HØGSKOLEN STORD/HAUGESUND

Bokmål

EKSAMENSINNLEVERING

Opplysningene finner du på din StudentWeb

Emnekode: Masteroppgave IKT i læring

Emnenavn: MASIKT-OPG

Eksamensdel: Masteroppgave IKT i læring

Leveringsfrist: 15.11.2016 Høsten 2016

Kandidatnummer: 2

Veileders navn: Anders Grov Nilsen

Vi minner om regler for plagiering og fusk, husk spesielt på:

1. At teksten ikke tidligere har vært brukt til skriftlige innleveringer ved Høgskolen Stord/Haugesund eller annet lærested.
2. At du ikke gjengir andres arbeid uten at dette er oppgitt ved litteraturhenvisning.
3. At du ikke gjengir eget tidligere arbeid uten at dette er oppgitt ved litteraturhenvisning.
4. At du oppgir alle referanser/kilder (også hentet fra Internett) i litteraturlisten.
5. At du markerer sitater med anførselstegn eller innrykk og anviser hvor sitatet er hentet.

Brudd på disse punktene er å betrakte som fusk. Vi viser til Lov om universiteter og høgskoler § 4-7 og § 4-8, Forskrift om studier og vurdering for Høgskolen Stord/Haugesund § 5-1 og til og Retningslinjer for behandling av fusk ved Høgskolen Stord/Haugesund.

Høgskulen Stord/Haugesund

Master IKT i læring – MASIKT-OPG

Høsten 2016

Egil Bjørnevoll

Forord

At jeg nå sitter og skriver dette forordet markerer at masterarbeidet mitt begynner å bli ferdig. Det ble mer utfordrende enn jeg hadde sett for meg. Jeg har alltid hatt det sånn at når presset blir stort nok, når tidsfrister nærmer seg, har jeg hatt et ekstra gir, som har slått inn og jeg har kommet meg igjennom det.

Men det å skrive en masteroppgave var nok hakket for stort til å jobbe på denne måten, fristene har kommet og gått to ganger, uten at jeg har vært i nærheten av å kunne levere. Nå er jeg altså ferdig, og det er en veldig lettelse.

Jeg vil gjerne rette en stor takk til Anders Grov Nilsen, som er min veileder i dette arbeidet. Han har vært tålmodig, og oppgaven har blitt til i et tempo som har fungert bra for meg. Anders har også vært flink til å gi gode tilbakemeldinger for å få meg videre i arbeidet.

Støtten fra min arbeidsgiver, Ski ungdomsskole, har også vært av stor betydning for at jeg klarte å fullføre dette. Jeg har blitt møtt med fleksibilitet og forståelse hver eneste gang jeg har bedt om noe!

Som så mange andre, har jeg opplevd at det å kombinere studier med jobb og familieliv har vært en stor utfordring, og jeg hadde nok ikke kommet i mål med dette om det ikke hadde vært for tålmodigheten og støtten fra min kone Trine, som har trådt til ekstra i forhold til alt som må følges opp i forhold til barna våre. En stor takk til deg også!

Ski, 13/11-2016

Egil Bjørnevoll

Sammendrag

Formålet med denne studien var å undersøke hvordan programmeringsaktiviteter er organisert for barn i grunnskolealder i kodeklubber og enkelte skoleklasser og hva barna som deltar lærer av å arbeide med programmering.

Forskningsspørsmålene oppgaven søker svar på er: 1) Hvilke erfaringer har man fra kodeklubbkonseptet? og 2) Hvilken funksjon kan koding ha for barn i grunnskolealder?

Dette har jeg undersøkt ved hjelp av en kvalitativ, fenomenologisk tilnærming. Jeg har intervjuet fem personer som har drevet mye med dette, to lærere og tre kodeklubbveiledere. De forteller om sine erfaringer fra denne aktiviteten og forteller om hva de tenker er den viktigste læringen for barna som deltar.

Interessen for å undersøke dette kom fra at mine egne første møter med datamaskinen var gjennom enkel programmering, og at vi som hadde dette som hobby var godt forberedt da det å bruke datamaskiner og etter hvert internett ble grunnleggende kompetanse for å fungere i arbeidsliv, skole og i samfunnet for øvrig.

Programmeringsaktivitet for barn og kodeklubber har blitt stadig mer utbredt i løpet av de senere årene, og det har også kommet mer og mer inn i skolen i mange land.

Det jeg har funnet er at de voksne som bruker slik aktivitet mye forteller at barna lærer om hvordan forskjellig programvare virker, de øver opp det vi kaller algoritmisk tankegang, de får utfolde seg kreativt med datamaskinen og de får et basisgrunnlag i programmering som kan gi dem motivasjon for å velge teknologifag når de siden skal velge karriere.

Konklusjonene mine handler om at hvis, eller når, programmering blir en vanlig aktivitet i norsk skole, vil kompetansemål som handler om det som på engelsk heter *Computational thinking*, oversatt til norsk er det *algoritmisk tankegang*, være å foretrekke framfor vanlige programmeringskonsepter. Fra erfaringene til kodeklubbene ser jeg også for meg at et fokus på spill, lek og kreativitet er det beste for å fenge elevene og holde på interessen. Utfordringene i forhold til å begynne med programmering i norsk grunnskole vil spesielt handler om at mange lærere ikke har kompetansen som kreves for å undervise i et slikt fag. I denne sammenheng kan det

være nyttig å tenke andre yrkesgrupper inn i skolen, det er mange frivillige i kodeklubbene som kanskje ville vært med på et opplæringsprogram av lærere.

Abstract

The objective of this study was to explore how programming activities for school aged children is organised in code clubs and some schools in Norway and what the children learn from doing these activities.

My research questions are: 1) What experiences do we have from the Code Club Concept in Norway? 2) What purpose does teaching coding to school aged children serve?

I have used a qualitative and phenomenological method and interviewed five people who have experience with teaching programming to children, two teachers and three code club volunteers. They share their experiences from this activity and what they believe are the most important things the children learn.

My personal objectives for doing this study was partly how I first experienced computing back in the 1980s, through simple programming. I, and many others who were very interested in computing when we grew up, believe we were well prepared for the digital society we now live in.

Learning to code, both in code clubs and at school has become increasingly popular during recent years and it is part of the School Curriculum in many countries.

My findings are that the adults who are involved in coding activities express that the children learn a lot about how different computer programs work and they strengthen their computational thinking. They also get a basic knowledge in computer science/programming, which can be a motivation for choosing Computer Science/Engineering when it is time for them to determine what higher education to choose.

My conclusions are about that if, or when, programming becomes a mandatory activity in Norwegian schools, the objectives for it should be mainly about computational thinking rather than programming concepts. Based on the experiences from code clubs, I believe that teaching programming in schools should be about games, play and creativity. One of the challenges related to introducing programming in Norwegian schools is the teachers' competence when it comes to teaching such a subject. In relation to this, it can be useful to design a model where people with technological rather than pedagogical background become involved. There are many volunteers in the code clubs who could be interested in helping out with a trainee program for teachers.

Innhold

Forord 3

Sammendrag 4

Abstract 5

1. Innledning 10

1.1 Temaets aktualitet 10

1.2 Programmeringens plass i skolen 12

1.2.1 Læreplanen i England. 13

1.2.2 Estland 15

1.2.3 Læreplanarbeid i Norge 16

1.3 Formål med studien, problemstilling og forskningsspørsmål 17

1.4 Noen viktige refleksjoner 18

1.5 Programmeringsverktøy som ofte brukes i kodeklubber og skole 19

1.5.1 Blokkbaserte verktøy 19

1.5.2 Tekstbaserte verktøy 21

1.5.3 Fysiske gjenstander som kan programmeres 22

1.6 Tidligere forskning 22

1.6.1 Litteraturgjennomgang 23

1.6.2 Erfaringer og opplevelser 24

1.6.3 Programmering for å lære faginnhold 26

1.6.4 Hva lærer barn som programmerer? 27

2. Teori knyttet til programmeringsundervisning 29

2.1 Konstruksjonisme og Papert 29

2.2 Computational thinking/algoritmisk tankegang 32

2.3 21st Century Skills 37

2.3.1 Fagspesifikk kompetanse 37

2.3.2 Å kunne lære ... 38
2.3.3 Å kunne kommunisere, samhandle og delta ... 38
2.3.4 Det å kunne utforske og skape ... 38
2.4 Motivasjon og programmering ... 40
3. Metode ... 42
3.1 Kvalitativt forskningsdesign ... 42
3.1.1 Kvalitativt forskningsintervju ... 43
3.1.2 Utvalg ... 43
3.2 Gjennomføring av intervjuer ... 46
3.3 Transkribering og analyse av intervjuer ... 47
3.4 Validitet og reliabilitet ... 49
3.5 Etikk ... 49
4. Funn ... 51
4.1 Erfaringer fra kodeklubb/programmeringsundervisning ... 51
4.1.1 Praktisk tilrettelegging i kodeklubbene ... 51
4.1.2 Veiledernes motivasjon ... 52
4.1.3 Antakelser om barnas motivasjon ... 53
4.1.4 Erfaringer med programmeringsaktivitet i skolen ... 54
4.1.5 Veilederrollen ... 56
4.2 Hvilken funksjon kan koding ha for barn i skolealder ... 58
4.2.1 «Fikling» ... 58
4.2.2 Kreativitet ... 59
4.2.3 Debugging ... 60
4.2.4 Utholdenhet ... 61
4.2.5 Samarbeid og delingskultur» ... 61
5. Drøfting. ... 63

5.1 Erfaringene fra kodeklubb og koding i skolen 63

5.2 Hva kan barn lære av å drive med koding? 67

6. Avslutning 71

6.1 Konklusjon 71

6.2 Avgrensninger 74

6.3 Veien videre 74

7. Kilder og referanseliste 76

Vedlegg 1 - intervjuguide 83

Vedlegg 2 – Kvittering fra NSD omkring behandling av personopplysninger 85

Vedlegg 3 – Informasjonsskriv til deltakere 86

Vedlegg 4 – Matrise for litteratursøk 87

1. Innledning

1.1 Temaets aktualitet

På hvilken måte skal det undervises i forskjellige digitale ferdigheter? Etter hvert har det kommet mange kritiske røster mot at elevene først og fremst lærer å bli konsumenter av digitalt innhold og ikke produsenter. For eksempel argumenterer boka «Program or be Programmed – ten commandments for the digital age,» (Rushkoff, 2010) sterkt for at det er skummelt for oss alle, at så få egentlig forstår teknologien som finnes overalt. Det er problematisk at de som mestrer den vil få, eller har allerede, for stor makt over livene til vanlige folk. Denne bekymringen kan også gjelde også norske skoleelever. De lærer seg å skrive i Word, lage presentasjoner, bruke regneark, søke og bruke innhold fra internett (med et økende fokus på nettvett, kildebruk og kildekritikk) og gjøre øvelser og tester på datamaskinen. Privat bruker de datamaskiner, mobiltelefon eller nettbrett til sosiale medier, spill og annen underholdning (Bakken, 2015). Men det er også viktig å forstå hvordan teknologien vi omgir oss med virker, og derfor har programmeringsundervisning av barn i skolealder blitt et svært aktuelt tema.

I 2014 hadde 12 land i Europa allerede programmering i sine nasjonale læreplaner mens 7 andre var i ferd med å implementere det. (EUN, 2014). For eksempel var England tidlig ute og har innført programmering på alle trinn organisert i et eget fag som på de fleste skoler heter «Computing.»

Det er også vanlig mange steder å ha et fritidstilbud for teknologiinteresserte barn/unge, og bevegelser som Lær Kidsa Koding[1] og code.org arbeider for at så mange som mulig skal kunne lære seg å programmere. Det er vanskelig å spå noe om framtiden, men at internett og ulike datamaskiner vil spille en stor rolle er det ikke tvil om. Så hva må da skoleelevene lære? Hvilke ferdigheter på datamaskinen trenger de for å få seg en relevant utdanning og kunne finne noe meningsfullt å jobbe med i det 21. århundre? Norge har ikke satset på programmering som skoleaktivitet, eller valgt å inkludere dette i rammeverk for grunnleggende ferdigheter. Fra høsten 2016 settes

[1] Kidsakoder.no

det i gang et forsøk med programmering som valgfag på ungdomstrinnet. (Regjeringen, 2015), og 146 skoler deltar i dette.

På 1980-tallet begynte hjemmedatamaskiner og mikrocomputere og få en viss utbredelse. Siden den gang har datamaskiner blitt noe omtrent alle husstander har en eller flere av og i tillegg har vi fått mindre mobile enheter som avanserte mobiltelefoner og nettbrett som i seg selv er kraftige dataverktøy. I Norge i dag er det f.eks. 95 datamaskiner, 116 mobilabonnementer og 96 internettbrukere per 100 innbyggere (Globalis.no, u.å).

I Storbritannia har de introdusert faget «Computing» som inkluderer programmeringsaktivitet på alle trinn, som en erstatning av faget som het «ICT» som hadde mer fokus på grunnleggende bruk av datamaskinen. Den viktigste grunnen for denne endringen var at det var stor bekymring knyttet til sviktende søkertall til høyere utdanning i teknologifag og konkurransevilkårene i teknologibransjen (Brown et al., 2014). På samme måte har også teknologibransjen i USA presset på for å få mer fokus på det de kaller STEM-fagene i skolen (Science, Technology, Engineering and Mathematics). I sin årlige «State of the Union»-tale i 2016 lanserte president Barack Obama en intensjon om å innføre et informatikkfag for alle også i den amerikanske grunnskolen (M. Smith, 2016).

Partnership for 21st Century skills er en av aktørene som har tatt på seg oppdraget å forsøke å definere hva fremtidens skole bør inneholde. Partnerskapet ble grunnlagt i 2002 og består av aktører fra næringsliv, politikk og skole. Dette har resultert i en del konkrete ting en læreplan for framtiden bør bestå av. Det første elementet er at det legges opp til en skole hvor man har definert basisfag som ikke er så ulike de man har de fleste steder i dag, men i tillegg tverrfaglige emner som miljøbevissthet, global forståelse, finansiell/økonomisk/entreprenør-kompetanse, medborgerskap og helseforståelse. Element nummer 2 er lærings- og innovasjonsferdigheter, som kreativitet, evne til kritisk tenkning og problemløsning. Det tredje elementet er informasjons-, media- og teknologiferdigheter. Det fjerde elementet partnerskapet mener bør inn i skolen, er livs- og karriereferdigheter (P21.org, 2015). Spesielt innenfor det som handler om kreativitet, problemløsning og teknologi har grunnleggende programmering og informatikk har en plass.

Lær Kidsa Koding (LKK) er en bevegelse som ble startet våren 2013 av Simen Sommerfeldt, Torgeir Waterhouse og Beathe Due. De mente det var nødvendig at det ble gjort en innsats for at barn skulle lære å kode blant annet fordi at forelesere på universitetet fortalte at mange startet på informatikkstudier uten å ha grunnleggende forståelse av hva programmering handler om. Dessuten bekymret de seg over hva Norge skal leve av når oljen tar slutt (Sommerfeldt, 2013). LKK har hele tiden fungert på den måten at sentralleddet bistår alle som har lyst til å starte kodeklubb med kursmateriale og praktisk hjelp for å komme i gang. Nå, i oktober 2016, er det nesten 100 kodeklubber i Norge, med god geografisk spredning (Kidsakoder, 2016).

Både økende fokus på programmering i skolen i mange land og fremveksten av kodeklubber og skapende aktivitet på datamaskiner, gjorde at jeg ble interessert i å undersøke programmeringsaktivitet for barn. Det jeg ønsket å finne ut, i dette arbeidet, var hva de som underviser barn i programmering har av erfaringer i forhold til hvordan dette skal organiseres, hvordan undervisning i lærestoffet skal foregå og hva de opplever at barna som lærer å programmere utvikler av ferdigheter på datamaskinen av denne aktiviteten. For meg er dette viktig å finne ut av fordi at mye tyder på at dette kommer mer og mer inn i skolen og da er det fint å kunne lære av erfaringer som allerede er gjort.

1.2 Programmeringens plass i skolen

Landene som allerede har tatt inn programmering og informatikk som en del av læreplanene har gjort dette på litt forskjellige måter. Det viktigste man kan lese ut fra planverkene er hva myndighetene og fagpersonene som har utformet læreplanen mener det er viktig at elevene lærer, men like viktig er hvordan og hvorfor disse mener ferdighetene skal læres. Det er sannsynlig at vi skal ha mer programmering også inn i norsk skole etterhvert og da er det viktig å unngå å være for ambisiøse slik at satsingen blir mislykket og/eller få motstand fra lærerne fordi man dytter mer inn i skolen og fagene uten å ta noe ut. Den ferske rapporten, «Teknologi og programmering for alle» (Sanne et al., 2016) anbefaler at norsk skole innfører et teknologifag, hvor programmering inngår, som obligatorisk skolefag. Utfordringene er å få lærere som har

kompetanse til å undervise i dette, definere en plass et slikt fag i en skole hvor lærerne opplever at det allerede er mye fra før og ressurser og engasjement fra skoleeier og skoleledelse. Hvis anbefalingene fra denne gruppa får gehør hos skolemyndighetene, og et slikt fag skal implementeres, kan det være en fordel at Norge var litt sent ute med å innføre programmeringsfag i skolen, siden mange andre land i Europa allerede har høstet en del erfaringer med dette som Norge kan lære av.

1.2.1 Læreplanen i England.

I Storbritannia opplevde de en fallende interesse for informatikk og teknologifag på begynnelsen av 2000-tallet, de hadde en negativ utvikling på søkertall både på universitets- og ingeniørutdanninger og på hva elevene valgte på grunnskole/videregående (Furber, 2012). Man mente at dette skyldtes to ting: At skolen i for stor grad underviste i bruk av kontorapplikasjoner og internettsøk og at det var vanskelig å rekruttere lærere med kompetanse innenfor IT fordi de hadde gode muligheter til å få bedre betalte jobber utenfor skolen. Dette førte igjen til at elevene valgte bort informatikkfag fordi at de hadde liten forståelse for hva faget gikk ut på (Brown et al., 2014).

Fra 2008 og utover fikk dette problemet etter hvert mye oppmerksomhet. Interessegrupper og teknologibransjen presset på for å få en endring. Storbritannia har hatt og har en aktiv videospill- og datagrafikkindustri, og i 2010 bestilte ministeren for kultur, kommunikasjon og kreative yrker (min oversettelse) en rapport for å kartlegge hva denne bransjen så på som nødvendige tiltak for å snu en negativ utvikling. Alex Hope og David Livingstone skrev rapporten “Next Gen. Transforming the UK into the world’s leading talent hub for the video games and visual effects industries” (Livingstone & Hope, 2011). Her beskrev de både hvilken betydning denne industrien har for Storbritannia og hva som matte gjøres i utdanningssektoren for at posisjonen skulle holdes også i framtida og kom med klare anbefalinger om at Storbritannia trengte å styrke informatikkfag (herunder programmering) både i grunnskolen og i

høyere utdanning. I 2012 lanserte utdanningsminister Michael Gove et nytt «Computing»-fag i sin tale på BETT Show[2] (Gove, 2012).

I 2014 ble ny læreplan implementert hvor fokuset er at skolen skal gjøre elevene i stand til å bruke algoritmisk tankegang og kreativitet for å forstå, og være i stand til å forandre verden. Kjernen i «Computing» er informatikk, kunnskap om prinsippene for informasjon og databehandling, hvordan digitale systemer virker og hvordan man kan bruke denne kunnskapen gjennom programmering (Department of Education, 2013).

