Index country rankings. Retrieved March 12, 2017, from http://www.itu.int/net/pressoffice/press_releases/2014/68.aspx#.WMV_YRLhCAy

Iversen, O. S. (2017a). IT-professor: Sådan giver vi folkeskolen en digital opstrammer. Retrieved May 9, 2017, from http://www.altinget.dk/uddannelse/artikel/it-professor-saadan-giver-vi-folkeskolen-en-digital-opstrammer

Iversen, O. S. (2017b). Professor: Gør hurtigst muligt IT-valgfaget i folkeskolen obligatorisk. Retrieved May 25, 2017, from http://www.altinget.dk/uddannelse/artikel/professor-goer-hurtigst-muligt-it-valgfaget-i-folkeskolen-obligatorisk

Jacobsen, B., Schnack, K., Wahlgren, B., & Madsen, M. B. (2009). *Videnskabsteori*. Kbh.: Gyldendal.

Jank, W., & Meyer, H. (2006). *Didaktiske modeller: grundbog i didaktik* (1. udg., 1). København: Gyldendal.

Jerlang, E. (1998). Jean Piagets teori om erkendelsen. In *Udviklingspsykologiske teorier: en introduktion* (pp. 233–277). Kbh.: Munksgaard.

Johnsen, E., Bollerslev, P., Brøndum, U., Engberg, O., Hansen, B., Juhl, H., … Nilsson, L. (1972). *Betænkning om EDB-Undervisning i Det Offentlige Uddannelsessystem*. København: S. L. Møllers Bogtrykkeri.

Kereluik, K., Mishra, P., Fahnoe, C., & Terry, L. (2013). What Knowledge Is of Most Worth. *Journal of Digital Learning in Teacher Education*, *29*(4), 127–140.

https://doi.org/10.1080/21532974.2013.10784716

Klafki, W. (2016). Dannelsesteori og didaktik: nye studier (pp. 27–99). Århus: Klim.

Kode, 1. (n.d.). In *Den Danske Ordbog*. Retrieved from http://ordnet.dk/ddo/ordbog?query=kode

Kode, 3. (n.d.). In *Den Danske Ordbog*. Retrieved from http://ordnet.dk/ddo/ordbog?select=kode,3&query=kode

Kodeks. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?dict=ddob&word=kodeks

Kolb, D. A. (2015). *Experiential learning : experience as the source of learning and development* (Second, Vol. 82). Pearson Education Inc.

Konak, A., Clark, T. K., & Nasereddin, M. (2014). Using Kolb's Experiential Learning Cycle to improve student learning in virtual computer laboratories. *Computers and Education*, *72*, 11–22. https://doi.org/10.1016/j.compedu.2013.10.013

Kreativitet. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?word=kreativitet&dict=auto

Kudahl, S. (2015). It-investeringer i skolen kan betale sig. Retrieved May 10, 2017, from http://www.kl.dk/Folkeskolen1/It-investeringer-i-skolen-kan-betale-sig-id186899/

Kvale, S., & Brinkmann, S. (2014). *Interview - Det kvalitative forskningsinterview som håndværk* (Third). Kbh.: Hans Reitzels Forlag.

Lewis, C. M. (2011). Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education*, *21*(2), 105–134. https://doi.org/10.1080/08993408.2011.579805

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

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (Moti). (2013). Learning computer science concepts with Scratch. *Computer Science Education*, *23*(3), 239–264. https://doi.org/10.1080/08993408.2013.832022

Ministeriet for Børn Undervisning og Ligestilling. (2016). Bekendtgørelse af lov om folkeskolen. Retrieved May 13, 2017, from https://www.retsinformation.dk/Forms/r0710.aspx?id=182008

MIT Media Lab. (n.d.). Scratch | Home | imagine, program, share. Retrieved April 9, 2017, from https://scratch.mit.edu/about

Moreno-Leon, J., & Robles, G. (2016). Code to learn with Scratch? A systematic literature review. In *2016 IEEE Global Engineering Education Conference (EDUCON)* (Vol. 10–13–Apri, pp. 150–156). IEEE. https://doi.org/10.1109/EDUCON.2016.7474546