Hovedmålene i læreplanen er at elevene skal:

- Forstå og kunne ta i bruk de fundamentale prinsippene og konseptene fra informatikkfaget, inkludert abstraksjon, logikk, algoritmer og datarepresentasjon
- Kunne analysere problemer på en algoritmisk tenkende måte og ha gjentatte praktiske erfaringer med å skrive dataprogrammer for å kunne løse slike problemer
- Kunne evaluere og ta i bruk informasjonsteknologi, inkludert nye og ukjente teknologier, på en analytisk måte for å løse problemer
- Være ansvarlige, kompetente, trygge og kreative brukere av informasjons- og kommunikasjonsteknologi.
  (Department for Education, 2013)

Altså er to av fire hovedmål i læreplanen direkte knyttet til algoritmisk tankegang og programmering og det er ganske radikalt. Det skal programmeres på alle trinn, fra barna er 5 år gamle. På ungdomstrinnet skal de skrive programmer i minst ett tekstbasert språk. De har klare fagmål på alle trinn på forskjellige dataoperasjoner barna skal lære, f.eks. variabler og løkker på «key stage» 2 (3-6. trinn) og boolske variabler[3] på «key stage 3.» Den engelske læreplanen er ambisiøs og har blitt kritisert fra forskjellig hold. En bekymring er hvilke kvalifikasjoner lærerne har for å undervise i dette. Computing at School (CAS) har opplæringsprogrammer for lærere i alle skoleslag, men dette er tenkt slik at en «mesterlærer» i et område lærer opp lærere

[2] https://en.wikipedia.org/wiki/BETT
[3] https://no.wikipedia.org/wiki/Boolsk_variabel

ved flere skoler. Dette skal de bruke ca. en dag i uka på å gjøre. Det er tvilsomt at dette er nok ressurser til å få mange nok lærere til å føle at de er trygge i dette faget (Brown et al., 2014). En annen ting mange er opptatt av, er spørsmålet om hvordan digital kompetanse i alle fag prioriteres, når det så klart legges opp til et eget datafag. Det er få av barna i en vanlig skoleklasse som kommer til å jobbe som programmerer/utvikler i framtida, men alle trenger å forstå hvordan datamaskinen virker og kunne arbeide effektivt med den (Haigh, 2016).

Spørsmålet om lærerne har kompetanse til å undervise i programmering vil også være aktuelt i Norge. Rapporten Monitor skole 2013 viser at veldig mange lærere bruker PC til administrasjon og planlegging, men at mange fortsatt bruker dem lite i klasserommet, sammen med elevene. For eksempel bruker 70% av lærerne i undersøkelsen PC seks timer eller mindre per uke i klasserommet (Hatlevik et al., 2013). Denne undersøkelsen viser at lærere er blitt gode til å bruke tekstbehandling og læringsplattform, men at det er få som har mer spesialiserte IKT-ferdigheter, som for eksempel programmering.

1.2.2 Estland

Estland har valgt en annen vei enn Storbritannia når det gjelder å lære skolebarn programmering. De har et tverrfaglig emne som heter teknologi og innovasjon, som fordrer alle lærere til å implementere teknologi i sin undervisning. Læreren står fritt til hvordan dette skal gjøres, men typisk vil det være å bruke Scratch til å arbeide med emner innenfor matematikk eller f.eks. robotbygging i kunst og håndverk. Det legges til rette for at skoler søker om midler og støtte til forskjellige prosjekter, som roboter, 3d-printing, informatikk osv. På ungdomsskole og videregående kan elever lære tekstbaserte programmeringsspråk, 3d-grafikk, lage spill, nettsider og apper. Programmering er også eget valgfag i den estiske ungdomsskolen.

Estland har gjennomført dette med et program som heter ProgeTiiger. De har definert mål for hva de ønsker skolene skal gjøre, også har de satt inn ressurser på å utvikle læremider, opplæring av lærere, utstyr til skolene, aktiviteter for at involverte lærere får truffet andre og utvekslet ideer (nettverksbygging) og arrangering av aktiviteter for elever og lærere (HITSA, 2015).

Estlands modell og gjennomføring gir lærerne muligheter og rom for å implementere programmeringsaktiviteter for elevene sine i sitt eget tempo. De har stor metodefrihet og myndighetene har først og fremst lagt til rette for en ønsket utvikling. Dette er en annen innfallsvinkel en England har valgt, hvor de valgte å ta programmering inn i et eget fag på skolen sammen med annet datarelatert faginnhold.

1.2.3 Læreplanarbeid i Norge

Fremtidens skole – fornyelse av fag og kompetanser (NOU 2015:8, 2015) var en utredning bestilt av regjeringen for å gjøre anbefalinger hva Norge trenger av kompetanser og fag i framtida. Utvalget som arbeidet med denne utredningen, Ludvigsen-utvalget, anbefaler fire kompetanseområder som er viktige for fornyelse av skolens innhold:

- Fagspesifikk kompetanse
- Kompetanse i å lære
- Kompetanse i å kommunisere, samhandle og delta
- Kompetanse i å utforske og skape

Fagspesifikk kompetanse handler om at elevene skal utvikle kompetanse innenfor sentrale fagområder og få et fundament for ulike utdannings- og yrkesvalg. Naturfag og teknologi nevnes spesielt, og man kan se for seg at programmering kan få en plass innenfor et slikt fag. Dessuten skrives det om evne til å tenke kritisk og å kunne løse problemer, noe som også undervisning i programmering vil kunne legge til rette for.

Programmeringsaktivitet kan være utforskende og kreativt, og har også en plass under dette kulepunktet. Kompetanse i å utforske og skape handler om at elevene kan komme ut i framtidas arbeidsliv og være rustet til å være innovative og skapende. For å utvikle slike egenskaper må elevene utfordres i forhold til problemløsning og lære at det finnes ulike veier til målene.

Anbefalingene fra Ludvigsen-utvalget resulterer først og fremst i en fornyelse av Kunnskapsløftet (UDIR, u.åb). I Stortingsmelding 28 (2015-2016), «Fag-Fordypning-Forståelse – En Fornyelse av Kunnskapsløftet» (Kunnskapsdepartementet, 2016) står det at tankegangen rundt grunnleggende ferdigheter videreføres. Men videre står det

at i lys av økt internasjonalt fokus på at det å kunne produsere innhold på datamaskinen er minst like viktig som å kunne konsumere innhold, bør undervisningen i digitale ferdigheter styrkes. I tillegg til dette generelle, som både vil kunne få en innvirkning på tverrfaglig tenkning og faginnhold i enkelte skolefag, så går stortingsmeldingen inn for programmering som valgfag i ungdomsskolen (som det blir satt i gang forsøk med høsten 2016,) og programmering mer inn i faget informasjonsteknologi på videregående skole (UDIR, u.åa).

Programmering har fått en plass i skole og planverk og mange land, og det er også et økende fokus på dette i Norge. Ludvigsenutvalgets rapport er ikke konkret på i hvor stor grad norske skolebarn skal lære å programmere, men ekspertgruppa som laget rapporten «Teknologi og programmering for alle,» (Sanne et al., 2016) anbefaler klart at Norge må satse på et teknologifag hvor programmering inngår. At formuleringen «å gå fra å konsumere innhold til å produsere» er brukt i Stortingsmeldingen om en fornyelse av Kunnskapsløftet (Kunnskapsdepartementet, 2016) er også et sterkt signal om at regjeringen vil ha en endring i hva elevene skal lære av ferdigheter på datamaskinen.

1.3 Formål med studien, problemstilling og forskningsspørsmål

Siden det å lære enkel programmering kan være svært nyttig for norske skolebarn og for framtidens arbeidsmarked, ønsker jeg å finne ut mer om hvordan programmeringsundervisning foregår i Norge og hva deltakerne får ut av det. Jeg ønsker videre å se på hvordan erfaringene fra kodeklubb og lærere som har prøvd ut programmering i klasserommet kan være nyttige for lærere som ønsker å ta verktøyene i bruk i skolen eller hvis det kommer inn, enten som et eget fag eller som en fagovergripende kompetanse eller en del av de grunnleggende ferdighetene.

Problemstilling:

Hvordan kan erfaringer fra kodeklubbkonseptet og forsøk med programmering i klasserommet være med å utvikle elevenes programmeringskompetanse i skolen?

Forskningsspørsmål:

1. Hvilke erfaringer har voksne som driver med programmeringsundervisning og kodeklubb?

2. Hvilken funksjon kan koding ha for barn i skolealder?

1.4 Noen viktige refleksjoner

Forskjell på programmering og koding?

Igjennom denne oppgaven bruker jeg begrepene koding og programmering litt om hverandre. Likevel må det understrekes at begrepene har litt forskjellig klang hos de som er involvert i informatikkyrker, spillutvikling, apputvikling osv. Opprinnelig var det slik at de som hadde tatt utdanning og jobbet i programvareselskaper og laget programmer kommersielt ble kalt programmere, mens de selvlærte amatørene, hackerne, drev med koding. Jonah Bitautas argumenterer for at programmering handler om helhetsvurderinger, design og mye mer enn bare å skrive inn kode, og de som bare koder kjenner verktøyene og arbeider på et produksjonsnivå (Bitautas, 2013). Siden har dette skillet blitt overført til bevegelsene som er opptatt av at barn skal lære dette, men der har det blitt slik at det lekende og utforskende med å lage småspill og animasjoner ofte blir kalt koding (Sevik, 2015). Jeg vil i denne oppgaven bruke begrepene om hverandre, siden de praktiske forskjellene er små og at i mine kilder, både teori og innsamlet data, skilles det lite mellom dem.

Computational Thinking = Algoritmisk tankegang

Computational thinking er et begrep som ofte blir brukt når i artikler om programmering og barn. Det omhandler hvordan vi kan angripe utfordringer ved å tenke som en dataingeniør. Artikkelen *Computational Thinking* (Wing, 2006) har hatt stor innflytelse på tankegangen i både fagartikler og læreplaner. Begrepet det norske fagmiljøet har landet på, er *algoritmisk tankegang* (Sevik, 2015). I Sverige brukes begrepet «datalogiskt tänkande,»[4] som man kan se på som en litt mer presis

[4] https://sv.wikipedia.org/wiki/Datalogiskt_t%C3%A4nkande

oversettelse siden algoritmisk tankegang er et konsept i seg selv innenfor Computational thinking i mye av det som er skrevet om dette. Det svenske begrepet favner litt videre. Uansett, så er det viktigste man gjør, når man tenker som en dataingeniør, å bryte ned komplekse problemer til mindre, mer håndterbare biter, og nærme seg en løsning på den måten, derfor fungerer begrepet *algoritmisk tankegang* godt i denne sammenhengen.

1.5 Programmeringsverktøy som ofte brukes i kodeklubber og skole

Det finnes svært mange programmeringsverktøy som er bygget opp spesielt for å lære brukeren programmering. Begynnerverktøyene, som kan tas i bruk for barn ned i førskolealder, er gjerne blokkbaserte, dvs. at man programmerer ved hjelp av brikker med forskjellig funksjon som gjør ulike ting i programmet. Det brukes også forskjellige tekstbaserte språk, som ligner på, eller er mer «vanlige» programmeringsspråk. Her skriver man kommandoer i tekstformat med en bestemt syntaks når man programmerer.

Det finnes mange språk å velge imellom, i dette kapittelet presenterer jeg bare de som mine informanter har erfaring med og hvor en finner tips og undervisningsopplegg på Lær Kidsa Koding sitt nettsted (kidsakoder, u.å).

1.5.1 Blokkbaserte verktøy

Scratch

Mange norske kodeklubber bruker programmet Scratch[5], som er utviklet ved Lifelong Kindergarten Group ved MIT. Til å begynne med var dette et gratisprogram for nedlasting, men nå er det først og fremst nettbasert. Det er likevel mulig å laste det ned, slik at det kan brukes uten nettilgang hvis dette er en utfordring. Programmeringen foregår ved at man bruker blokker som inneholder forskjellige kommandoer og drar og slipper inn i en handlingsrute. Blokkene er sortert ut fra

[5] https://scratch.mit.edu

funksjon, og settes sammen som legoklosser. Logo[6], som regnes som det første programmeringsspråket laget for at barn skulle lære koding, hadde «low floor – high ceiling» som hovedprinsipp. Dette innebar at det skulle være enkelt å komme i gang med, men også ha muligheter til å lage avanserte programmer. Dette er adoptert i Scratch, men man snakker heller om «low floor – wide walls,» som handler om at de har laget et verktøy som har som mål at brukeren skal få utforske et bredt spekter av muligheter som ligger innenfor dataprogrammering (Resnick & Silverman, 2005). Dette ser man for eksempel ved at det er mulighet til å integrere Scratch med Picoboard[7], som er et sensorbrett som kan kobles til datamaskinen, og Lego WeDo[8], som er Lego sitt robotbyggesett laget for utdanningsformål. En kan bruke lyd, webkamera, tastatur og mus til å styre programmene med.

Code studio

Dette er også et nettbasert verktøy utviklet av organisasjonen code.org. Det var tvillingene Hadi og Ali Partovi som startet organisasjonen med formål om fornyet interesse for teknologifag og arbeide for å få dette inn i skolen. Organisasjonen har mektige støttespillere som for eksempel Microsoft og Facebook, og står bak Kodetimen[9] som er en verdensomspennende aksjon for å få flere skolebarn til å bli kjent med koding. Programmeringen her er også blokkbasert, men tilnærmingen er litt forskjellig fra Scratch; her blir barnet introdusert for hva blokkene gjør gradvis, de begynner med svært enkle kommandoer også blir nye blokker gradvis introdusert. I Code Studio får barna opp koden som de programmerer med blokker som JavaScript og dette kan brukes til å introdusere tekstbasert koding. Code Studio bruker kjente karakterer fra spill- og filmverdenen i de forskjellige leksjonene. F.eks. Angry Birds, Frost og MineCraft.

[6] https://en.wikipedia.org/wiki/Logo_(programming_language)
[7] http://www.picocricket.com/picoboard.html
[8] https://education.lego.com/en-us/elementary/shop/wedo-2
[9] https://hourofcode.com/no

Kodu

Kodu er en plattform for å programmere egne spill i et 3d-miljø. Microsoft står bak dette produktet som først kom på spillkonsollen Xbox men ble også fort tilgjengelig som gratisprogram på PC. Fordelen med Kodu er at det er visuelt tiltalende, brukeren kjenner igjen elementer fra vanlige 3d-spill og det er lett å kjøre simuleringer av hvordan programmet vil fungere kontinuerlig i arbeidsprosessen.

Learn-to-mod

Spillet Minecraft er veldig populært blant barn og unge. I utgangspunktet er det en spillverden med enkel og lett gjenkjennbar grafikk, hvor spilleren bygger opp egne verdener med hus, husdyr, mange slags byggematerialer osv. Målet med spillet er for de fleste å overleve, utforske og bygge, men man kan også «vinne» spillet ved å nedkjempe en drage. Siden spillet er så åpent og lar seg modifisere er det blitt svært vanlig å lage egne programtillegg og dermed utvikle spillet i nye retninger (Wikipedia, 2016). Dette kalles å «modde». Minecraft er Javabasert, så modding har stort sett vært gjort av folk som kan programmere i Java eller Javascript.

Learn-to-Mod er en plattform som er utviklet for at barna selv skal lære seg å modde Minecraft. Dette kan gjøres både blokkbasert eller i JavaScript. Kodeklubber som har arrangert kurs i modding, enten med Learn-to-Mod eller mer tekstbaserte ComputerCraft, opplever stor interesse og motivasjon.

1.5.2 Tekstbaserte verktøy

Python

Dette er et tekstbasert programmeringsspråk som regnes som enkelt å lære og med mange muligheter for å lage forskjellige slags programmer. Selve syntaksen er litt enklere enn i andre språk og man kan bruker færre linjer for å utføre de enkelte operasjoner (Ceder & Yergler, 2003).

Processing

Dette er et verktøy som først og fremst lager grafikk av matematikk. Med litt enkel programmering kan man lage visuell kunst. Programmet brukes både i skoler (ofte i kunstfag), av kunstnere, designere, på utstillinger og museer. Programmet har åpen kildekode og er gratis å laste ned. Programmeringen foregår enten i Java, Javascript eller Python. (Processing.org, u.å)

Unity

Dette er en 3D spillmotor, hvor brukeren kan lage 3d-objekter og miljøer til spill. Man lager spillet i et grensesnitt som kan minne om grafiske programmer, og koder objektene i et av de profesjonelle språkene, som C# eller JavaScript. Man kan lage spill som funger både i nettleser og på telefoner.

1.5.3 Fysiske gjenstander som kan programmeres

Lego Mindstorms

Dette er først og fremst et robotbyggesett med komponenter som man kjenner igjen fra den tradisjonelle Lego-verdenen. Byggesettene består av forskjellige sensorer og motorer som kan bygges sammen til forskjellige roboter. Dette styres med en kloss (kontroller) som enten programmeres direkte eller man kan programmere på PC eller nettbrett/telefon og overføre programmene via USB, Wifi eller bluetooth til klossen. Her står brukerne ovenfor to utfordringer; både bygge en fungerende robot og å programmere den.

Arduino

Dette er en mikrokontroller med diverse sensorer som kan kobles til forskjellige typer elektronikk. F.eks. kan de brukes til å styre motorer med, slik at man kan lage enkle roboter. De finnes i litt forskjellige utgaver, men de er billige i innkjøp og kobles til PC/Mac ved hjelp av USB-kabel.

1.6 Tidligere forskning

1.6.1 Litteraturgjennomgang

Tidligere i studiet skrev jeg en litteraturgjennomgang knyttet til emnet barn og programmering. Da konsentrerte jeg meg om nye artikler, med arbeid som er lett å sammenligne med situasjonen i skoler og kodeklubber i dag. Søkeordene jeg brukte var teaching, programming, learning, constructionism, computational thinking på norsk og engelsk. Jeg brukte søkemotorene Oria, Academic Search Premier og ERIC, fordi jeg ofte fikk tilgang på fullversjon av artiklene gjennom Proxy-løsningen til HSH. Jeg satt da igjen med 97 treff som ut fra tittel og sammendrag så ut til å være relevante for meg, men reelt var det kun ca 20, fordi mange artikler dukket opp i flere av søkene og på flere av søkemotorene.

Den tidligere forskningen som er gjort på mitt område siden 2006 har mange forskjellige fokus, noen ser på problematikk knyttet til kjønn/klasse, andre prosjekt sammenligner tiltak for å nå forskjellige læringsmål hvor programmering inngår som et tiltak, evne til problemløsning og samarbeid går igjen i flere av artiklene. Denne tabellen viser mine søkekriterier da jeg søkte etter artikler, matrisen for litteratursøk (Vedlegg 4) viser antallene treff jeg hadde på de forskjellige søkeordene.

Programmering som læringsaktivitet for barn ble gjort mange steder allerede på 1970-tallet. Seymour Papert har skrevet mange artikler og bøker som omhandler dette arbeidet. For eksempel *Mindstorms: Children, Computers and Powerful Ideas (Papert, 1980),* har hatt veldig stor betydning for studenter, skoleelever og utviklere av teknologi som f.eks robotbyggesett og kontrollere. Lego kalte f.eks. robotbyggesettet sitt *Mindstorms* på grunn av innflytelsen Papert og MIT har hatt på dette feltet. Jeg har likevel valgt å se spesielt på det som foregår nå, i lys av at vi nå har internett, helt andre muligheter for grafikk og lyd og det at nå er digitale verktøy en stor del av nesten alles hverdag.

Tema	Inkludert	Ekskludert
Database	Oria, Academic search premier, ERIC	
Tid	2006-2015	Før 2006
Publikasjonstype		
Fokus	Vitenskapelige (fagfellevurderte) artikler som omhandler læringsaktiviteter hvor programmering inngår, spesielt skoleslag/aktiviteter i aldersgruppen som tilsvarer norsk grunnskole (1-10)	Bøker, bokkapitler, rapporter og konferanseinnlegg.
Type aktivitet	Opplæring i programmering rettet mot barn, enten i frivillige tilbud eller i vanlig skole.	Artikler som omhandler videregående skole, høyere utdanning eller andre utdanninger utenom utdanningssektoren. Alt som ikke er lærings- eller undervisningsrelatert.
Språk	Engelsk, norsk, svensk og dansk	
Søkeord	Teaching + programming, constructionism + programming, learning + programming, «code club» + programming, computational thinking + programming, pluss tilsvarende begreper på norsk.	Kindergarten, higher education, health
Metode	Kvalitative og kvantitative studier og aksjonsforsking	

Figur 1 Oversikt over litteratursøk

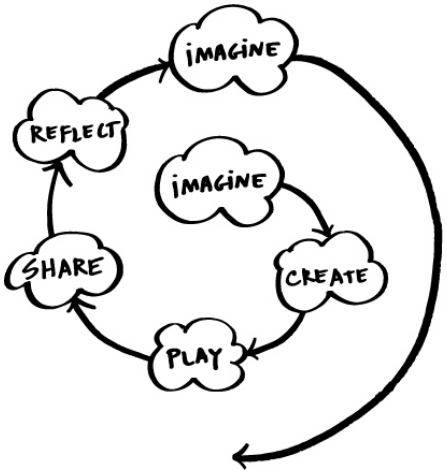
1.6.2 Erfaringer og opplevelser

Artikkelen *Code Club: bringing programming to UK primary schools* (N. Smith et al., 2014) er en evaluering av det første året med kodeklubb i England, hvor over 1000 skoler deltok. Code Club organiserte seg slik at en lærer jobbet i tospann med en person med databakgrunn på skoler etter skoletid. Læreren brakte pedagogikk og evne til å strukturere elevgrupper mens den andre hadde teknologiekspertisen. Dette ble gjort fra 2012 og utover mens arbeidet med det nye Computing-faget i skolen pågikk. I fra høsten 2014 ble det nye faget introdusert, og da hadde mange lærere allerede en del erfaringer og læringsopplevelser som gjorde dem bedre i stand til å undervise i det nye faget. I Norge har det ikke på samme måten vært en selvfølge at dette skal inn i

skolen, så her har kodeklubbene i større grad vært fritidstilbud. Erfaringene fra Code Club er at barna lærte både om digitalt design, hvordan lage og implementere dataprogrammer og de taklet flere programmeringskonsepter forbausende enkelt. (N. Smith et al., 2014).

Flere studier viser at barn som programmerer opplever dette som spennende, utfordrende og at de får positive holdninger til programmering og informatikk (Furber, 2012; Giannakos et al., 2013; Kalelioğlu, 2015; Lee et al., 2014). Barna opplever glede over at de kan manipulere datamaskinen i grafiske eller andre brukergrensesnitt. Dette er også viktig å se på i mitt datamateriale, om veilederne mener barna opplever det å programmere som positivt og motiverende.

Mitchel Resnick er leder av Lifelong Kindergarten Group som blant annet har bygget opp Scratch. Han har skrevet og vært med å skrive mange artikler og bøker, spesielt om det å bruke Scratch for å lære programmering. «Lifelong Kindergarten Group» utarbeidet Scratch, og begrepet barnehage står sentralt hos Resnick, fordi at han mener måten barn lærer på i barnehagen burde vært modell for læring andre steder. Resnicks modell (figur 2) går i spiral, og det begynner med at den lærende har en idé, så settes denne ut i praksis ved at f.eks. noe blir bygget, barnet utforsker produktet sitt gjennom lek, viser det til andre barn, får kanskje nye innspill eller nye ideer og gjør forbedringer (Resnick, 2007). Denne måten å lære på kan også foregå når barn programmerer og jeg vil se på om mine informanter i undersøkelsen har erfaringer som passer med denne modellen.



Figur 2 – Gjengitt med tillatelse fra Mitchel Resnick

1.6.3 Programmering for å lære faginnhold

I fra Norge har jeg funnet fire masteroppgaver om programmering og barn. Alle fire er undersøkelser gjort med skoleelever i forskjellig alder hvor programmeringsverktøy/roboter har blitt brukt for å se på læring og fagmål. *First Lego League og motivasjon for realfag (Jåtten, 2006)* tar for seg om hvorvidt elever som har brukt LegoMindstorms, og deltatt i First Lego league, blir mer interessert i realfag enn de som ikke har vært med på dette. Jåtten konkluderer med at det i hvert fall på kort sikt virker å være slik. *Bruk av Lego Mindstorms i natur- og miljøfag på 9. trinn (Haaland & Rosvold, 2006)* er et kvasi-eksperimentelt studium som også undersøker elevers holdninger, men i denne oppgaven spesifikt knyttet til natur og miljøfag, som det het den gangen. Haaland og Rosvold finner at motivasjonen ser ut til å øke når man bruker data og programmering, men finner ikke grunnlag for å si noe om det faglige utbyttet blir større. Begge disse oppgavene ble skrevet av studenter ved Høgskulen i Stord/Haugesund og levert i 2006. *Koding som digital grublis* (Iversen, 2015) omhandler problemløsningsstrategier og ser på om det å jobbe med oppgaver i Scratch gjør elevene (på mellomtrinnet) til bedre problemløsere. *Vi må tenke og ikke bare tegne* (Lang-Ree, 2016) handler også om matematikkundervisning og Scratch. Denne oppgaven har fokus på ulike tilnærminger til matematikk som kan brukes med Scratch og om det å bruke dette endrer elevenes holdninger til faget.

Scratch er mye brukt i kodeklubb og er også et naturlig startpunkt for barn som skal lære grunnleggende programmering, så oppgavene til Iversen og Lang-Ree er de som er mest relevante for meg. Iversen fant lite forskjell i hvor stor grad elevene klarte å arbeide problemløsende med matematikkoppgaver og Scratch, men så at i timene de programmerte, var de mer villige til å dele og diskutere løsninger. Dette delingsaspektet er jeg opptatt av også i mine undersøkelser. Lang-Ree beskriver at elevene viste engasjement, positive holdninger, skaperglede og utforskertrang da de arbeidet med Scratch, og dette ser jeg også etter når jeg drøfter erfaringer fra kodeklubb og programmeringsundervisning i kapittel 5.

Det er en del som tyder på at vi etter hvert får programmering inn i skolen i Norge. Da kommer det til å være et spørsmål om hvilke lærere som har kompetanse i å undervise

i et slikt fag, eller i fagmålene i andre fag hvor dette eventuelt skal inn. Her er det mulighet for at andre yrker kan være med å styrke opplæringen i skolen. Lektor 2-ordningen er et eksempel på en organisering hvor skolene kan få inn eksperter fra næringsliv, teknologi og realfag for å forbedre tilbudet til elevene (Kostøl & Remmen, 2016).

Kodeklubbens oppgaver ligger åpent på nett, og lærere og kodeklubber står fritt til å bruke dem. Scratch-oppgavene er i stor grad trinn-for-trinn oppgaver som barn som kan lese og klarer å konsentrere seg om dem, løser greit (figur 3.)

Figur 3 - Hentet fra kidsakoder.no. Creative Commons lisens

1.6.4 Hva lærer barn som programmerer?

Da jeg skrev litteraturgjennomgang så jeg på svært mange artikler som er skrevet om spillkonstruksjon, programmering, robotbygging og simulering gjort over hele verden. Fokuset i norske masteroppgaver har først og fremst vært programmeringsundervisning knyttet opp mot faglige mål, spesielt i realfag. Jeg intervjuer mine informanter om hva barna lærer av koding ved å være med i kodeklubb eller programmere i klasserommet og jeg ser på disse tingene i en norsk kontekst. Men det er også interessant å se på hva barna som er med på slik aktivitet må bruke av andre ferdigheter når de arbeider med dette, matematiske begreper, lesing, kreativitet og naturfag (f.eks tid og fart.) Dessuten er delings- og

samarbeidsaspektet viktig i både tidligere forskning og i det mine deltakere rapporterer.

Artikkelen «Programming in the Wild: Trends in Youth Computational Participation in the Online Scratch Community,» (Fields et al., 2014) tar for seg en undersøkelse av et tilfeldig utvalg av 5000 Scratch-programmerere som er aktive delere av prosjekter i «Scratch-samfunnet.» Det som ble undersøkt var hvor avanserte programmene til brukerne var ut fra hvilke programmeringskonsepter de tok i bruk. Tittelen på arbeidet er valgt fordi at barna som bidro til undersøkelsen ikke fikk noe systematisk opplæring eller oppfølging. Denne studien viste en sammenheng mellom hvor komplekse og store programmer brukerne laget og hvor lenge og hvor ofte de programmerte. Dette var forventede funn, men de fant at de som i minst grad brukte avanserte programmeringskonsepter var de som var nye i Scratch-samfunnet og jenter. «Skill Progression Demonstrated by Users in the Scratch Animation Environment» (Scaffidi & Chambers, 2012) omhandler en liknende undersøkelse hvor funnene viser at svært mange Scratch-brukere, som bare har Scratch-samfunnet som sitt «programmeringsnettverk», mister interessen og slutter ganske fort. Det hevdes også at utviklingen flater ut hos de som ikke dropper ut. Forfatterne stiller spørsmålstegn ved hvor vidt et sosialt samfunn er nok i seg selv for å holde barn/ungdom interesserte lenge nok til å tilegne seg grunnleggende programmeringsferdigheter. Dette er interessant fordi at det viser at alle er ikke selvgående og produktive, det er forskjeller på hvordan barn og unge gjør problemløsningsoppgaver og hvor motiverende og interessant de synes dette er. Dette er viktig å ha et bevisst forhold til hvis vi får et programmeringsfag i skolen, i kodeklubbene kommer barn/unge som er spesielt interesserte i spill og teknologi mens skolen må lære opp alle.

2. Teori knyttet til programmeringsundervisning

2.1 Konstruksjonisme og Papert

En definisjon på læring er *"the activity or process of gaining knowledge or skill by studying, practicing, being taught, or experiencing something: the activity of someone who is learning"* (Merriam-Webster, u.å).

Konstruktivisme er et læringssyn som omhandler prosessen med hvordan mening og kunnskap skapes ut fra erfaringer og opplevelser. Hvordan vi konstruerer kunnskap avhenger av hva vi vet fra før, tidligere erfaringer og hvordan vi har organisert disse i kunnskapsstrukturer. Altså handler læring om at den lærende selv aktivt må arbeide med lærestoffet for at effektive nye strukturer skal dannes. Konstruktivister mener at læreren ikke kan overføre sine fortolkninger av virkeligheten til den lærende fordi de ikke har gjort seg de samme erfaringene i livet (Jonassen & Reeves, 1996). Sosial konstruktivisme er en variant av konstruktivisme som understreker betydningen av kultur og sosiale interaksjoner i læringsprosessene. Vygotskij og Bruners teorier om barns utvikling står sentralt i sosialkonstruktivismen (Kim, 2001).

Da datamaskiner kom inn i skolen på slutten av 1970-tallet foregikk det en del programmering i Basic, Pascal og Logo. Men mye av tiden skoleelever fikk med datamaskiner ble også brukt til drilløvelser og det som kalles Computer Aided Instruction (CAI). Tanken om datamaskinen skulle erstatte læreren i situasjoner hvor det skulle pugges eller matematikkferdigheter skulle drilles var ikke ny. Det var forsøk allerede på 1960-tallet med dette (Molnar, 1997). Konstruktivistene var kritiske til CAI fordi nå hadde man tilgang på et verktøy hvor barna kunne være kreative, konstruere, bygge og utvikle sin evne som problemløsere og tenkere, og det åpnet seg så mye større muligheter. Seymour Papert skriver:

> *"One might say that computer is being used to program the child… (om CAI) …In my vision, the child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building."*
>
> (Papert, 1980).

Papert var meget sentral i bevegelsen som så på muligheten for å lære barn programmering og som tenkte at ved å lære dette får de brukt intellektet sitt, kreativiteten sin og engasjeres i en helt annen grad i lærestoffet. Han var også den som introduserte begrepet konstruksjonisme. Dette ordspillet på konstruktivisme handler om troen på at læring skjer best når barn er engasjerte i å lage personlig meningsfulle objekter og dele ideene sine med jevnaldrende (Maxwell, 2006). Papert selv beskriver i artikkelen «Situating Constructionism» (1991) hvordan han begynte å tenke på denne måten å lære på: Han var hentet inn på en skole for å lære elevene å jobbe med matematikk i Logo, men på veien til klasserommet passerte han et rom hvor en kunstklasse laget såpeskulpturer. Han ble opptatt av hvor annerledes dette arbeidet var fra matematikk: Istedenfor å måtte angripe et problem og finne en løsning, ble de gitt et materiale som de arbeidet med i lang tid. Prosessen handlet om å designe, rive ned, endre og bygge opp flere ganger, samtidig som elevene hele tiden så hvordan de andre tenkte og jobbet og utvekslet tanker og ideer med disse. Papert så dette som en veldig hensiktsmessig måte å arbeide med fagstoff på, og han fikk en drøm om at mer undervisning skulle foregå på denne måten.

Seymour Paperts teorier og forklaringsmodeller ble møtt med skepsis mange steder, spesielt at forkjemperne for LOGO argumenterte for at barn som lærte å programmere utviklet kognitive ferdigheter innenfor planlegging, problemløsning og evne til å reflektere rundt de kognitive prosessene i seg selv. Skeptikerne mente at det ikke fantes bevis for at barna som lærte å programmere utviklet evner ut over selve programmeringen (Pea & Kurland, 1984). Papert beskyldte skeptikerne for å være det han kalte teknosentriske, at de hadde misforstått ham på den måten at det ikke var maskinene som skulle revolusjonere læring og skole, men menneskene som forsto den nye teknologien. Spørsmålet bør ikke være «hvordan kan datamaskinen endre skolen, men hvordan kan mennesker endre skolen ved hjelp av teknologi?» Dessuten mente han at effektstudiene Pea og Kurland hadde tatt utgangspunkt i, da de skrev de kritiske artiklene, ikke tok tilstrekkelig høyde for langtidsvirkninger og at de målte effekt ut fra et snevert syn på hva kognitiv utvikling var (Papert, 1987). Mye på grunn av disse kontroversene ble det etter hvert færre forsøk med programmering i skolen ut over på 1990-tallet.

Konstruksjonisme omhandler læring som skjer i situasjoner hvor prøving, feiling og fikling (tinkering) er primæraktiviteten. De lærende må gjerne konstruere og dekonstruere flere ganger for å oppnå det de ønsker. Fokuset er på prosessen og samhandlingen som foregår underveis i aktiviteten (Kynigos, 2008). Dette bryter med manges oppfatning av undervisning hvor feil svar gjerne forbindes med dårlig resultat og hvor det ofte er sluttproduktet som er det viktigste.

Kafai, Burke og Resnick skriver om fire dimensjoner som kjennetegner konstruksjonistisk tankegang: sosialt, personlig, kulturelt og håndgripelig. Den personlige dimensjonen handler om hvordan kunnskap og ferdigheter blir konstruert i den enkelte når han/hun strever for å løse et problem. Det sosiale manifesterer seg i at de programmerende barna (eller voksne for den saks skyld) gjør dette i et sosialt fellesskap (enten virtuelt eller i løst eller fast organisert klubbaktivitet) og gjør det for å dele. Kulturelt kan man si at det å arbeide med koding i et konstruksjonistisk tankesett bryter med den tradisjonelle måten programmerere har arbeidet på, hvor designet og overordnet planlegging kommer før selve kodingen. Motsetningen, å begynne med det konkrete (fikle og eksperimentere) kan å fungere like godt for å lære seg programmering (Turkle & Papert, 1992). I dette kulturelle landskapet finner vi Open Source-bevegelsen[10] og det som kalles Maker-bevegelsen[11]. Den håndgripelige (tangible) dimensjonen handler om at konstruksjonistiske læremidler er innrettet mot prøving og feiling. De har en fysisk dimensjon, selv om det godt kan være på en skjerm, de kan flyttes på og manipuleres. Det finnes også mange fysiske objekter som kan programmeres, sånn som robotbyggesett, kontrollere med sensorer osv. som gir læringen en ekstra taktil dimensjon (Kafai et al., 2014).

Et annet perspektiv som i denne sammenhengen blir svært viktig er hvilken rolle den voksne tar i en setting hvor barn skal lære programmering. Det er viktig at barna programmerer selv for å lære det, men graden av instruksjon er viktig å reflektere over, hvilken rolle veilederen tar. Hvis man tenker seg at konstruktivisme eller konstruksjonisme i sin mest ekstreme form innebærer at barna får vite hva programmene kan gjøre og får en liste som omtaler de forskjellige kommandoene, og

[10] https://en.wikipedia.org/wiki/Open-source_software
[11] https://en.wikipedia.org/wiki/Maker_culture

med det utgangspunktet kan de programmere datamaskiner, er det lett å bli skeptisk. Forskning på dette feltet viser at barn som fikk opplæring i LOGO med veiledning underveis, «guided discovery,» presterte langt bedre enn barn som prøvde ut det samme i en mer ustrukturert setting (Mayer, 2004). Hvordan undervisning og veiledning foregår i kodeklubb og skole, og hvilken rolle de voksne tar, er viktig for at programmeringsundervisningen skal være god, så dette er et tema i intervjuene mine, i forhold til veilederrollen, organisering av undervisningen og arbeidsmåter forøvrig.

2.2 Computational thinking/algoritmisk tankegang

> *"Computational thinking is taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamentals to computing»*
>
> (Wing, 2006)

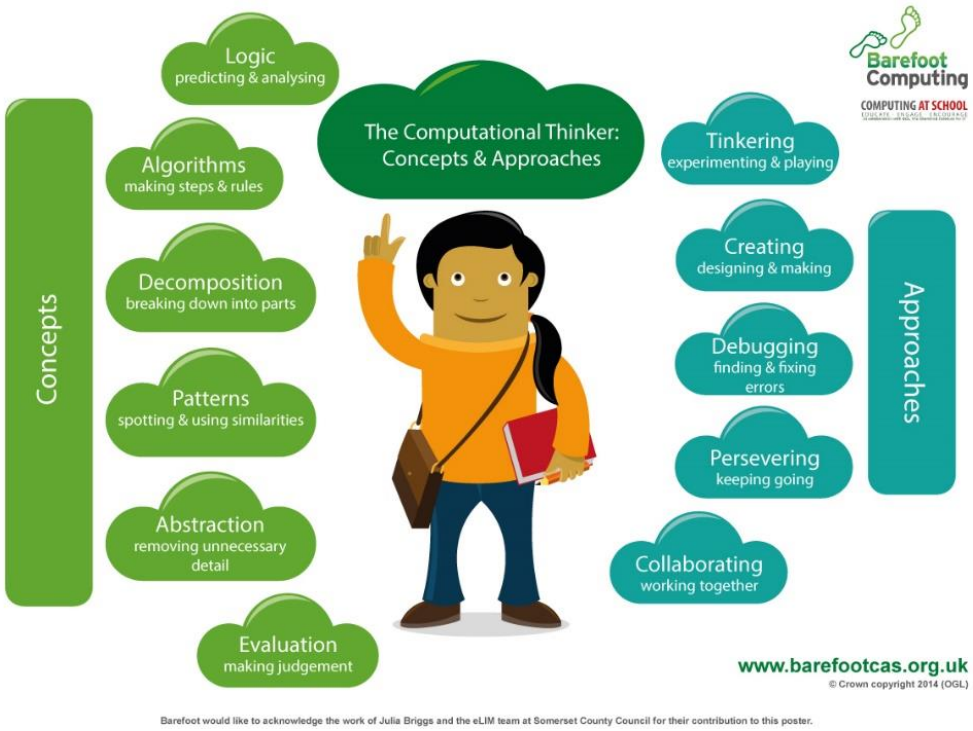
I boka *Mindstorms: Children, Computers and Powerful Ideas* (Papert, 1980), skrev Seymour Papert om å tenke som en datamaskin, og i 1996 tok han i bruk begrepet «Computational thinking» i artikkelen "An *Exploration in the Space of Mathematics Educations."* (Papert, 1996). Begrepet computational thinking, eller beskrivelser av ferdigheter/aktiviteter som forbindes med dette, er ofte sentralt i fagartikler og bøker som omhandler programmeringsundervisning av barn.