Møller, K. M., Tosca, S., Husfeldt, T., & Thomassen, C. (2014). Evaluering af informationsteknologi c/b som forsøgsfag på STC, HF, HTX OG HHX, (November), 0–72. Retrieved from http://uvm.dk/-/media/filer/uvm/udd/gym/pdf15/maj/150512-evaluering-af-informationsteknologi-c-b-som-forsoegsfag-paa-stx--hf--htx-og-hhx.pdf

Nash, J. (2017). Turn coders into computational thinkers. Retrieved May 23, 2017, from https://www.iste.org/explore/articleDetail?articleid=936&category=Innovator-solutions&article=Turn+coders+into+computational+thinkers

Nofre, D. (n.d.). A. J. Perlis - A.M. Turing Award Winner. Retrieved April 28, 2017, from http://amturing.acm.org/award_winners/perlis_0132439.cfm

Nordenbo, S. E. (2011). Dannelse, kompetence og uddannelse. In *Gyldendals pædagogikhåndbog* (pp. 44–65).

OECD. (2003). The definition and selection of key competencies - Executive summary. *DeSeCo*, 1–20. https://doi.org/10.1080/2159676X.2012.712997

P21. (n.d.-a). Our History. Retrieved March 24, 2017, from http://www.p21.org/about-us/our-history

P21. (n.d.-b). Our Vision and Mission. Retrieved April 23, 2017, from http://www.p21.org/about-us/our-mission

P21. (n.d.-c). The 4Cs Research Series. Retrieved March 24, 2017, from http://www.p21.org/our-work/4cs-research-series

P21. (2002). Learning for the 21 st Century: A Report and Mile Guide for 21 st Century Skills, 36. Retrieved from https://eric.ed.gov/?id=ED480035

P21. (2015). P21 Framework Definitions. *Framework*, 1–9. Retrieved from http://www.p21.org/storage/documents/docs/P21_Framework_Definitions_New_Logo_2015.pdf

Pahuus, M. (2011). Hermenuetik. In *Humanistisk videnskabsteori* (pp. 139–170). Søborg: DR Multimedie.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York, NY, USA: Basic Books, Inc.

Papert, S., & Solomon, C. (1972). Twenty things to do with a computer. *Educational Technology*, *12*(4), 9–18. Retrieved from https://dspace.mit.edu/bitstream/handle/1721.1/5836/AIM-248.pdf?sequence=2&origin=publication_detail

Parker, K. R., & Davey, B. (2012). The History of Computer Language Selection. In A. Tatnall (Ed.), *Reflections on the History of Computing: Preserving Memories and Sharing Stories* (pp. 166–179). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-33899-1_12

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction design: beyond human-computer interaction*.

New York, NY: J. Wiley & Sons.

Problemløsning. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?word=problemløsning&dict=auto

Programmering. (n.d.). In *Den Danske Ordbog*. Retrieved from http://ordnet.dk/ddo/ordbog?query=programmering

Prottsman, K. (2015). Coding vs. Programming - Battle of the Terms! [Blog Post]. Retrieved April 27, 2017, from http://www.huffingtonpost.com/kiki-prottsman/coding-vs-programming-bat_b_7042816.html

QSR International. (n.d.). NVivo product range. Retrieved May 2, 2017, from http://www.qsrinternational.com/nvivo-product

Qvortrup, L. (2005). Viden og vidensformer. In *Det vidende samfund - mysteriet om viden, læring og dannelse* (2nd ed., pp. 67–112). København: Unge Pædagoger.

Regeringen. (2016). For et friere, rigere og mere trygt Danmark. https://doi.org/978-87-93422-35-3

Regeringen/KL/Danske-Regioner. (2016). *Et stærkere og mere trygt digitalt samfund: den fællesoffentlige digitaliseringsstrategi 2016-2020*. Regeringen : KL : Danske Regioner.

Renumol, V. G., Jayaprakash, S., & Janakiram, D. (2009). *Classification of cognitive difficulties of students to learn computer programming*. Retrieved from https://pdfs.semanticscholar.org/e505/0484e9b26340afb5575f95055dbf896b4bde.pdf

Resnick, M. (2012). Mitch Resnick: Let's teach kids to code. Ted Talk. Retrieved from https://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code

Resnick, M., Maloney, J., Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, … Yasmin Kafai. (2009). Scratch: Programming for All. *Communications of the Acm*, *2009*(Vol. 52). Retrieved from http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf

Riise, A. B. (2017). Ph.d.-studerende: Ingen evidens for, at it i undervisningen virker. *Folkeskolen.dk*. Retrieved from https://www.folkeskolen.dk/601660/phd-studerende-ingen-evidens-for-at-it-i-undervisningen-virker

Rouse, M. (2013). What is STEM (science, technology, engineering, and mathematics)? - Definition from WhatIs.com. Retrieved May 23, 2017, from http://whatis.techtarget.com/definition/STEM-science-technology-engineering-and-mathematics

Rychen, D., & Salganik, L. (2003). *Key competencies for a successful life and well-functioning society*. Retrieved from https://www.google.com/books?hl=da&lr=&id=CUhfAgAAQBAJ&oi=fnd&pg=PR5&dq=key+competencies+rychen&ots=faRYfeTdi-&sig=SwaHJ4uFe2QKTfh6kPh165-OkdQ

Sáez-López, J.-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education*, *97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

Samarbejde. (n.d.). Retrieved May 14, 2017, from http://ordnet.dk/ddo/ordbog?query=samarbejde

Schnack, K. (2011). Dannelsesbegrebet i skolen. In *Gyldendals pædagogikhåndbog* (pp. 28–43).

Scratch-wiki. (n.d.). Sprite. Retrieved May 21, 2017, from https://wiki.scratch.mit.edu/wiki/Sprite

Screencastify. (n.d.). Screen Video Recording. Retrieved April 1, 2017, from https://www.screencastify.com/

Shallit, J. (1995). History of Computer Science. Retrieved May 27, 2017, from https://cs.uwaterloo.ca/~shallit/Courses/134/history.html

Shawwa, N. El. (2011). Scripting vs. Coding vs. Programming [Blog Post]. Retrieved April 27, 2017, from http://www.naelshawwa.com/scripting-coding-programming/

Siri-kommissionen, R. (2016). Kunstig Intelligens Morgendagens Job og Samfund. Retrieved from http://ida.dk/sites/default/files/kunstig_intelligens_-_morgendagens_job_og_samfund.pdf

Skills. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?word=skills&dict=a000

Smith, M. (2016). Computer Science For All [Blog Post]. Retrieved February 19, 2017, from https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

STEM to STEAM. (n.d.). Retrieved May 23, 2017, from http://stemtosteam.org/

Teknologi. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?word=teknologi&dict=auto

TIOBE Index. (n.d.). Retrieved April 28, 2017, from https://www.tiobe.com/tiobe-index/

Toikkanen, T. (2015). Coding in school: Finland takes lead in Europe. Retrieved March 11, 2017, from https://legroup.aalto.fi/2015/11/coding-in-school-finland-takes-lead-in-europe/

Trial and error. (n.d.). In *Ordbogen.com*. Retrieved from http://www.ordbogen.com/opslag.php?word=trial+and+error&dict=auto

UFM. (n.d.-a). Begreber — Uddannelses- og Forskningsministeriet. Retrieved May 7, 2017, from http://ufm.dk/uddannelse-og-institutioner/anerkendelse-og-dokumentation/dokumentation/kvalifikationsrammer/begreber

UFM. (n.d.-b). Bologna-processen — Uddannelses- og Forskningsministeriet. Retrieved May 7, 2017, from http://ufm.dk/uddannelse-og-institutioner/internationalisering/internationalt-samarbejde-om-uddannelse/bologna-processen

UFM. (n.d.-c). Europæisk kvalifikationsramme (EQF) — Uddannelses- og Forskningsministeriet. Retrieved May 7, 2017, from http://ufm.dk/uddannelse-og-institutioner/anerkendelse-og-dokumentation/dokumentation/kvalifikationsrammer/europaeisk-kvalifikationsramme-eqf

UFM. (n.d.-d). Kvalifikationsrammen for Livslang Læring — Uddannelses- og Forskningsministeriet. Retrieved May 7, 2017, from http://ufm.dk/uddannelse-og-institutioner/anerkendelse-og-dokumentation/dokumentation/kvalifikationsrammer

UFM. (n.d.-e). Niveau 1 — Uddannelses- og Forskningsministeriet. Retrieved May 7, 2017, from http://ufm.dk/uddannelse-og-institutioner/anerkendelse-og-dokumentation/dokumentation/kvalifikationsrammer/niveauer-i-kvalifikationsrammen/niveau-1