Dette har senere Jeanette Wing videreført i flere artikler og foredrag, og hun går mer i dybden på hva det vil si å tenke på denne måten. Wing mener at det er ikke et mål å lære mennesker å tenke som datamaskiner, men det handler om å løse problemer slik de som programmerer og forstår datamaskiner gjør det (Wing, 2006). Mye av det hun beskriver som computational thinking er skrevet om i læreplanverk og fagartikler som omhandler programmeringsaktivitet med barn. Det er også nedfelt i den engelske læreplanen i *computing* hvor programmering er en viktig aktivitet. I første setning i beskrivelsen av læreplanen står det: *"A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world"* (Department for Education, 2013). I forsøkslæreplanen for det nye valgfaget på ungdomstrinnet, programmering, har de oversatt begrepet til *algoritmisk tankegang.*

Dette defineres som evne til å identifisere problemer og utforme mulige løsninger, lage kode som kan forstås av en datamaskin, systematisk feilsøke og forbedre koden samt dokumentere løsningen på en forståelig måte, å kunne forutse og analysere hva et program skal gjøre, kjenne igjen mønstre, eksperimentere og evaluere mulige løsninger (UDIR, 2016).

Jeannette Wing skriver at essensen i algoritmisk tankegang er abstraksjon. (Wing, 2008). Abstraksjon handler for eksempel om å kunne generalisere ut fra like egenskaper til forskjellige objekter og å se på konkrete eksempler og formulere regler ut fra dem (Kramer, 2007). Abstraksjonene man må gjøre når man skal tenke som en datamaskin kan være kompliserte og føles mindre logiske enn de man må gjøre i vanlig matematikk. For eksempel hender det ofte, når man programmerer en datamaskin, at dataprogrammet arbeider med serier med data som ikke hører sammen i matematisk forstand, slik som f.eks. ord/setninger og tall som settes sammen og får en fornuftig output. Når et barn lager et lite spill i Scratch, lager det kanskje et poengsystem med variabler, instruksjoner med skrevet språk, input fra mus og/eller tastatur, lyd og bilde. Dette er abstraksjon på mange nivåer.

I mitt arbeid har jeg fokusert på erfaringer fra barn og programmeringsaktivitet, både på praktiske observasjoner som for eksempel hvordan veiledning skjer eller om de har brukt egne eller lånte datamaskiner. Det andre jeg har sett spesielt på er hvilke programmeringsferdigheter de tilegner seg, og mine funn peker i retning av at barna først og fremst lærer algoritmisk tankegang, mens det undervises lite i rene programmeringskonsepter. Når jeg senere i denne oppgaven skal beskrive mine funn tar jeg utgangspunkt i konseptene og tilnærmingene som Barefoot Computing, som er et program for lærere og andre interesserte som vil eller må undervise i programmering i England laget av *Computing At School*, som er et nettverk av mennesker som ønsker at barna skal få en god utdannelse i å bruke datamaskiner (CAS, 2014).



Figur 4. Hentet fra http://barefootcas.org.uk/ med tillatelse.

Her er det altså fem viktige konsepter og seks tilnærminger for å tilegne seg de konseptene. Tilnærmingene på høyre side av bildet er i stor grad representert i programmeringsaktivitet for barn, de eksperimenterer og leker, de er kreative, de må jobbe med feilsøking og debugging, de må ha utholdenhet og det er mye samarbeid. Altså er forutsetningene til stede for at barna skal tilegne seg konseptene på venstre side av bildet, men spørsmålet er om dette skjer automatisk eller i hvor stor grad de voksne bør ha fokus på de overordnede konseptene når de planlegger aktiviteten. Denne grafiske fremstillingen er ikke laget for akademiske formål, men den har tatt opp i seg hovedoverskrifter fra mye av de vitenskapelige artiklene som er skrevet om emnet. Pea og Grover (2013) skriver «CT's essence is thinking like a computer scientist when confronted with a problem." I artikkelen *New frameworks for studying and assessing the development of computational thinking* (Brennan & Resnick, 2012) bruker forfatterne følgende begreper for å beskrive situasjoner hvor algoritmisk tankegang blir praktisert (min oversettelse): 1. Gradvis ekspanderende og gjentakende, handler om at man ikke nødvendigvis har full oversikt fra starten, men bygger stein på stein. Man må også ofte gå tilbake, gjøre en liten endring og prøve igjen. 2. Testing og feilsøking, dette er en prosess som gjennomsyrer algoritmisk tankegang, det gjøres kontinuerlig. 3. Gjenbruk og remiks, som handler om deling og samarbeid. 4.

Abstrahere og modulere, som handler om å sette sammen større helheter av små enheter og forstå hvordan de virker. Denne artikkelen er skrevet spesifikt om barn som designer og lager Scratch-prosjekter, så begrepene er tilpasset det formålet.

Jeanette Wing understreker at det å definere konsepter er en pågående prosess, og at i utdanningssektoren må man tenke også på hvilken aldersgruppe som skal lære hva. (Wing, 2008). Derfor tjener den enkle og grafiske modellen (figur 1) mitt formål i denne oppgaven, og jeg vil belyse funnene mine blant annet med bakgrunn i punktene fra denne.

«Tinkering» er et engelsk begrep som ofte er brukt i litteraturen som omhandler en lekende og eksperimentell tilnærming til lærestoff, f.eks slik Mitchel Resnick (2007) tenker på fordelene med læringen som foregår i barnehagen. En «tinker» var opprinnelig en omreisende reparatør av husholdningsutstyr (kjeleflikker), og ordet «tinkering» har fått betydningen av en ukvalifisert eller eksperimentell reparasjon av saker og ting.[12] Dette gir et godt bilde av hva som kjennetegner programmeringsaktivitet i kodeklubb/klasserom, at barna prøver, feiler og eksperimenterer for å finne løsning på det de jobber med. I denne oppgaven kaller jeg denne prosessen «fikling,» og «prøving og feiling» det engelske begrepet fungerer bedre når dette skal beskrives. I artikkelen «Designing for Tinkerability» beskriver Mitchel Resnick og Eric Rosenbaum hvordan forskjellige mennesker har forskjellig tilnærming til problemer, noen angriper dem med en overordnet plan, et mål, mens andre har en vagere oppfatning av det de skal få til, og så leker og eksperimenterer de seg fram til en løsning. Førstnevnte blir en top-down tilnærming, mens den andre er bottom-up. Forfatterne mener at innenfor realfag og informatikk er det så mye fokus på evner til å planlegge og skaffe seg oversikt før man begynner å jobbe med noe, at de som er mer lekende og eksperimenterende velger bort sånne fagretninger. Forfatterne mener dette er uheldig fordi at innenfor disse feltene har de mange av største bragdene kommet som resultater av undring, utforsking og lek (Resnick & Rosenbaum, 2013).

[12] http://www.thefreedictionary.com/tinkering

Mange som jobber i teknologibransjen mener selv at de har kreative yrker. De jobber med design, de finner løsninger på uforutsette problemer og det er ofte forventet av dem at de skal ha ideer omkring det de jobber med (Knobelsdorf & Romeike, 2008). På samme måte kan barn som lærer å programmere gå inn i prosesser hvor de bruker fantasien og kan være skapende sammen med datamaskinen. Mantraet «de skal være produsenter og ikke konsumenter av digitalt innhold,» spiller en viktig rolle i hvordan man ser på hva det er viktig at barna lærer. At framtidens arbeidstakere må være kreative, er de som arbeider med å se på 21st century skills enige om (Binkley et al., 2012; Dede, 2010; P21.org, 2015), så det er viktig at kodeklubb/programmering i skolen er grunnleggende kreativt.

Det å være i stand til å effektivt effektivisere eller feilsøke koden sin er en prosess alle programmere er kjent med, det kalles *debugging*[13]. Når profesjonelle programmere snakker om dette, tenker de på store systemer, gjerne dedikerte feilsøkingsprogrammer og konsulenter som går igjennom koden som er skrevet. I mindre målestokk, handler det om å ha evnen til å isolere et problem og løse det. En fire-trinns debuggingsprosess kan være som følger: 1. Forstå at noe ikke er som det skal. Dette er selvfølgelig enkelt når ingenting skjer når man setter programmet i gang, men det kan være vanskeligere hvis problemet bare oppstår under visse forutsetninger under programmets gang. Da må man kanskje kjøre programmet mange ganger før man begynner å skjønne hva som er galt. 2. Ta en avgjørelse om målet skal justeres eller om man skal gjøre en feilsøkingsprosess for å få til det man opprinnelig hadde planlagt. 3. Gjøre antakelser om hvorfor programmet ikke virker som det skal. 4. Forsøke å løse problemet, for barn handler dette ofte om prøving og feiling (Bers et al., 2014).

Den fjerde tilnærmingen til algoritmisk tankegang er utholdenhet. Det å f.eks. programmere datamaskiner er komplekst og elementene må være helt korrekte for å fungere. Dessuten tar det lang tid, og ofte kan man jobbe i lange perioder uten å få noe feedback fra systemet. Det å holde ut i en aktivitet lenge er grunnleggende for suksess i skole/utdanning, spesielt når kompleksiteten blir stor. Utholdenhet defineres

[13] https://www.techopedia.com/definition/16373/debugging

som tiden den lærende er i stand til å holde fokus på det som skal læres (Bloom, 1968; Carroll, 1989).

Colloboration handler om samarbeid. For å klare å arbeide fram et produkt hvor algoritmisk tankegang står sentralt, er ulike former for lagarbeid viktig. Det samme er å finne på de forskjellige beskrivelsene av nødvendige ferdigheter i framtida, f.eks. Ludvigsen-utvalgets rapport, hvor det anbefales å ha evne til å samhandle og delta blir definert som grunnleggende kompetanser (NOU 2015:8, 2015). Kafai og Burke mener samarbeid er så viktig i forhold til algoritmisk tankegang og programmerings-undervisning, at vi må slutte å se på det å programmere datamaskiner som et individuelt arbeid, det er grunnleggende viktig at flere må samarbeide for å lykkes i vår tid (Kafai et al., 2014).

2.3 21st Century Skills

Samfunnet har endret seg radikalt etter at datamaskiner og internett har blitt vanlig, både i private hjem og i forskjellige yrker. Informasjonssamfunnet vi har nå er svært forskjellig fra industrisamfunnet. Hva folk arbeider med har endret seg siden ensformig fysisk og intellektuelt arbeid i større og større grad kan gjøres av datamaskiner og roboter, mens samfunnet trenger mennesker til å ta komplekse beslutninger i samarbeid med andre (Dede, 2010). Derfor er det viktig at skolen også endrer seg for at elevene skal være forberedt på framtidens arbeidsmarked.

Ludvigsen-utvalget har vurdert mange av aktørene som fokuserer på dette, og har endt opp med fire kompetanseområder som er viktig i framtidens skole: -Fagspesifikk kompetanse, -å kunne lære, -å kunne kommunisere, -samhandle og delta og -å kunne utforske og skape (NOU 2015:8, 2015).

2.3.1 Fagspesifikk kompetanse

Dette handler om at elevene skal ha kompetanse i realfag, språk, samfunnsfag og etikkfag og praktisk-estetiske fag. Innenfor realfag nevnes teknologi spesielt, så det er sannsynlig at programmering og f.eks. robotbygging vil kunne bli en del av et slikt fag.

2.3.2 Å kunne lære

Å kunne lære, metakognisjon og selvregulert læring, mener Ludvigsen-utvalget er viktig for elevene, fordi at kravet til å kunne fornye og omstille seg er sentralt i mange deler av arbeidslivet allerede, og kommer til å bli enda viktigere i framtida. Læring krever utholdenhet, forventninger til egen mestring og evne til å planlegge egen læringsprosess. Programmering er en aktivitet som fordrer utholdenhet, for å klare å fullføre et prosjekt, så dette ser jeg også etter i mine innsamlede data.

2.3.3 Å kunne kommunisere, samhandle og delta

Disse er viktige kompetanser for framtidens arbeidsliv. Ludvigsen-utvalget ser både på at evne til samarbeid må læres og utvikles, men er også opptatt av at det å lære demokratiske mekanismer og det å kunne fremme en sak er viktig. Programmeringsaktivitet er en god arena for å samarbeide, dette tar flere av mine informanter opp, og dette er interessant å se på i analysen av mine intervjuer.

2.3.4 Det å kunne utforske og skape

Dette handler i utredningen til Ludvigsen-utvalget om kreativitet, innovasjon, kritisk tenkning og problemløsning. Dette er viktige kompetanser som elevene må øve opp på skolen, og de er viktige i arbeidslivet. De som er opptatt av barn og programmeringsaktivitet rapporterer at det er stort rom for kreativitet når man driver med dette. Dessuten er problemløsningskompetanse sentral i algoritmisk tankegang og mye forskning knyttet til barn og programmering fokuserer på problemløsning. Dette vil jeg også se på når jeg analyserer og drøfter funnene mine.

Det å kunne samarbeide er viktig for fremtidens arbeidstakere, og mine informanter var opptatt av hvordan barna samarbeider når de programmerer. Parprogrammering er et begrep innenfor høyere utdanning i informatikkfagene. Forskning som er gjort på dette tyder på at universitetsstudenter som arbeidet med en partner i introduksjonskursene i programmering, ble tryggere på løsningene de valgte, var mer motivert for å gå videre med informatikkstudier, fikk bedre karakterer og laget bedre

programmer enn studenter som jobbet alene (McDowell et al., 2002). Parprogrammering er å jobbe to sammen på en datamaskin, en betjener mus og tastatur mens den andre instruerer og holder fokus på oppgaven. Siden de jeg har intervjuet beskriver at barn ofte velger å jobbe to og to og det ofte fungerer bra, er dette et viktig perspektiv når arbeid med barn og programmering skal beskrives.

STEM-fagene, Science-Technology-Engineering-Mathematics, står sentralt i begrunnelsene for at skolebarn skal lære å kode. Mye av det som har hatt betydning for å oppmerksomheten kodebevegelsen har fått, har kommet fra et ønske fra næringslivet om det skal satses mer på disse fagene på grunn av sviktende rekruttering. I tillegg har mange tatt til orde for at man skal begynne å bruke begrepet STEAM, fordi at Art også har en plass i den moderne teknologibransjen, at kunstneriske uttrykk og prinsipper kan komme teknologibransjen til nytte når de f.eks. skal uttrykke sine ideer, utforske sammenhenger ut over fagfeltet sitt eller når de skal løse vitenskapelige problemer. Tanken var også at ved å oppmuntre til, og trekke inn elementer fra kunstens verden, kunne øke de som jobber med teknologi og realfag sine evner til å tenke kreativt og finne løsninger på problemer (Catterall, 2013).

Kreativitet og innovasjon er svært viktige områder i forhold til kompetanse som trengs i framtida. Kreativitet som måte å tenke på handler om blant annet å være i stand til å være kreativ i samarbeid med andre, komme opp med nye og verdifulle ideer og å være åpen om ideene til andre, slik at fokuset hele tiden er på å forbedre det eksisterende og tilføre nye og spennende ting (Binkley et al., 2012).

Noe annet som taler for at teknologi og kreativitet kommer til å høre sammen i enda større grad i framtiden, er fremveksten av teknologi som kan integreres med andre produkter, for eksempel sensorer som er koblet til internett i klær eller utstyr som snakker med mobiltelefonen. Mulighetene som ligger her er interessante for produktutviklere både som sysler med teknologi og design. Artikkelen «Teaching Computer Science to Young Children through Creativity: Lessons learned from the Case of Norway» (Giannakos et al., 2013) handler om et prosjekt hvor en gruppe barn programmerte I Scratch og uttrykte seg kunstnerisk med gjenbrukte materialer og kombinerte disse to med Arduino-brett og sensorer.

Argumentene mot at det å kunne programmering vil være viktig i framtida, er at systemene blir stadig mer komplekse og etter hvert vil datamaskinene være i stand til å programmere seg selv (Bottou, 2014). Med nåværende utviklingshastighet, regner man med at datamaskiner vil utkonkurrere menneskehjernen innen 2045 (Cadwalladr, 2014). Slike scenarier kan både tas til inntekt for et syn om at det å ha kjennskap til programmering ikke nødvendigvis er et godt utgangspunkt i framtida og det motsatte. Det at vi i større og større grad får systemer med kunstig intelligens, som i stor grad vedlikeholder og utvikler seg selv, gjør det enda viktigere at det finnes så mange mennesker som mulig som forstår systemene.

2.4 Motivasjon og programmering

I intervjuguiden min hadde jeg et par spørsmål om motivasjon, både hvordan veilederne/lærerne opplevde barnas motivasjon når de kodet, og hvorfor de selv bruker tiden sin på å lære barn programmering. Motivasjon er et begrep vi bruker til å forklare hva som får mennesker til å gjøre noe og hva som holder aktiviteten i gang (Imsen, 2014). Dette handler om å sette mål, og å finne ut hvordan du skal nå de målene, prosessen som dreier seg om å iverksette og opprettholde mål (Schunk, 1996). Motivasjon er viktig i alle læringsprosesser fordi det er det som setter dem i gang og gir læringen kraft og retning (Grepperud, 2012; Imsen, 2014).

Sentrale begreper er ytre og indre motivasjon, hvor førstnevnte handler om belønning eller konsekvenser mens den andre handler om en indre driv for å lære og forstå noe. I skolehverdagen er både ytre og indre motivasjon viktig, men den indre er det som virker lengst og får oss til å gå inn i dypere tankeprosesser. Indre, eller naturlig, motivasjon handler for eksempel om det lystbetonte og det som gir mening (Grepperud, 2012; Imsen, 2014). Samtidig så vil ytre motivasjon, for eksempel en belønning, ofte gi en indre tilfredsstillelse. Hvis eleven får penger av foreldrene for å oppnå gode karakterer, vil pengene kunne kjøpe det de egentlig ønsker seg. For førsteklassingen å få et stjerneklistremerke i lekseboka, kan gi indre glede og følelse av mestring.

I forhold til å lære programmering kan den indre motivasjonen for eksempel handle om å mestre noe som tilsynelatende er vanskelig, eller å føle at en selv har direkte

innflytelse på hendelser på dataskjermen på en ny måte. Annen indre motivasjon kan være en følelse av at dette er meningsfullt, at man lærer noe som blir nyttig en dag. Konstruktivisme eller konstruksjonisme fokuserer på det at barnet konstruerer noe meningsfullt selv og det er aktivt i sin egen læring.

Det er forsket en del på det som heter *Game Based Learning*[14] og motivasjon, hvor funnene peker i retning av at dataspill er motiverende fordi det er noe elevene er kjent med fra fritida, det er morsomt og utfordrende og de har ofte innebygget, ytre, motivasjon i form av poengsystemer og økende vanskegrad (Mozelius, 2014; Vos et al., 2011). I samme artikkel refereres det til Malone og Lepper, som definerte fire motivasjonsfaktorer som spesielt viktige i spillbasert læring: Utfordring, nysgjerrighet, kontroll og fantasi (min oversettelse.) De fleste oppgavene[15] LKK har lagt ut på hjemmesiden sin er spill og det lære å programmere handler til å begynne med om å kunne lage spill.

Oppgavene på kidsakoder.no er trinn-for-trinn og når barna kommer inn i det, klarer de å programmere ganske avanserte spill. I forhold til teorien og tidligere forskning beskrevet i denne oppgaven, springer tenkningen knyttet til å hvorfor det er bra å lære programmering ofte ut fra konstruksjonisme/konstruktivisme. Læring skjer når den lærende utforsker og skaper et produkt. Grunnen til at oppgavene er utformet slik er at man sikrer mestringsopplevelser når barnet først begynner å programmere. Mestringsmotivasjon handler om at barnet selv får tro på at det kan få til og får en positiv selvbekreftelse av dette. Dette skaper motivasjon til videre innsats (Imsen, 2014).

[14] http://www.newmedia.org/game-based-learning--what-it-is-why-it-works-and-where-its-going.html
[15] http://kodeklubben.github.io/

3. Metode

I mitt arbeid ønsket jeg å undersøke erfaringene voksne som har arbeidet med programmeringsundervisning for barn i ulike settinger kan fortelle om. Krumsvik (2014) understreker viktigheten av at målet med studien må stå sentralt i forskningsdesignen, og deler opp dette i personlige, intellektuelle og praktiske mål med studien. Mine personlige mål handler om at jeg synes det er spennende og interessant å studere barn/unge som programmerer, fordi jeg selv tidlig var interessert i hvordan datamaskiner virket og fordi koding utfordrer både kreativitet og analytisk-matematisk tenkemåte hos den som programmerer. Jeg ønsker å finne ut mer om hvordan et programmeringsfag i skolen kan legges opp, og hvilket innhold det bør ha. Mine intellektuelle mål handler om at jeg ønsker å utforske teoretiske konsepter som omhandler læring av programmeringsferdigheter, forskning knyttet til dataundervisning for barn og hvordan voksne som jobber med å lære opp barn i å programmere tenker og arbeider for å gjøre dette best mulig. Pedagogiske refleksjoner rundt organiseringen av programmeringsundervisninger er også viktig. De praktiske målene mine handler om at det ser ut til at programmering kommer mer og mer inn i skolen. Jeg har lyst til å se på hvor det passer å inkludere ulike programmeringsaktiviteter, ut fra hva barn på ulike alderstrinn gjør i kodeklubben. Jeg vil forsøke å finne ut om skolen kan lære noe av kodeklubbene.

3.1 Kvalitativt forskningsdesign

I mitt arbeid står fortellingen til informantene sentralt, jeg ønsker altså å gå i dybden, med fokus på erfaringer og opplevelser. «Å forske kvalitativt innebærer å forstå deltakerens perspektiv. En kvalitativ forsker retter blikket mot mennesker hverdagshandlinger i sin naturlige kontekst, men dette forskerblikket blir selvsagt farget av forskerens teoretiske ståsted» (Postholm, 2010). Min problemstilling og mine forskningsspørsmål omhandler hva erfaringer fra kodeklubb og forsøk med programmeringsaktivitet i skolen kan tilføre skolen, hvilke opplevelser og erfaringer de voksne som har holdt på med dette kan fortelle om.

Dette gjør at tilnærmingen min til fagfeltet er fenomenologisk på den måten at jeg søker å finne ut hvilken mening menneskene jeg intervjuer legger i opplevelsene de har i møtet med programmerende barn. Tilnærmingen er individuell; jeg har intervjuet veilederne/lærerne en etter en. Fenomenologi er et begrep som peker på en interesse for å forstå sosiale fenomener ut fra aktørenes egne perspektiver og beskrive verden slik den oppleves av informantene (Kvale & Brinkmann, 2015).

3.1.1 Kvalitativt forskningsintervju

Fordi det er fortellingen og opplevelsene som står sentralt i mine undersøkelser, valgte jeg intervju som informasjonskilde. Grunnen til at dette passet best var at jeg ønsket å få et bilde av erfaringene og perspektivene hos de jeg intervjuet, ut fra at forskningsspørsmålene søker de voksnes fortelling og beskrivelser. Jeg benyttet det Kvale og Brinkmann (2015) kaller *et semistrukturert intervju.* Da har man en intervjuguide som inneholder hovedemnene og forslag til spørsmål og intervjueren selv hvor nøye guiden skal følges og i hvilken grad man kan følge opp informantens svar (Kvale & Brinkmann, 2015).

Krumsvik (2014) bruker et eksempel i boka om den mulige diskrepansen mellom selvrapportering og hvordan ting virkelig arter seg: En person rapporterer at han trener hardt og intensivt på treningsstudioet, mens observasjoner viser at han mest prater med kamerater og har lange pauser. I min undersøkelse spurte jeg for eksempel om hvordan informantene ser på seg selv som veiledere, og selvrapportering kan være en feilkilde. Det er viktig å være oppmerksom på dette.

3.1.2 Utvalg

Jeg ønsket at mine informanter skulle ha erfaring med programmeringsundervisning, hadde forsøkt de fleste verktøyene og har tanker både om koding i skolen og hva barna lærer av programmering. Jeg skulle altså snakke med folk som hadde erfaring fra programmeringsundervisning for barn, både fra skole og kodeklubb. Dette var viktig fordi jeg ønsket informanter som hadde sett og opplevd en del sammen med programmerende barn og at de hadde gjort seg noen refleksjoner om denne aktiviteten.

Jeg hadde tidlig bestemt meg for at masteroppgaven min skulle omhandle programmeringsaktivitet for barn og Lær Kidsa Koding-bevegelsen. Dette var landskapet jeg skulle orientere meg i. Så på høsten 2015 deltok jeg på et stort seminar i Bergen om «Dataspill og koding i skolen» i regi av IKTsenteret[16] og en lærerkonferanse i Oslo i regi av Lær Kidsa Koding[17]. Disse møtepunktene, spesielt det siste, var viktige for meg. Her fikk jeg etablert kontakt med eller ble oppmerksom på kodeklubbveiledere og lærere som kunne hjelpe meg. I tillegg ble jeg kjent med bevegelsen LKK og så og hørte ildsjeler fra inn- og utland fortelle om det de hadde gjort innenfor emnet.

Rune Krumsvik refererer til Smith og Osborn som mener at et utvalg på tre er perfekt for en nybegynner, fordi det legger til rette for å gå ordentlig i dybden på stoffet og gjøre en grundig jobb med analysene (Krumsvik, 2014). Kvale og Brinkmann poengterer at størrelsen på utvalget må bestemmes ut fra hvordan best få svar på problemstillingen (Kvale & Brinkmann, 2015). I mitt tilfelle hadde jeg et todelt fokus; jeg ønsket å få fram synet både til noen som hadde brukt programmering i skolen og på kodeklubber. Derfor valgte jeg å intervjue fem personer. Jeg kontaktet tre kodeklubbveiledere som hadde vært med omtrent siden starten i 2013 og hadde arrangert mange kurs for ulike aldersgrupper og med ulike verktøy. De har også drevet kodeklubb både på små og store steder og hatt forskjellig grad av støtte fra større institusjoner. På lærerseminaret som var arrangert av LKK fant jeg fram til en lærer som har prøvd ut Scratch i klasserommet, og av veilederen min fikk jeg tips om en annen lærer som har prøvd ut forskjellige programmeringsverktøy i skolen. Begge er meget engasjerte og reflekterte rundt programmeringsundervisning og har drevet med dette i henholdsvis 1 og 3-5 år. Jeg kontaktet alle på epost hvor jeg beskrev hva jeg ønsket å gjøre og ba om et møte. I disse mailene la jeg ved et informasjonsskriv til deltakerne med problemstilling og forskningsspørsmål, hva jeg ønsket å gjøre og hvordan jeg kom til å bruke innsamlet datamateriale (Vedlegg 3.)

[16] http://iktsenteret.no/kalender/temakonferanse-om-dataspill-og-koding-i-skolen
[17] http://kidsakoder.no/kurs_konferanse/oslosamlinghost2015/

Alle fem var svært positive til å være med i undersøkelsen min og vi fant tid for å gjøre intervjuene kort tid etter første henvendelse. Jeg har gitt forskningsdeltakerne mine fiktive navn. De to som jobber i skolen har fått navn som begynner på L (for lærer), de kaller jeg Leif og Lars. Kodeklubbveilederne har jeg gitt navn som begynner med K, Karsten, Knut og Kristian.

Kristian er 23 år gammel og har bachelorutdanning i matematikk. Han ble interessert i koding gjennom at han tok et programmeringskurs som en del av utdanningen. Han og noen medstudenter ble interessert i Lær Kidsa Koding og fikk låne en ungdomsskoleklasse i ti uker for å prøve ut opplegget høsten 2013. Kristian har vært involvert i kodeklubb i byen han studerte, drevet kodeklubb på hjemstedet sitt i feriene og har en sentral plass i LKK-bevegelsen.

Karsten er 26 år gammel og er doktorgradsstipendiat i informatikk på universitetet i hjembyen og også leder i en kodeklubb. I hans hjemby startet de opp med kodeklubb et halvt år etter Lær Kidsa Koding sentralt ble stiftet (den 27.februar 2013.) Karsten var med fra starten. Han har bred erfaring, både med kodeklubb og prosjekter rettet inn mot skole.

Mens både Karsten og Kristian har en sterk forankring til universiteter, har Knut drevet kodeklubb i litt mindre skala, på biblioteket i en mellomstor kommune på Østlandet. Han er 52 år gammel og jobber innen IT. Knut har drevet kodeklubb siden 2014. Han beskriver seg selv som nærmest maker-bevegelsen i tankegang og er glad i elektronikk og roboter.

Leif er 41 år gammel og jobber på en skole på Vestlandet. Han har mye erfaring fra IKT i skolen og har også arbeidet som programmerer. Han har vært med flere ganger som leder for lag i First Lego League[18] og brukt Scratch mye i undervisningen. Han underviser 7. trinn i programmering en klokketime i uka.

Lars er 32 år gammel og beskriver at han for alvor oppdaget datamaskinens muligheter da han begynte på lærerskolen og gjorde en oppgave i Kodu og dette tok han med seg inn i skolen. Lars er kontaktlærer på 7. trinn på en skole i en av de større byene på

[18] http://hjernekraft.org/

Østlandet og har prøvd ut Scratch i undervisningen i klassen sin. Han har gjort kodu-opplegg for mange elever på skolen, og har fått oppgaver fra sentralt hold i kommunen i forhold til programmering.

3.2 Gjennomføring av intervjuer

Etter at jeg hadde vært på seminarene høsten 2015 hadde jeg et antall navn som jeg tenkte ville gi meg svar på mine forskningsspørsmål, men for å være mere sikker på at intervjuguiden min var til god hjelp, gjennomførte jeg et pilotintervju mens jeg fortsatt ventet på svar fra NSD. Det var viktig å gjøre dette fordi jeg ønsket å finne ut om spørsmålene jeg hadde tenkt ut fungerte for å gi meg svar på forskningsspørsmålene mine, hva som fungerte godt, og hva jeg måtte endre. Jeg intervjuet en elektroingeniør som hadde vært med i kodeklubb i tre semestre et sted på Vestlandet. Han har ikke erfaring fra skole, men var opptatt av utdanning og kompetanseheving. I hjembygda hans er det den lokale TEKNA-klubben[19] som tok initiativ til å starte kodeklubb. De låner en datalab på høyskolen som finnes på stedet. I dette intervjuet fikk jeg mange svar som gikk på veldig grunnleggende, praktiske ting. Sånt som har med maskintetthet, nettverk og samarbeid internt i gruppa å gjøre. Sammen med veilederen min jobbet vi videre med intervjuguiden slik at den i enda større grad fokuserte på å få svar på problemstilling og forskningsspørsmål.

Etter gjennomgang og spissing av intervjuguiden (vedlegg 1) sammen med veileder og NSD hadde godkjent søknaden min (Vedlegg 2) var jeg klar for å begynne med intervjuene med deltakerne jeg hadde funnet fram til. Jeg intervjuet alle fem i løpet av 14 dager, to av informantene som holder til i nærheten av der jeg bor møtte jeg på Kafé og gjorde opptak med telefon m/ekstern mikrofon, mens de tre andre bor lengre unna, så de intervjuet jeg på videokonferanseplattformen Adobe Connect[20] som jeg har tilgang på gjennom HSH. Intervjuene var på mellom 45 og 65 minutter. De tekniske løsningene, både Adobe Connect og mobilopptak, fungerte tilfredsstillende. Jeg var redd på forhånd om at det skulle være plagsom bakgrunnsstøy på kafé, men lyden på

[19] https://www.tekna.no/
[20] http://www.adobe.com/products/adobeconnect.html

opptakene mine er klar og tydelig så jeg hadde ikke behov for å kontakte deltakerne for oppklarende spørsmål i forhold til spørsmålene fra intervjuguiden. Likevel sendte jeg en epost etterpå fordi jeg var nysgjerrig på kjønnsfordeling/-forskjeller, uten at dette var et tema i forhold til forskningsspørsmålene mine.

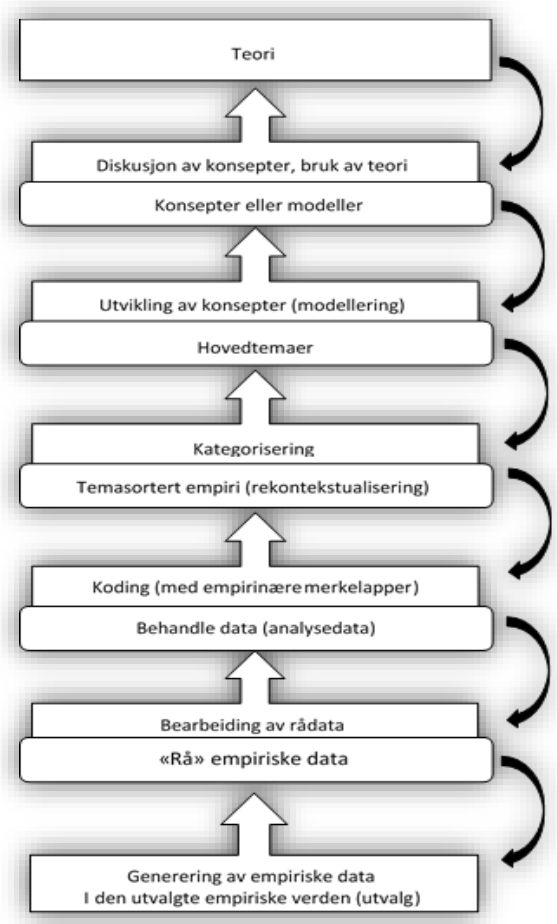
3.3 Transkribering og analyse av intervjuer

«Å transkribere betyr å transformere, skifte fra en form til en annen.» (Kvale & Brinkmann, 2015). Altså er arbeidet med å transkribere intervjuer en prosess hvor du omarbeider muntlig tale til skriftlig tekst, og dette kan være utfordrende. Jeg erfarte gjennom intervjuene at deltakernes beretninger kunne være preget av gjentakelser eller at de begynte å snakke om en ting, fikk en assosiasjon, og begynte å snakke om noe annet. Så det ligger en utfordring i å gjøre om muntlig tale til tekst, det er viktig å være sikker på at man har forstått hva intervjuobjektet faktisk mener om de forskjellige emnene. Målet med transkripsjon er å gjøre de muntlige fortellingene mer egnet for analyse, og struktureringen av den muntlige talen er i seg selv en begynnelse på analysen (Kvale & Brinkmann, 2015).

Intervjuene er altså råmaterialet i undersøkelsen min, og det er nødvendig å arbeide mye med slike tekster for at de skal forstås inn i helheten som denne oppgaven er. Det å analysere et slikt datamateriale er en dynamisk prosess, hvor man går inn og ut av materialet mange ganger, med litt ulike perspektiver hver gang (Postholm, 2010).

Innenfor kvalitativ forskning snakkes det om koding og kategorisering av råmaterialet, intervjuene. Koding i denne sammenheng handler om å gå igjennom materialet og samtidig sette merkelapper på de forskjellige bitene av informasjon, slik at man får dybdekjennskap til materialet sitt, og kan trekke mening ut fra en stor tekstmengde. Slik jeg har jobbet med mine intervjuer, er det nærmest det som kalles *Grounded Theory,* hvor innsamlet data er styrende for hvilke teoretiske tilnærminger man tar til stoffet. Åpen koding (førstefasen i et analysearbeid) handler om å bryte ned materialet sitt, undersøke, sammenligne, konseptualisere og kategorisere data (Kvale & Brinkmann, 2015). Jeg brukte programmet HyperResearch til sette koder på tekstmaterialet. Disse kodene peker enten mot forskningsspørsmål 1 eller 2. Underveis

i arbeidet med oppgaveskriving, har jeg flere ganger vendt tilbake til kodene mine, når jeg har gjort meg nye refleksjoner etter å ha lest teori eller drøftet forskjellig med veileder. Jeg var tidlig bestemt på å bruke stegvis-deduktiv induktiv metode i arbeidet med analysen, fordi jeg mener at det illustrerer hvordan dynamikken i arbeidet bør være: Man har teorien i den ene enden og rådata i den andre, og arbeider dynamisk den ene eller andre veien for å skape mening og forståelse.



Figur 5 Stegvis-deduktiv induktiv metode (Tjora 2010)

Det er viktig å ha tenkt igjennom hvordan materialet skal analyseres både når man planlegger intervjuundersøkelsen og når intervjuene gjennomføres (Kvale & Brinkmann, 2015). Modellen (figur 3) har vært med meg hele veien gjennom forberedelser og gjennomføring av intervjuene. Etter at intervjuene var ferdige, kodet jeg materialet med spesielt fokus på forskningsspørsmålene, slik at informasjonen som ligger i intervjutranskripsjonene er sortert ut fra dette. Kvale og Brinkmann bruker begrepene begrepsstyrt og datastyrt koding. I førstnevnte lager forskeren kodene ut

fra teorigrunnlaget før analysen begynner, mens i datastyrt koding dannes kodene etter hvert som man arbeider seg igjennom transkripsjonene (Kvale & Brinkmann, 2015). Jeg opprettet koder mens jeg arbeidet meg igjennom intervjuene, altså brukte jeg datastyrt tilnærming.

3.4 Validitet og reliabilitet

Validitet i forskningen handler om forskeren undersøker det han/hun hadde til hensikt å undersøke. Dette blir i sin tur spørsmål om troverdighet, bekreftbarhet og overføringsverdi (Krumsvik, 2014). Videre snakker man om ekstern og intern validitet. Den interne validiteten handler om for eksempel sammenheng mellom teoretisk rammeverk og funn. Ekstern validitet er viktig for å vurdere forskningens verdi på tvers av sosiale settinger. Her brukes ofte ordet overførbarhet (Krumsvik, 2014)

Reliabilitet handler om forskningens pålitelighet. En vanlig måte å beskrive det på er å vurdere om en annen forsker, som gjorde nøyaktig de samme undersøkelsene, ville fått nøyaktig de samme resultatene. Dette er litt vanskelig når det gjelder kvalitativ forskning, fordi fortolkningsaspektet er så viktig. For eksempel er det lite trolig at et intervjuobjekt ville sagt nøyaktig det samme første og andre gang han/hun ble intervjuet (Postholm, 2010). Når reliabilitet i kvalitativ forskning vurderes, ser man på tre ting: 1. Reliabiliteten til intervjueren – er det mye ledende spørsmål? 2. Transkripsjonen – er den nøyaktig og formidler den det samme som selve intervjuene? 3. Er analyse- og kategoriseringsarbeidet godt utført? (Krumsvik, 2014).