UFM. (n.d.-f). Typer af uddannelsesbeviser og grader i kvalifikationsrammen — Uddannelses- og Forskningsministeriet. Retrieved May 7, 2017, from http://ufm.dk/uddannelse-og-institutioner/anerkendelse-og-dokumentation/dokumentation/kvalifikationsrammer/typer/hardtableview

UFM. (n.d.-g). Typer af uddannelsesbeviser og grader i kvalifikationsrammen — Uddannelses- og Forskningsministeriet. Retrieved May 20, 2017, from http://ufm.dk/uddannelse-og-institutioner/anerkendelse-og-dokumentation/dokumentation/kvalifikationsrammer/typer/hardtableview

UFM. (2016). Optag 2016 It-uddannelser. Retrieved from http://ufm.dk/uddannelse-og-institutioner/statistik-og-analyser/sogning-og-optag-pa-videregaende-uddannelser/2016/notat-6-it.pdf

UVM. (2010a). Faghæfte 48 It- og mediekompetencer i folkeskolen, (5).

UVM. (2010b). Informationsteknologi C – Forsøgslæreplan. *Læreplan C-Niveau*. Retrieved from https://www.uvm.dk/-/media/UVM/Filer/Udd/Gym/PDF11/110830_Informationsteknologi_-C_-forsoegsslaereplan.ashx?la=da

UVM. (2016). Aftale mellem regeringen, Socialdemokraterne, Dansk Folkeparti, Liberal Alliance, Det Radikale Venstre, Socialistisk Folkeparti og Det Konservative Folkeparti om styrkede gymnasiale uddannelser. Retrieved from http://www.uvm.dk/-/media/UVM/Filer/Udd/Gym/PDF16/Jun/160603-Styrkede-gymnasiale-uddannelser.ashx

UVM. (2017a). Fælles Mål for folkeskolens fag og emner. Retrieved May 7, 2017, from http://www.uvm.dk/folkeskolen/fag-timetal-og-overgange/faelles-maal/om-faelles-maal

UVM. (2017b). Ny aftale giver øget frihed om Fælles Mål i folkeskolen. Retrieved May 25, 2017, from http://www.uvm.dk/aktuelt/uvm/2017/maj/170519-ny-aftale-giver-oeget-frihed-om-faelles-maal-i-folkeskolen

UVM. (2017c). Undervisningsministeren nedsætter rådgivningsgruppe for digital læring. Retrieved April 1, 2017, from https://www.uvm.dk/aktuelt/uvm/udd/folke/2017/mar/170330-undervisningsministeren-nedsaetter-raadgivningsgruppe-for-digital-laering

Wiingaard, J. (2007). kommunikation i Gyldendals Teaterleksiikon, Alette Scavenius (red.). Retrieved May 14, 2017, from http://denstoredanske.dk/Gyldendals_Teaterleksikon/Begreber/kommunikation

Wing, J. M. (2006). ComputationalThinking. *COMMUNICATIONS OF THE ACM March*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2010). Computational Thinking: What and Why? Retrieved from https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Wing, J. M. (2014). Computational Thinking Benefits Society [Blog Post]. Retrieved from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html

Wing, J. M. (2016). Computational thinking, 10 years later [Blog Post]. Retrieved February 19, 2017, from https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later/

Winther-Jensen, T. (n.d.). Dannelse. In *Den Store Danske, Gyldendal*. Retrieved from http://denstoredanske.dk/Erhverv,_karriere_og_ledelse/Pædagogik_og_uddannelse/Højskoler_og_oplysningsforbund/dannelse

Aalborg Kommune. (n.d.). Programmering naturligvis. Retrieved March 29, 2017, from http://www.programmeringnaturligvis.dk/

AAU. (n.d.). NVIVO - Aalborg Universitetsbibliotek. Retrieved May 2, 2017, from http://www.aub.aau.dk/software-web/nvivo/

FIGURLISTE

Figur 1: Scratch community (https://scratch.mit.edu/) 21

Figur 2: Scratch udviklingsværktøj 22

Figur 3: Kodeblokke, med hverdagsagtig sprog - her dansk 23

Figur 4: Testområde i Scratch 23

Figur 5: Micro:bit, programmerbar mini computer 24

Figur 6: Google for Educations beskrivelse af koncepter i Computational Thinking (Google for Education, n.d.) 26