3.5 Etikk

I mitt arbeid har jeg gjort etiske refleksjoner rundt planleggingen av intervjuene, hvor jeg innhentet informert samtykke til å delta i mitt studie (vedlegg 3) og sørget for å innhente tillatelse fra Norsk senter for forskningsdata, tidligere Norsk samfunnsvitenskapelig datatjeneste, hvor jeg skisserte hvordan jeg tenkte å gjennomføre arbeidet (vedlegg 2). Postholm (2010) skriver at hovedinformantene bør ha så inngående kjennskap som mulig til hva studien handler om, hva de kommer til å bli spurt om, omfanget av oppdraget og hvordan dette skal rapporteres. I informasjonsskrivet jeg sendte til deltakerne gjorde jeg rede for problemstilling og forskningsspørsmål, hvor lang tid intervjuet ville ta, at jeg benyttet lydopptak som skal

slettes ved prosjektets slutt og at de ville få fiktive navn i rapporten. Det er viktig å dele denne informasjonen av flere grunner, men det som for meg var viktigst var at informantene skulle føle seg helt trygge på at personopplysninger og personlige synspunkter ikke ville bli brukt til noe annet enn mitt masterarbeid. Dette er viktig fordi at et gjensidig tillitsforhold er svært viktig for at informantene skal åpne seg og forskeren skal få gode svar på sine spørsmål (Postholm, 2010).

I forhold til selve intervjusituasjonen er de etiske problemstillingene knyttet til at forskeren må være oppmerksom på at relasjonen til intervjuobjektet er viktig. Bevisst eller ubevisst kan forskeren og intervjuobjektet ha forskjellig autoritet og det kan være asymmetrisk slik at informanten ikke føler han/hun kan snakke fritt. Eller man kan få en situasjon hvor intervjuobjektet svarer det de tror forskeren vil høre (Kvale & Brinkmann, 2015). I mine intervjuer var jeg ikke redd for det første, fordi at i mine undersøkelser er det informantene som er ekspertene. Når det gjelder det at informanten sier det forskeren ønsker å høre, var dette viktig å være svært bevisst på for meg, siden jeg i utgangspunktet har tro på at programmering er en fornuftig aktivitet for barn, og kunne ende opp med å være enig med informanten i ett og alt. Det som var mest viktig for meg var å styre intervjuet, slik at jeg fikk så utfyllende svar som mulig på spørsmålene som var nærmest knyttet til forskningsspørsmålene.

4. Funn

Problemstilling og forskningsspørsmål står i fokus for hvordan jeg beskriver funnene mine fra intervjuene. Derfor blir det todelt slik at jeg først skriver om hva jeg fant på forskningsspørsmål 1: Hvilke erfaringer har man fra programmerings-undervisning/kodeklubb? Og 2: Hvilken funksjon kan koding ha for barn i skolealder? Så første del av dette kapittelet vil fokusere på erfaringer, om barns møte med verktøyene, om motivasjon hos voksne og barn, perspektiver på hvordan undervisningen foregår, erfaringer fra forsøk i skolen og noen refleksjoner av praktisk art, fordi dette er hovedområder i datagrunnlaget mitt. I andre del av dette kapitlet ser jeg på hvilken læring mine deltakere mener barna har hatt av å være med på kodeklubb eller undervisning. Her vil fokuset mitt være på konseptene og tilnærmingene fra Barefoot Computing, samtidig som jeg tar med noen begreper som mine informanter bruker om hva barna trenger å kunne om programmering.

4.1 Erfaringer fra kodeklubb/programmeringsundervisning

4.1.1 Praktisk tilrettelegging i kodeklubbene

Pedagogikk (Undervisningsopplegg – program)

Kidsakoder.no har samlet oppgaver innenfor mange former for programmering (Kidsakoder, u.å). Oppgavene er ofte utformet som sjekklister slik at barna kan være selvgående i aktiviteten. Det betyr at det går an å samle barn som jobber på forskjellig nivå i samme rom, som ofte er situasjonen i kodeklubber. De voksne går rundt og løser problemer som oppstår underveis og hjelper til der det trengs.

Organisering og voksentetthet

Veilederne jeg har snakket med forteller om litt forskjellig organisering, og litt forskjellige utfordringer. Karsten og Kristian har begge drevet kodeklubb med tilknytning til universitet/høyskoler. De har hatt få utfordringer med nettverk og maskinpark, mens Knut, som hovedsakelig har arrangert kodeklubb i bibliotekets

lokaler, har måttet løse noen flere tekniske utfordringer underveis. Flere av informantene mine beskriver at barna ofte har arvet en litt eldre PC av foreldre eller storesøsken, og den er til og med ofte full av reklame og forskjellige typer «spamprogrammer,» men tilstand og alder på maskinen er ikke så veldig viktig når det skal jobbes f.eks. i Scratch, men kan være en avgjørende suksessfaktor hvis de skal programmere f.eks. i Unity.

Noe annet som er svært viktig når det kommer til praktisk tilrettelegging er voksentettheten i rommene. Her beskriver kodeklubbveilederne at behovene er veldig forskjellige avhengig av hva det jobbes med, og hvor gamle barna er. Det er vanlig at når man arrangerer kodeklubb for de minste, ber man foreldrene være med. Disse får en assistentfunksjon, hjelper til med å lese opp oppgavetekst osv. De større barna må jo kunne lese oppgavene selv, og de voksne går rundt og hjelper. Mye av aktiviteten i kodeklubben er jo selvgående, som «Kristian» sier: «Når vi har hjulpet dem med innlogging, opprettet bruker på Scratch og delt ut oppgaver, da kan vi i grunnen bare gå.»

4.1.2 Veiledernes motivasjon

De jeg har intervjuet forklarer sin egen interesse og motivasjon for å holde på med dette er at de synes dette er grunnleggende og viktig læring som kan gjøre barna bedre rustet til å fungere i arbeidsmarkedet de skal ut i om noen år. Kodeklubbveilederne jeg har intervjuet har tatt forskjellige slags datastudier og har interesse for å formidle kunnskap om det. En av lærerne har også arbeidet med programmering før han begynte å jobbe i skolen. De er alle av den oppfatning at dette er viktig å lære og ønsker å gjøre en forskjell. En rapporterer at han ønsker han hadde denne muligheten da han var yngre, flere forteller om stor entusiasme og glede hos deltakerne som en viktig motivasjon.

I denne sammenhengen kan man se motivasjon som to forskjellige ting; det første er motivasjonen for å starte opp en slik aktivitet og den andre handler om motivasjonen for å fortsette med det. Alle tre kodeklubbveilederne var først drevet av en interesse for data og teknologi, og syntes konseptet var spennende og ble med av den grunn. Det som har gjort at de har fortsatt med kodeklubb handler om det er morsomt å drive med noe som engasjerer barn/unge og som de mestrer. De jeg har intervjuet,

kodeklubbveiledere og lærere, mener at det er svært viktig å forstå hvordan datamaskiner virker og vite litt om hvordan man programmerer dem for å kunne nyttiggjøre seg digitale verktøy i framtiden.

4.1.3 Antakelser om barnas motivasjon

Informantene i min undersøkelse kunne fortelle mye om hvordan programmering fungerer sammen med barn. De fortalte at de absolutt fleste deltakerne var entusiastiske og motiverte, men noen få mister interessen og faller fra. Motivasjonen til deltakerne handler, ifølge mine informanter, om at det i stor grad er en spilltilnærming på oppleggene som kjøres i kodeklubbene.

> *«Det må jo være gøy. Vi prøver å gjøre det sånn vanskelig gøy. Vanskelig, men gøy, slik at de lærer noe!»*
>
> «Knut»

I intervjuene sier informantene at det som barna først viser interesse for er muligheten for å lage dataspill. Barna ønsker å holde på mer med data og liker dataspill, foreldrene vil gjerne at de skal lære noe de håper skal komme til nytte, så informantene i min undersøkelse mener dette er en viktig grunn til at kodeklubb har blitt så populært. For eksempel forteller både Karsten og Knut har brukt MineCraft og Learn to Mod i et par semestre, og forteller om økt interesse når de tilbyr dette.

Informantene kunne også peke på en del faktorer som er viktige for at barna ønsker å fortsette å kode: Det at man føler mestring er en viktig forutsetning for å holde ut med en aktivitet. Derfor er det svært viktig at barna i kodeklubben får oppgaver som de får til, samtidig som det må være utfordrende. «Kristian» sier at i kodeklubbene han har deltatt i, har de organisert det slik at de er mange veiledere i starten, for å få alle i gang. Foreldre kan også gjerne være med. Dette fører til at barna ganske raskt føler at de mestrer dette, og klarer seg fortere selv.

Siden kodeklubbene er et frivillig tilbud, er det lett for veilederne å anta noe om hva som er motiverende og hva som ikke er det, fordi en del barn/ungdommer slutter fort å komme hvis de ikke synes aktiviteten er morsom. For eksempel er det begrenset

hvor lenge Scratch er interessant for litt større ungdommer, og et naturlig steg videre er å prøve seg på et av de tekstbaserte språkene. For mange er dette demotiverende, plutselig må man skrive ganske mye tekst etter et helt bestemt mønster for å få utført ganske enkle ting på skjermen. Mine informanter er opptatt av den utfordringen, og mener at på et eller annet tidspunkt er en overgang fra det rent blokkbaserte til tekstbasert syntaks nødvendig, men de har ikke noe klart svar på hvordan det burde gjøres og forebygge det at noen mister interessen når de får den utfordringen i tillegg. «Karsten» har brukt Unity en del, med eldre ungdommer, hvor muligheten til å lage spill med en ordentlig spillmotor har virket motiverende for å lære å programmere i C#.

4.1.4 Erfaringer med programmeringsaktivitet i skolen

To av informanter jeg intervjuet er lærere i barneskolen og bruker delvis Kidsakoder sine Scratch-opplegg, i tillegg til oppgaver de lager selv, til forskjellige læringsaktiviteter. Dette gjør de fordi de mener det gir engasjement, motivasjon, læring innenfor problemløsning og samarbeidslæring. De opplever at elevene er svært positive til aktiviteten, også de som ikke var spesielt datainteresserte fra før, og at det er timer hvor elevene er kreative og får brukt fantasien. «Leif» og «Lars» underviser begge mest på mellomtrinnet, og har i samarbeid med skolene de jobber på fått til en time per uke til å ha kodetime. Dette har de gjort gjennom å sette sammen «prosenttimene» i forskjellige fag. Dette kommer av at når man gjør om årstimer til 45/60 minutters økter, blir det en rest på et antall timer. Hvis man da har en rest på 25% matematikk, 25% norsk og 50% andre fag kan det legges opp til en time med et tverrfaglig innhold. Kodetimen begrunnes på ut fra kompetansemål i matematikk, lesing, kunst og håndverk, musikk og digitale ferdigheter, ettersom hvilke aktiviteter det jobbes med.

Lærerne forteller at de har en tilnærming hvor de brukte litt tid på å vise og gjennomgå i introduksjonen av Scratch, slik at elevene hadde en felles, grunnleggende, forståelse for en del av mulighetene som ligger i verktøyet. «Leif» pleier å gjøre det slik at når noen sitter fast, utfordrer han resten av gruppa til å finne løsningen og presentere den for de andre.

Begge informantene som arbeider i skolen, mener at det å bruke programmering i klasserommet er svært positivt, fordi det blant annet er kreativt og engasjerende, elevene lærer å samarbeide, å være problemløsere og de lærer seg mye om hvordan datamaskiner fungerer.

> «Største motivasjonen er vel gleden og entusiasmen og motivasjonen ungene får for dette. Jeg har unger som kommer syke på skolen på torsdagen fordi de nekter å gå glipp av kodetime, og det er ganske ekstremt.»
>
> «Leif»

Elevene ønsker å samarbeide, og de opplever at arbeidet med programmering har gjort at elevene er opptatt av å finne samarbeidspartnere som de fungerer godt sammen med. «Leif» sier: «De har blitt veldig flinke til å finne samarbeidspartnere på sitt eget nivå. Jeg har fått helt nye konstellasjoner av elever jeg aldri hadde tenkt på før. Fordi de trives i hverandres selskap.» Hans erfaring er at elevene har mer fokus på å løse oppgaven, enn på hvem de skal løse den sammen med, og det tyder på at elevene finner oppgavene meningsfulle og motiverende. Av samme årsaker opplever både «Leif» og «Lars» at klassemiljøet blir bedre av at de har kodetime.

«Leif» hadde også en annen observasjon, at spesielt en av elevene hans, en som ikke så ofte gjorde seg bemerket i klassen, blomstret veldig når de hadde kodetime, fordi plutselig var han eksperten. «Lars,» som ifølge seg selv kan mindre om data og dataspill enn elevene, har også erfaringen at det er elever som fort tar ekspertrollen, og kjenner det som positivt at de kan mye om noe de jobber med på skolen.

Alle mine informanter rapporterer om en viktig læringsgevinst av å gjøre f.eks. Scratch-oppgaver: Dette er utfordrende for barn som ikke er nøyaktige og gode lesere. Det er viktig å kunne lese oppskrifter og arbeidsbeskrivelser nøyaktig, og bare små, tilsynelatende ubetydelige, feil i et dataprogram gjør at det ikke virker.

«Man kan knytte det til norskfaget, man må lese instruksjoner og følge instruksjoner. I matematikk lærer de å regne ut hastigheter og sånn. Jeg ser altså dette som veldig tverrfaglig,» sier «Lars.» Kombinasjonen av at motivasjonen er sterk for å lykkes og at de ofte jobber i par gjør at barna konsentrerer seg godt når de koder.

En ting som også kom opp i samtalene med lærerne som programmerer er hva skolen, ledelsen og lærerne, tenker om programmering som fag. For svært mange er det ganske ukjent, og det er utenkelig for svært mange lærere å undervise i programmering.

> «Og jeg tenker på den generasjonen som jeg har i sjuende, hvor er det de holder til? Jo, det er på dataen. Så ting som faller veldig naturlig for dem kan virke veldig skremmende på lærerne.»
>
> «Lars»

«Lars» forteller at det som har vært mest utfordrende for ham er hvordan ledelse og kolleger først møtte initiativet hans om å programmere, men han har etter hvert fått gehør. Nå har skolene i kommunen han jobber, svært høy deltakelse på kodetimen og «Lars» har holdt Scratch-kurs for skoleledere fra hele kommunen.

4.1.5 Veilederrollen

> *«Jeg tror det er nyttig at ungene ser at den som holder kurset også er i en læringsprosess, at vi undrer oss og lærer sammen. Når jeg tenker tilbake på min egen skolegang, husker jeg ingen lærer som var lærende. Han var en ekspert som kunne øse av sin kunnskap.»*
>
> *«Knut»*

Dette sitatet handler om en antakelse om at barn lærer godt av en som ikke alltid har svarene, som kan undre seg sammen med dem. Læreren «Lars» sier noe av det samme: «Det vanskeligste med det å være en god veileder i programmering, er det å erkjenne at elevene faktisk kan mer enn meg på en del ting.»

Hvilken rolle man skal ta som veileder er et tema som alle informantene har drøftet sammen med sine samarbeidspartnere i kodeklubb og skole. Spesielt «Kristian», «Karsten» og «Knut,» som ikke har lærerutdanning og mye erfaring fra skole, har tenkt mye på dette og gjort seg noen interessante observasjoner. Lærerne som jeg har intervjuet opplever større trygghet på graden av instruering og hvordan organisere læringsøkter på en best mulig måte, men det er ikke sikkert at en tradisjonell skoletilnærming er det beste i forhold til dette lærestoffet.

«Kristian» forteller at i kodeklubben han har ledet, har de hatt mye fokus på veilederrollen, og dette ofte er et tema når veiledergruppa har møter. Dette vitner om en bevisst holdning til dette. De mener også at for å lære å programmere, må du programmere mye selv for å mestre det. Så de kjører veldig kort input, eller type «dagens tips» som en igangsetter, også jobber barna selv. Deltakerne jobber med sine prosjekter i sitt tempo, mens veilederne går rundt og hjelper. De er opptatt av at når et av barna rekker opp hånda, skal de hjelpe ved å stille dem noen gode spørsmål framfor å gi dem svaret. «Karsten» forteller om noe av det samme, og beskriver også at han i begynnelsen hadde en tendens til å peke ut feilen, og fortelle hva som burde gjøres. Det er så viktig at barna reflekterer selv, framfor å få servert ferdige svar. «Knut» forteller at det blir en blanding av å gjøre ting sammen med barna, og finne ut av dem selv. Når han har partier med mange nybegynnere, kan det hende at de kjører en oppstart hvor de går trinn for trinn igjennom et prosjekt sammen med barna, men som oftest bestreber han seg på at oppstarten skal være så kort som mulig, og at barna skal tenke og gjøre så mye som mulig selv.

«Lars» og «Knut» har begge gode erfaringer med en type undrende og utforskende tilnærming. «Knut» forteller at han ble interessert i barn og koding ut fra en egen interesse for elektronikk og roboter, fordi han deltok i forskjellige «makerspaces[21]» hvor han og folk i for eksempel USA satt med webkamera og mikrofon og løste utfordringer sammen. «Knut» mener at når den voksne også er lærende og man undrer seg sammen, kan dette skape god læring. «Lars» mener at siden han ikke kan spesielt mye om programmering eller data selv, har noen av elevene blitt ekspertene som har hjulpet både medelever og læreren. Dette har vært viktig for barna som har fått denne rollen, fordi at det gir posisjon i klassen og selvtillit å være den som kan mer enn læreren.

I sitt klasserom opplever «Leif» at veilederrollen varierer med aktiviteten. De første øktene i siste skoleår, brukte han Scratch på den måten at elevene ble gitt en oppgave i form av hva sluttresultatet skulle være, også jobbet de med dette. Hvis noen stod fast utfordret han de andre elevene til å presentere en løsning. «Leif» bestrebet seg på å

[21] http://renovatedlearning.com/2015/04/02/defining-makerspaces-part-1/

bruke åpne spørsmål og en undrende tilnærming når alternative løsninger ble drøftet. Siden har de fulgt noen av kidsakoder sine opplegg, og da blir ofte lærerens rolle å hjelpe barna finne feilen, når ting ikke fungerer. Han understreker hvor viktig det er at barna selv finner feilen, at læreren ikke blir en «feilretter.»

4.2 Hvilken funksjon kan koding ha for barn i skolealder

Informantene i min studie mener at det viktigste barna lærer når de koder er å det som handler om å forstå hva et dataprogram kan gjøre og hvordan finne en løsning på problemer som oppstår underveis. De mener at det viktigste er problemløsnings-kompetanse, kreativitet og det å ha en grunnleggende kjennskap til hvordan datamaskinen virker og hva et dataprogram gjør. Tidligere i denne oppgaven brukte jeg plakaten fra Barefoot Computing om Computational Thinking, hvor man hadde konseptene på ene siden, logikk, algoritmer, dekomposisjon, mønstre, abstraksjon og evaluering på venstre side. På høyre side var det tilnærminger for å lære om og å bruke disse konseptene: fikling, kreativitet, debugging, utholdenhet og samarbeid. Jeg har brukt disse tilnærmingene som underoverskrifter igjennom dette kapittelet.

4.2.1 «Fikling»

Innenfor det som er skrevet om algoritmisk tankegang er dette med prøving, feiling og eksperimentering viktig, og det er interessant å reflektere rundt hvilken plass dette har i kodeklubb/klasserom. Ved første øyekast kan det se ut til oppgavene til både Kidsakoder.no og Code.org ikke prioriterer denne arbeidsmåten fordi mange av oppgavene er trinn for trinn, med lite kreativt spillerom. Så opplever informantene i mine intervjuer at barna som lærer å kode sammen med dem ikke får utforsket og lekt? Det vi snakket om i intervjuene er hvordan trinn-for-trinn oppskriftene hjelper barna til å få en grunnleggende forståelse av hvordan f.eks. Scratch virker, og kan være med å trigge nysgjerrigheten som trengs for å produsere egne programmer. Friheten for mange av barna ligger i å velge andre figurer og bakgrunner enn det som står i oppskriften, og å endre parametere slik at programmet deres oppfører seg annerledes. I oppgavene på kidsakoder er det også mange forslag til videreutvikling og ekstra utfordringer for å forbedre det ferdige produktet.

«Knut» er den som er mest opptatt av dette aspektet i sin kodeklubb. Han er jo også den som beskriver seg selv som nærmest makerbevegelsen når det gjelder filosofi rundt egen og andres læring. Spesielt i det han har gjort i MineCraft i kodeklubben bærer preg av at de eksperimenterer seg fram til løsninger på forskjellige spørsmålstillinger og problemer som dukker opp underveis.

> *«Hvis noen har lyst til å lage en katapult og jeg ikke har laget en katapult før, så må vi finne ut hvordan vi lager en katapult sammen da!»*
>
> *«Knut»*

Kontrasten mellom en tilnærming hvor man prøver og feiler seg fram til en løsning og en tilnærming hvor man analyserer og planlegger kan virke stor. «Leif» mener at struktur og ryddighet er noe man lærer av å programmere, fordi at det å gjøre ting i riktig rekkefølge og hele tiden vite hva de forskjellige bitene med kode gjør er viktig for å få et program til å virke. Det er ikke nødvendigvis et motsetningsforhold mellom det å være opptatt av at godt skrevet kode er ryddig og velorganisert, og mene at prøving og feiling er en viktig inngangsport for mange, men det gjelder å være en god støtte underveis, slik at barna lærer av eksperimenteringen, sånn at de ikke tror et godt resultat kommer på grunn av tilfeldigheter.

4.2.2 Kreativitet

Utgangspunktet i Kidsakoder sine oppgaver er å følge trinn-for-trinn oppskrifter, men likevel holder mine informanter fram nettopp muligheten for å være kreativ og skapende med datamaskinen som et viktig argument for at barn skal lære dette. «Kristian» forklarer dette ut fra at man må kunne et rammeverk før man kan virkelig utforske det, og man må ha en idé om hva f.eks. Scratch kan gjøre før man kan begynne å bygge egne programmer fra grunnen av. Det kreative viser seg i form av variasjoner rundt oppgavene hos nybegynnerne, f.eks. hvilke figurer og bakgrunner som blir brukt osv. Når barna etter hvert forstår hvilke muligheter som finnes, slipper de seg mer løs, og begynner å finne på egne prosjekter.

Det kreative kommer også til uttrykk ved måten barna jobber når de programmerer datamaskiner. Samtalene mellom de som jobber i par handler om ideer og løsninger, og drøfting av hvor gode disse er.

> *«Den store gevinsten ligger i det å lage og tilrettelegge for at unger og ungdom blir problemløsere, ikke sant? Lærer seg å løse problemer på en kreativ måte.»*
>
> *«Knut»*

En annen ting som også handler om det kreative, er hvilke uttrykksformer man kan benytte på en programmeringsplattform. Du kan for eksempel lage bildekunst, interaktive fortellinger og musikalske uttrykk. Mine informanter var alle litt inne på programmeringsverktøy i kunst og håndverk i skolen, at det er et fag hvor dette kan ha en plass. «Leif» sier at en av Ludvigsen-utvalgets styrker er fokuset på kunstfag sammen med det digitale.

Det som kanskje er viktigst med programmeringsaktivitet og barn i forhold til det kreative er det at det primært er en skapende aktivitet. Barna oppdager at det går an å skape noe samtidig som man lærer.

4.2.3 Debugging

Debugging handler ifølge Bers et. al (2014) om på meningsfullt vis gjøre feilsøkingsprosesser eller gå igjennom koden for å gjøre den mer effektiv. Dette er viktige ferdigheter i forhold til å være kompetent i algoritmisk tankegang. I mine intervjuer gjorde jeg ikke disse underpunktene til poenger i seg selv, så debugging var ikke et spesielt tema i intervjuene. Likevel ble det sagt en del om prosessen, når barn programmerer, og debugging er jo en del av denne. «Knut» beskriver at det hender man går hjem med et uløst problem, og det kan være veldig frustrerende. Så, når det har gått litt tid, problemet har kommet litt på avstand, så plutselig forstår man. Det er lett for den voksne å la seg rive med når barnet er frustrert og fortvilet over at ting ikke virker, og være rask med å peke ut det som er feil. «Kristian» sier noe av det samme, men at man bestreber seg etter å stille ledende spørsmål for at barnet selv skal finne ut av hva som er feil.

«Kristian» bruker også begrepet *Rubber Duck*[22] som er et begrep innenfor programmering. Tanken er at programmereren skal lese opp koden sin til en utenforstående, og raskere forstå hvor det blir feil. Dette er en kjent debuggingsstrategi, og det viktigste er at den som inntar rollen som rubber duck ikke avbryter eller sier for mye.

> «Mm, det å si ting høyt. Det er jo et uttrykk som kalles for Rubber Duck. Du trenger ikke få svar, bare det at du sier det høyt til noen gjør at du skjønner hva du gjør feil, da. Og det kan være nok."
>
> "Kristian"

Altså er det viktig at de voksne som bistår barn i koding ikke hjelper de for mye. De må lære å søke etter feil og rette de opp selv, det er en meget viktig bit av det å programmere. «Leif» er opptatt av å holde fokus på at barna skal finne feilen selv: «Ikke fortelle dem hva de har gjort feil, men gjerne hinte dem dit hen at de klarer å finne det selv.»

4.2.4 Utholdenhet

«Lars» er den av mine informanter som nevnte utholdenhet som egenskap som kan fremme læring. Han mener at det å programmere i klasserommet kan være med å trene utholdenhet, som er svært viktig for å lykkes i skolen. Grunnen til koding er en god aktivitet for å trene dette, sier «Lars», er fordi at du har spillelementet, det skapende og utforskende som kan gjøre at elever orker litt mer med koding enn med for eksempel matematikkoppgaver.

4.2.5 Samarbeid og delingskultur»

Delingsaspektet er blitt en viktig del av vår digitale virkelighet, og plattformene vi nå bruker både til arbeid/skoleformål og privat har samarbeids- og delefunksjoner. Dette er også sentralt i et konstruksjonistisk læringssyn. At læring best skjer når man lager noe selv og viser det fram for andre. Deling er også viktig i kodeklubbene, at man får tid til å vise fram og demonstrere det man har laget, samarbeide om å lage et program

[22] https://en.wikipedia.org/wiki/Rubber_duck_debugging

og gjerne også deling på sosiale plattformer. Scratch har egen delefunksjon hvor du kan både se på, og videreutvikle andres prosjekter og legge ut dine egne.

> *«De har lyst å lage et spill alle kan spille, de har lyst til å lage en app som de kan vise på mobilen, eller som andre kan ha på mobilen. De har lyst til å lage noe konkret, som de kan dele med andre å si se hva jeg har laget!»*
>
> *«Kristian.»*

Mine informanter var svært opptatt av delingskultur og samarbeid. Spesielt «Leif» snakket mye om det at når de hadde kodetime, søkte elevene samarbeidspartnere som var spesielt nyttige i forhold til oppgaven som skulle løses, slik at en del vanlige «bestevenner» som søkte hverandre bare fordi de var venner gjorde det mindre enn vanlig. «Kristian» henviser til at Ludvigsen-utvalget og 21st century skills, alle legger stor vekt på det å kunne jobbe i team, å løse problemer sammen. Dermed blir parprogrammering en metode han har benyttet mye i kodeklubbene.

Konstruksjonismen er jo opptatt av at gode læringsopplevelser skjer når man konstruerer et produkt og deler/viser det til andre. Delingsaspektet er viktig i kodeklubb og f.eks. Scratch-bevegelsen. Alle mine informanter snakker om hvor viktig det er for mange av barna å vise fram og få anerkjennelse for det de har kodet.

5. Drøfting.

I dette kapittelet vil jeg drøfte funnene mine. Drøftingen tar utgangspunkt i problemstillingen og forskningsspørsmålene mine.

Problemstilling:

Hvordan kan erfaringer fra kodeklubbkonseptet og forsøk med programmering i klasserommet være med å utvikle elevenes programmeringskompetanse i skolen?

Forskningsspørsmål

1. Hvilke erfaringer har voksne som driver med programmeringsundervisning og kodeklubb?
2. Hvilken funksjon kan koding ha for barn i skolealder?

Drøftingen skal se på hva tidligere erfaringer, både gjort av informantene i mine undersøkelser og tidligere forskning, kan tilføre skolen, spesielt med en tanke om at det virker sannsynlig at vi får mer programmering inn i skolen.

5.1 Erfaringene fra kodeklubb og koding i skolen

Jeg har sett på hvordan voksne som har arrangert programmeringsaktivitet på kodeklubb og skole har tenkt rundt og organisert dette. Drøftingen min vil omhandle refleksjoner og erfaringer som ble formidlet til meg i intervjuene. Dette vil jeg se på i forhold til relevant forskning og teori presentert i denne oppgaven.

Organisering

Kodeklubbene er drevet av frivillige som har forskjellig bakgrunn. Det som undervises er ofte basert på en sjekkliste hvor det er lett for barnet, såfremt det klarer å lese godt nok, å følge progresjonen å lage et produkt. Dette passer godt i kodeklubbene hvor barn med forskjellig alder, ferdigheter og interesser ofte sitter på samme rom og arbeider. I skolen er det også forskjeller, naturligvis, men barna er på samme alder, læreren har god oversikt over deres styrker og svakheter og de har ofte fungert sammen over tid. Derfor er det muligheter for å organisere seg på en annen måte i skolen.

Erfaringene fra kodeklubbveilederne jeg intervjuet er at det er fint med mange voksne i starten av et prosjekt, også blir barna mer selvgående når de har lært seg det grunnleggende med verktøyene (4.1.1). En skole kan for eksempel la en assistentressurs gå på omgang slik at klassene legge introduksjon til nye programmeringselementer til litt forskjellig tid på året med litt flere voksne i rommet. En annen måte man kan få litt flere voksne inn i perioder er at skolen kan bruke ressurser til delingstimer til å sørge for at læreren som skal lære elevene å programmere enten får mindre grupper i perioder eller får to-lærer.

Motiverte og kvalifiserte voksne

Mange som har utdanning og erfaring med programmering er opptatt av at barn og unge har muligheten til å få grunnleggende opplæring i dette. Det er en mulighet at skoler inviterer veiledere fra kodeklubbene til å lære opp lærere eller arrangere kurs for elever i sammen med lærere, slik det ble organisert i England før Computing-faget for alvor ble innført (N. Smith et al., 2014). Det er mange motiverte og kvalifiserte voksne som er aktive i kodeklubbene, som ser behovet for at barna lærer å programmere (4.1.2).

Motivasjon og programmeringsundervisning

Programmeringsaktivitet er engasjerende og motiverende for mange barn/unge. De liker å skape med datamaskinen, det å kunne lage spill er spennende, det å vise til andre/dele er motiverende og det å arbeide med problemløsning (og lykkes) er motiverende. Malone og Lepper (sitert i Mozelius, 2014) hevder disse fire begrepene er viktige i Game-Based Learning når man skal beskrive indre motivasjon: Utfordring, nysgjerrighet, kontroll og fantasi (min oversettelse.) Dette stemmer godt overens med det som kom fram i mine intervjuer (4.1.3).

Dette tyder på at programmeringsaktivitet er lystbetont for deltakerne, fordi de opplever noe av det samme som de opplever når de spiller, de samarbeider med andre, gjerne på samme nivå som seg selv og de møter utfordringer.

Erfaringer med programmeringsundervisning

Både hos kidsakoder.no og code.org er prinsippet at barnet til å begynne med lærer å kode ved å følge trinn-for-trinn oppskrifter, og gradvis ta i bruk mer avanserte konsepter i programmene sine. Dette kan oppfattes som et paradoks, siden Seymour Paperts betydning så ofte trekkes fram og hans konstruksjonisme forbindes med at barnet engasjeres i egen læring gjennom å være den som bygger opp noe i et miljø hvor de lærende deler og snakker om prosessen de er inne i og hjelper hverandre videre (Papert & Harel, 1991).

Argumentene for at barn som skal lære å programmere med trinnvise oppskrifter på hva de skal gjøre er at de må få et repertoar før de kan begynne å utforske verktøyene på egenhånd. Ut fra en tanke om at mestring er grunnleggende viktig i møtet med nytt lærestoff, er det lurt å legge opp til barna lærer trinn for trinn. Selv om Scratch på mange måter fremstår som svært enkelt, vil det til å begynne med være uoversiktlig og vanskelig å forstå for nybegynneren.

> *«…og det ser vi veldig tydelig at hvis du har vært innom en tre-fire økter med det, på en måte gjort oppgavesett som er gradvis litt vanskeligere så ser de hvilke muligheter som finnes. Og da ser vi at 3, 4 eller 5. kvelden sier barna: "Åh! Kan vi ikke lage vårt eget spill" eller "jeg vil lage noe, men jeg vet ikke helt hva!" Så kan vi begynne å drodle litt og ender vi opp med et eller annet kjempekult. Det er ofte sånn alle steder hvor du skal lære noe, tenker jeg, du kan ikke bare si «vær kreativ!» <u>Men hvis du har litt bakgrunn og er kjent med det verktøyet du skal bruke, da kan du begynne å utforske.»</u>*
>
> *«Kristian»*

Når barna har lært det grunnleggende og skal prøve på egenhånd er det mer utfordrende for veilederen eller læreren, fordi han/hun må gjøre vurderinger rundt hvor mye hjelp som skal gis og om hva barnet må finne ut selv (4.1.4). Dette må vurderes både fra et programmeringsfaglig og pedagogisk ståsted; har den lærende kunnskapen/ferdighetene som skal til for å finne ut av dette på egenhånd? Er motivasjonen sterk nok til å overvinne dette problemet, eller vil barnet gi opp? Det er

en fare for at den voksne kan være for snar til å ordne opp, slik at barnet kommer videre uten å gjøre seg disse refleksjonene.

Å inkludere programmering i skolen, med fag- og timefordeling slik det er nå er fullt mulig. Det går an å trekke ut tid fra andre fag, siden årstimene ikke «går opp» når disse gjøres om til 45 minutters økter, for eksempel, og gi timene tverrfaglige mål knyttet til lesing, matematikk og praktisk-estetiske fag (4.1.4). En annen mulighet er å inkludere programmering med fagmål i for eksempel matematikk eller naturfag, slik som blant annet en del norske masteroppgaver skisserer (Haaland & Rosvold, 2006; Iversen, 2015; Jåtten, 2006; Lang-Ree, 2016).

Veilederrollen

De jeg har intervjuet har litt forskjellige erfaringer med hva de mener gir god programmeringsundervisning. Det er en fare for at noen av beskrivelsene jeg har fått av de gode læringsopplevelsene er litt spekulative, at erfaringene som blir beskrevet er påvirket av subjektive oppfatninger hos de jeg har intervjuet. «Knut» beskrev for eksempel en oppfatning om at en helt åpen og undrende tilnærming fungerte godt for ham i kodeklubben, men det at han synes det fungerer for ham, betyr ikke at dette vil fungere for alle. Det som er som er viktig, kanskje spesielt for de uten pedagogisk utdanning og erfaring med skole, er å reflektere rundt spørsmål som har med veilederrollen å gjøre sammen med de andre de samarbeider med og det forteller informantene i min undersøkelse er et tema som ofte blir tatt opp på veiledernes møter (4.1.6).

> *«Det å la barnet finne ut av ting selv er noe som jeg har blitt flinkere til etter hvert. I starten tror jeg at jeg veldig fort, når de satte seg fast, viste jeg dem bare hva de måtte gjøre. Men nå etterhvert har jeg blitt flinkere til å få dem til å finne ut av det selv. Kanskje hjelpe dem litt på veien. Selv om det tar litt lengre tid, så lærer de noe av det. Også prøver jeg jo å være behjelpelig. Så godt jeg kan, prøver å ikke si for mye, ikke avsløre alt…»*
>
> *«Karsten»*

Kodeklubbveilederne er opptatt av hvordan barna best lærer å programmere, og har mange erfaringer fra da de lærte dette selv. Men det å lære bort og å vite hvilken hjelp det enkelte barn trenger, er erfaringer de har gjort seg i møtet med barn. Dette er en større del av kompetansen lærere har fra sin utdanning og praksis. Lærerne jeg har intervjuet og kodeklubbveilederne gjør litt forskjellig,

5.2 Hva kan barn lære av å drive med koding?

Før jeg begynte arbeidet mitt så jeg for meg at informantene ville gi meg informasjon om konkrete programmeringsoperasjoner de mente barn/fremtidens arbeidstakere trengte å lære. Men gjennom intervjuene mine fikk isteden jeg mange innspill innenfor algoritmisk tankegang, kreativitet og forståelse av hvordan datamaskiner virker.

> *«Jeg mener at konseptene er viktigere enn selve kodingen. Forståelsen av hvordan ting fungerer og skjønne den logiske strukturen og alt det der.»*
>
> *«Kristian»*

Algoritmisk tankegang

Algoritmisk tankegang handler om en tilnærming til å løse problemer og finne ut av ting, en måte å tenke på. Det står i læreplanen for det nye valgfaget i programmering at algoritmisk tankegang er evne til å identifisere problemer og utforme løsninger, lage kode som kan forstås av en datamaskin og systematisk feilsøke og forbedre koden (UDIR, 2016). Det man kan spørre seg er; utvikler barn som leker med småspill i grafiske grensesnitt i klasserom eller kodeklubb spesielle evner i å angripe problemer og utfordringer på en analytisk og løsningsorientert måte? I større grad enn barn som ikke lærer å kode? Det barna som lærer å kode fort forstår, er at dataprogrammer består av mange små enheter (skript) som jobber sammen og det å kunne bryte ned et problem i mindre deler blir noe de fort blir i stand til. Jeanette Wing (2006) brukte begrepet at algoritmisk tenkning handler om å tenke som en som mestrer informatikk og det handler om hvordan mennesker kan lære og løse problemer med tilnærminger fra datamaskinens verden.

«Algoritmisk tankegang da, om du vil. Det er kanskje det viktigste du sitter igjen med. Det er de store linjene. Hvordan du tenker og bygger opp ting, og bryter ned. Å løse små problemer og sette sammen.»

«Kristian»

Mange barn som går i kodeklubb eller lærer programmering på skolen kommer til et punkt hvor de får lyst til å gå i gang med mer avanserte ting, som for eksempel tekstbaserte språk, eller å lage programmer fra grunnen av. Slike barn/unge tilegner seg mange konsepter, både innenfor programmering i seg selv og algoritmisk tankegang. Men det er også barn som er tilfredse med å følge oppskrifter og gå i samme sporet. Jeg mener at veilederen eller læreren må være oppmerksom på dette, og gi nye utfordringer når det er lite progresjon.