Figur 7: Barefoot Computing's definition på computational thinking 27

Figur 8: Frit efter skema i Didaktiske modeller – grundbog i didaktik (Jank & Meyer, 2006, p. 165) 32

Figur 9: Indplacering af uddannelses beviser på de 8 niveauer (UFM, n.d.-g) 34

Figur 10: Beskrivelse af niveau 1 i den danske kvalifikationsramme for livslang læring (UFM, n.d.-e) 35

Figur 11: Fagformål for fysik/kemi (EMU, n.d.-c) 36

Figur 12: Kompetencemål for fysik/kemi (EMU, n.d.-b) 36

Figur 13: Kompetenceområder for kompetencemålet undersøgelse (EMU, n.d.-e) 36

Figur 14: Viden- og færdighedsmål (målpar) inden for kompetenceområdet Produktion og teknologi (EMU, n.d.-e) 37

Figur 15: Vejledende eksempel på læringsmål i natur/teknologi (EMU, n.d.-a) 38

Figur 16: Ovortrups vidensformer (Qvortrup, 2005) sat sammen med folkeskolens formål (Ministeriet for Børn Undervisning og Ligestilling, 2016) og livslang læring (UFM, n.d.-d) 39

Figur 17: The DeSeCo Project's conceptual framework for key competencies (OECD, 2003, p. 5) .. 43

Figur 18: P21 framework (P21, 2015) 44

Figur 19: 21st Century Learning Design Rubricks og 21skills.dk 45

Figur 20: Synthesis of 15 different 21st century learning frameworks into one visual image (Kereluik et al., 2013) 47

Figur 21: Vores oversættelse af fig. 2.5 The Experiential Learning Cycle (Kolb, 2015, p. 51) 51

Figur 22: IKT-didaktisk design 54

Figur 23: Task board 55

Figur 24: "Interrelationship Between Data Collection and Analysis" (Corbin & Strauss, 2015, p. 8) 61

Figur 25: Oversigt over vores dataindsamlings- og analyseproces trin for trin 63

Figur 26: Omfanget af kilder 65

Figur 27: Benyttede kilder 66

Figur 28: Omfanget af bearbejdet data til analyse 67

Figur 29: Åben kodning alle grupper 70

Figur 30: Aksekodning af videoobservationer de fire grupper 72

Figur 31: Aksekodning gruppe E 73

Figur 32: Aksekodning gruppe F 73

Figur 33: Problemløsende gruppe E og F 75

Figur 34: Problemløsning gruppe F 76

Figur 35: Problemløsende gruppe E 77

Figur 36: Programmering gruppe E og F 80

Figur 37: Computational thinking gruppe E og F 81

BILAG

Bilag 1: Mailkorrespondance med Coding Pirates 104
Bilag 2: Eliteinterview, Jakob Harder (STIL) 105
Bilag 3: Eliteinterview, Brian Ravnborg Christensen (CFU) 107
Bilag 4: Samtykkeerklæring Masterprojekt 109
Bilag 5: Samtykkeerklæring interviews 109
Bilag 6: Skabelon til noter til deltagerobservationer 111
Bilag 7: Spørgeguide til interview med grupperne 112
Bilag 8: Skabelon til eliteinterview 115
Bilag 9: Spørgsmål til undervisningsministeriet og STIL 116
Bilag 10: Spørgsmål til Brian Ravnborg Christensen 117
Bilag 11: Skabelon til søgeprotokol 118
Bilag 12: Søgeprotokol, 21st century skills 118
Bilag 13: Søgeprotokol: Scratch og programmering 121
Bilag 14: Søgeprotokol: Kodning i folkeskolen i udlandet og DK 124
Bilag 15: Søgeprotokol: 21st century skills og programmering 124
Bilag 16: 21.03.17 Deltagerobservationer 127
Bilag 17: 28.03.17 Deltagerobservationer 129
Bilag 18: 30.03.17 Deltagerobservationer 129
Bilag 19: Videoobservation - E310317 132
Bilag 20: Videoobservation - F280317 139
Bilag 21: Videoobservation - F310317 156
Bilag 22: Videoobservation - E040417 171
Bilag 23: Videoobservation – E280317 175
Bilag 24: Videoobservation - E300317 178
Bilag 25: Fb-interview 181
Bilag 26: H-interview 184
Bilag 27: E-interview 191
Bilag 28: Alle deltager- og videoobservationer samt interviews 200