Prøving og feiling (tinkering)

Dette er en måte å angripe problemløsning på. Resnick og Rosenbaum (2013) skriver at vi lever i en tid hvor det å skape noe, *maker-bevegelsen,* har fått vind i seilene. Dette har både teknologiske og kulturelle årsaker. Selv om ikke maker-bevegelsen har noe direkte bånd til skole og utdanning, har Paperts konstruksjonisme og Deweys progressivisme, som oppmuntrer til prosjektbasert og utforskende læring, hatt innflytelse på tankesettet til mange som er opptatt av bygge og lage ting (Resnick & Rosenbaum, 2013). Kodeklubbene er beslektet med maker-bevegelsen, og prøving og feiling er en problemløsningmetode som ofte nevnes, både i forhold til algoritmisk tankegang og som en problemløsningsstrategi i seg selv. Både programmerere og lærere kan være skeptiske til det å prøve seg fram fordi at det krever lite planlegging og struktur. Når barn/unge programmerer bør det være rom for å prøve og feile, fordi det er en arbeidsmåte som passer mange barn (4.2.1).

«…tilrettelegge for at barn og ungdom blir problemløsere, ikke sant. Å lære seg å løse problemer på en kreativ måte, lære å feile og lære av sine feil. Det er kanskje mindre fokus på faktakunnskaper og sånne ting, for de finnes jo der ute. Just-in-time læring, da.»

«Knut,» om læring i kodeklubben.

Kreativitet

Det er rom for kreativitet i programmeringsundervisning, og det er også en viktig grunn til å bruke det i skolen. For eksempel Scratch gir muligheter til å lage interaktive kunstverk i kunst og håndverk, komponere i musikk og å lage tegnefilm eller bildefortellinger i norsk (4.2.2). Men programmering gir også muligheter for kreativitet også i andre fag. Det er stor enighet om at kreativitet og innovasjon er kompetanser som kommer til å bli etterspurt i framtida (Binkley et al., 2012; Dede, 2010; P21.org, 2015) og programmeringsverktøyene som er tilgjengelig egner seg godt for å styrke disse ferdighetene hos barna.

> «For samfunnet blir jo stadig mer preget av det, det kommer sensorer over alt. Roboter over alt. Det å kunne være kreativ med teknologi, da stiller de mye sterkere i det som kommer til å bli deres voksenliv, da»
>
> «Knut»

Debugging

I mine intervjuer fant jeg at informantene ikke er spesielt opptatt av å lære barna debugging. Det var et tema da vi snakket om veilederrollen, hvordan de var opptatt av at barna skulle finne feilen selv (4.2.3). Siden dette er regnet som en viktig kompetanse innenfor algoritmisk tankegang og noe de som jobber som programmere er godt kjent med, er dette viktig å inkludere i programmeringsundervisning av skolebarn eller i kodeklubb i større grad. Evnene til å finne feil og rette dem opp, blir gradvis bedre jo mer man programmerer, likevel, hvis man skal ha inn et teknologifag hvor programmering utgjør en viktig del i skolen, må dette defineres som en kompetanse barna må få opplæring i.

Utholdenhet

Det å holde ut over tid med en arbeidsoppgave, det å ikke gi opp når noe er vanskelig, det er viktig kompetanse når det jobbes med problemløsningsaktivitet. Barnet må investere i tid for å gjøre framskritt, spesielt når det arbeides med oppgaver som krever dyp tenkning. Carrol (1989) omtalte utholdenheten som tiden den lærende var villig til å bruke på lærestoffet, og videre at dette var en funksjonell indikator på hvor

motivert barnet egentlig er i å tilegne seg det som skal læres. Det var bare en av informantene i min undersøkelse som nevnte dette, læreren «Lars» som beskriver at programmering gir en god arena for å trene på dette, fordi det er såpass motiverende for mange, samtidig som det er utfordrende når du blir stående fast (4.2.4).

Siden utholdenhet også er en del av algoritmisk tenkning, som er grunnleggende kompetanse i mye teori og læreplaner, bør dette også være definert i fag hvor det skal programmeres i skolen.

Samarbeid og parprogrammering

Mye av forskning om barn som programmerer og 21st Century Skills har en tydelig fellesnevner; samarbeid og samhandling er viktig for å lykkes. Dette er også et funn som går igjen i mine undersøkelser, når barn programmerer kjennetegnes prosessen av at de prater sammen, leter etter løsninger sammen og søker ofte en partner de samarbeider godt sammen med, som ikke nødvendigvis er bestevennen (4.2.5). Dette er litt forskjellig fra hvordan lærere ofte opplever gruppearbeid og arbeid i par i skolen. Da er elevene ofte mest opptatt av hvem de jobber sammen med, og gruppearbeid betyr å dele opp et større arbeid i mindre deler. *«Kristian» sier: «Men det å jobbe med problemer sammen, tenke sammen og diskutere ting og komme fram til løsninger, den måten å jobbe på, det gjør vi mye i kodeklubben.»* Dette tyder på at det å drive med programmering som læringsaktivitet kan være med å styrke samarbeid og relasjoner i en gruppe barn. Samtidig, så er det forskjell på hvordan en barnegruppe fungerer i en fritidssetting, eller i kodetimer på skolen, og i aktiviteter som er mer dagligdagse og rutinepregede. Det er ikke sikkert at slike faktorer vedvarer hvis koding blir et fag i skolen på linje med norsk og matematikk.

Parprogrammering beskriver en metode som er brukt av profesjonelle programmerere for å effektivisere og forbedre prosessen. Det handler om at en har et overordnet lederansvar og tar beslutningene, mens den andre betjener mus og tastatur og får jobben gjort.

Konstruksjonismen er opptatt av at gode læringsopplevelser skjer når man konstruerer et produkt og deler/viser det til andre. Delingsaspektet er viktig i kodeklubb og i f.eks.

Scratch-bevegelsen. Alle mine informanter snakker om hvor viktig det er for mange av barna å vise fram og få anerkjennelse for det de har kodet.

6. Avslutning

6.1 Konklusjon

Målet med denne studien var å undersøke hvordan programmering i kodeklubb og skole er organisert, og hva barna lærer. Jeg ønsket også å finne ut hvordan disse erfaringene kan komme skolen til gode, spesielt hvis programmering kommer inn i skolen i Norge slik som i mange andre land.

Problemstilling:

Hvordan kan erfaringer fra kodeklubbkonseptet og forsøk med programmering i klasserommet være med å utvikle elevenes programmeringskompetanse i skolen?

Forskningsspørsmål 1: Hvilke erfaringer har voksne som driver med programmeringsundervisning og kodeklubb?

I denne kvalitative og fenomenologiske undersøkelsen har jeg brukt intervju som metode for å finne svar på mine forskningsspørsmål. Mine teoretiske briller i gjennom undersøkelsen var først og fremst algoritmisk tankegang og konstruksjonisme. Mine funn er at barn som lærer å programmere i kodeklubb eller på skolen motiveres av muligheten for å lage spill og arbeide med datamaskin, og de uttrykker glede og entusiasme når de holder på dette. I forhold til det første forskningsspørsmålet var det teoretiske perspektivet knyttet til motivasjonsteori. Ellers er intervjuene den klart viktigste kilden i forhold til erfaringene.

Når det gjelder det praktiske i kodeklubb, har ofte barnegruppene aldersmessig spredning, og deltakerne er på litt forskjellig nivå eller til og med holder på med forskjellige verktøy og oppgaver i samme rom. Derfor er oppgavene som kidsakoder.no benytter i stor grad bruksanvisninger, som er relativt enkelt å følge for barn som kan lese. Informantene i min undersøkelse som er lærere som bruker programmering i

vanlige skoleklasser opplever at elevene er kreative og engasjerte, de gleder seg til kodetimene og samarbeider godt når de holder på med aktiviteten.

En annen erfaring kodeklubbveilederne har, er at det er en fordel å kunne være flere voksne i oppstarten av et prosjekt, også ville dette behovet avta etter hvert som barna ble trygge på det de akkurat har lært. I Storbritannia var kodeklubborganiseringen slik, før det ble introdusert et eget skolefag med programmering, at en frivillig med teknologibakgrunn slo seg sammen med en lærer og organiserte kodeklubb på skolene, etter skoletid (N. Smith et al., 2014). Man kan tenke seg at Norge nå begynner å få et godt utbygget nettverk med kodeklubb og mange frivillige er involvert. Det er sikkert gode muligheter for at disse kan samarbeide med lærere som ønsker koding inn mer inn i klassen sin, eller hvis dette blir et eget fag eller i større grad tverrfaglig innhold i skolen.

Det er altså mye som tyder på at programmering får større plass i skolen i løpet av noen år enn det har nå. Både stortingsmelding 28 «Fag - fordypning – forståelse» (Kunnskapsdepartementet, 2016) og Ludvigsen-rapporten (NOU 2015:8, 2015) signaliserer tydelig at man tenker en endring i skolen, også rundt hva elevene skal lære på datamaskinen. Utdanningsdirektoratet bestilte en ekspertrapport for å ha et kunnskapsgrunnlag om hvilke kompetanser norske skoleelever skal ha innenfor teknologiemnene, og denne går inn for at teknologi og programmering skal utgjøre et obligatorisk fag i skolen (Sanne et al., 2016).

Forskjeller på skole og kodeklubb er at en skoleklasse er mer homogen enn en gruppe i kodeklubb på en måte. Det vil si at barna er like i alder og som oftest også ganske like i utvikling. I kodeklubben har du større aldersspredning og forskjeller i hvor avanserte ting deltakerne jobber med. Likevel har de interessen for å drive med data og å lage spill eller andre programmer til felles. Lærerne som jeg intervjuet i forbindelse med dette arbeidet hevder at engasjementet og interessen er stor også i en vanlig skoleklasse, slik at det er sannsynlig at også dette vil være motiverende og spennende for de fleste elever.

Forskningsspørsmål 2:

Hvilken funksjon kan koding ha for barn i skolealder?

I denne oppgaven har jeg presentert noen av de punktene som går igjen når kravene til neste generasjons jobbsøker blir presentert: Blant annet gode samarbeidsevner, god omstillingsevne (har god kompetanse på å lære nye ferdigheter), teknologiske ferdigheter, kreativitet og evne til problemløsning (Dede, 2010; P21.org, 2015).

Om algoritmisk tankegang

Noe som barn kan lære av programmering, er å bli gode problemløsere. Det å prøve å feile for å finne løsninger på utfordringer de møter når de prøver å få et program til å virke. Det er en fare for at prøving og feiling, istedenfor å fungere i den spiralmodellen Resnick (2007) skisserer, blir en tilfeldig gjettelek. Derfor er det viktig at de voksne som underviser barn i programmering stiller de rette spørsmålene og reflekterer sammen med dem for at de skal ha læringsutbytte av utprøvingen.

Både i mine funn fra intervjuundersøkelsen, i teori og tidligere forskning framholdes kreativitet som en viktig egenskap hos det programmerende barn. Miljøet som har utviklet Scratch (Lifelong Kindergarten[23]) fremmer et syn om at det lekne og kreative er veldig sentralt i barns læring. Det er viktig at dette blir et sentralt punkt hvis en læreplan i et programmerings- og teknologifag skal utarbeides.

Mye teori og forskning peker i retning av at programmering læres best når barnet aktivt arbeider med problemløsningsoppgaver eller åpne oppgaver i samarbeid med andre (Kafai et al., 2014; Papert, 1980). I kodeklubbene er oppgavene trinn-for-trinn oppskrifter, og barna kjenner ikke på denne dimensjonen før de har holdt på en stund. For å introdusere programmering i en løst sammensatt gruppe med barn med veldig forskjellig utgangspunkt og erfaringer, er det sannsynligvis nødvendig å gjøre det på denne måten. Også skolen kan med fordel ta i bruk slike oppskrifter når de enkelte programmeringsverktøyene og konseptene introduseres, men der har man også mulighet til å jobbe med prosjekter og problemløsning på en annen måte, siden gruppa er mer homogen og stabil.

Samarbeid og delingskultur er ord jeg har brukt mye i denne oppgaven, og både lærerne som bruker programmering i sine timer og i kodeklubbene jeg har fått innsyn i

[23] https://llk.media.mit.edu/

er samarbeid og parprogrammering viktig. Det som ble sagt i intervjuene var blant annet at barna fant seg samarbeidspartner ut fra nivå og interesser, de drøfter underveis og beskriver det de får til eller ikke får til og de er glad i å dele og vise fram det de har laget. Dette er viktig å få til i skolen, i mange fag, men er meget viktig når det er programmering som står på timeplanen.

6.2 Avgrensninger

Et masterarbeid har begrensninger, jeg har hatt et lite utvalg med informanter som har delt sine erfaringer og synspunkter i forhold til min problemstilling og forskningsspørsmål. Deres erfaringer, i kodeklubb og klasserom, er ikke direkte overførbare til andre kodeklubber og klasserom. Det er vanskelig å generalisere direkte, fordi at utvalget i min undersøkelse er lite og at de jeg har intervjuet ikke nødvendigvis er representative for alle som er involvert i programmeringsaktivitet med barn. Beskrivelser av forskningsfeltet eller fenomenet som undersøkes legger til rette for det som kalles en naturalistisk generalisering (Postholm, 2010). Det handler om at den som leser forskningen kan trekke paralleller til sin praksis og få innsikt gjennom dette.

Det forskningen min ikke gir svar på er barnas opplevelser, dette er bare beskrevet gjennom antakelser de voksne har gjort, eller igjennom bøker og artikler skrevet av andre. Jeg har heller ikke noen nøyaktig oversikt over alder og kjønn på deltakere i kodeklubbene jeg har undersøkt, om det er noen ukjente motivasjonsfaktorer som for eksempel press hjemmefra eller at bestevennen vil gå der.

6.3 Veien videre

Det er flere ting som kan være interessant å undersøke i forhold til kodeklubb. Det første som jeg ville undersøke, hvis jeg skulle arbeide videre med dette feltet, er å se på aktiviteten med hjelp av observasjon og intervjuer av deltakerne. Både barnas interesser og motivasjon og områder innenfor programmeringskonsepter og algoritmisk tankegang som med fordel kan undersøkes på den måten.

Kjønnsfordelingen i teknologiyrker er skjev, i kodeklubbene er prosentandelen jenter variabel. Jeg gjorde en kjapp forespørsel til de tre kodeklubbveilederne jeg har intervjuet og de opplevde en overvekt av gutter, men de jobber aktivt for å rekruttere jenter og forteller at det nytter. Det er mange interessante kjønnsperspektiver som kan undersøkes, det påstås for eksempel at jenter er flere sammen hvis de begynner i kodeklubb og at jentene ikke bryr seg om å lære de avanserte konseptene så lenge de får til det de ønsker med det de kan. Hvis dette er tilfelle, hvilke tiltak bør iverksettes?

7. Kilder og referanseliste

Bakken, A. (2015). *Ungdata: nasjonale resultater 2014* Norsk institutt for forskning om oppvekst, velferd og aldring.

Bers, M. U., Flannery, L., Kazakoff, E. R. & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145-157.

Binkley, M.,Erstad, O.,Herman, J.,Raizen, S.,Ripley, M.,Miller-Ricci, M. & Rumble, M. (2012). Defining Twenty-First Century Skills. *Assessment and teaching of 21st century skills* (s. 17-66) Springer.

Bitautas, J. (2013). The Differences Between Programmers and Coders. 04/18, 2016,Hentet fra http://workfunc.com/differences-between-programmers-and-coders/

Bloom, B. S. (1968). Learning for Mastery. Instruction and Curriculum. Regional Education Laboratory for the Carolinas and Virginia, Topical Papers and Reprints, Number 1. *Evaluation Comment, 1*(2), n2.

Bottou, L. (2014). From machine learning to machine reasoning. *Machine Learning, 94*(2), 133-149.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada,* 1-25.

Brown, N. C., Sentance, S., Crick, T. & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE), 14*(2), 9.

Cadwalladr, C. (2014). Are the robots about to rise? Google's new director of engineering thinks so…. 11/07, 2016,Hentet fra https://www.theguardian.com/technology/2014/feb/22/robots-google-ray-kurzweil-terminator-singularity-artificial-intelligence

Carroll, J. B. (1989). The Carroll model a 25-year retrospective and prospective view. *Educational Researcher, 18*(1), 26-31.

CAS. (2014). What is Barefoot? 21/04, 2016,Hentet fra http://barefootcas.org.uk/about/

Catterall, J. (2013). Getting Real about the E in STEAM. *The STEAM Journal, 1*(1), 6.

Ceder, V. & Yergler, N. (2003). Teaching Programming with Python and PyGame. *Apresentado Na PyCon,*

Dede, C. (2010). Comparing frameworks for 21st century skills. *21st Century Skills: Rethinking how Students Learn, 20*, 51-76.

Department for Education. (2013). National curriculum in England: computing programmes of study. 04/18, 2016,Hentet fra https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

Department of Education. (2013). National curriculum in England: computing programmes of study. 08/09, 2016,Hentet fra https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

EUN. (2014). Computing our future – Priorities, school curricula and initiatives across Europe. 11/9, 2015,Hentet fra http://www.eun.org/c/document_library/get_file?uuid=521cb928-6ec4-4a86-b522-9d8fd5cf60ce&groupId=43887

Fields, D. A., Giang, M., & Kafai, Y. (2014). Programming in the wild: Trends in youth computational participation in the online scratch community. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education,* 2-11.

Furber, S. (2012). Shut down or restart? The way forward for computing in UK schools. *The Royal Society, London,*

Giannakos, M. N., Jaccheri, L., & Proto, R. (2013). Teaching computer science to young children through creativity: Lessons learned from the case of norway. *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research,* 103-111.

Globalis.no. (u.å). Norge. 08/05, 2016,Hentet fra http://www.globalis.no/Land/Norge/(show)/indicators

Gove, M. (2012). Michael Gove Speech at the BETT Show 2012. 08/09, 2016,Hentet fra https://www.gov.uk/government/speeches/michael-gove-speech-at-the-bett-show-2012

Grepperud, G. (2012). I Skrøvset S. (Red.). *Undervisningslære : eksempler, ideer og refleksjoner*. Oslo: Gyldendal akademisk.

Haaland, S. & Rosvold, J. S. (2006). *Bruk av Lego Mindstorms i natur- og miljøfag på 9. trinn*. Stord: Høgskolen Stord/Haugesund.

Haigh, G. (2016). Why the government's computer science strategy is completely wrong. 08/25, 2016,Hentet fra http://schoolsweek.co.uk/bring-back-core-digital-skills/

Hatlevik, O. E., Egeberg, G., Guðmundsdóttir, G. B., Loftsgarden, M. & Loi, M. (2013). *Monitor skole 2013 - om digital kompetanse og erfaringer med bruk av IKT i skole.* (No. 6). Oslo: Iktsenteret.

HITSA. (2015). ProgeTiger programme 2015-2017. 08/09, 2016,Hentet fra http://media.voog.com/0000/0034/3577/files/Programm%20ProgeTiiger%202015_2017eng.pdf

Imsen, G. (2014). *Elevens verden : innføring i pedagogisk psykologi* (5. utg. utg.). . Oslo: Universitetsforl.

Iversen, S. (2015). Koding som digital grublis. 06/28, 2016,Hentet fra http://munin.uit.no/bitstream/handle/10037/8089/thesis.pdf?sequence=2&isAllowed=y

Jåtten, E. (2006). First Lego League og motivasjon for realfag. 3/20, 2015,Hentet fra http://hdl.handle.net/11250/152237

Jonassen, D. H. & Reeves, T. C. (1996). Learning with Technology: Using Computers as Cognitive Tools. I D. H. Jonassen (Red.). ***Handbook of research for educational communications and technology)*** ((1st utg).). Bloomington IN: The Association for Educational Communications and Technology.

Kafai, Y. B., Burke, Q. & Resnick, M. (2014). *Connected code: Why children need to learn programming* Mit Press.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior, 52*, 200-210.

Kidsakoder. (2016). 03/23, 2016,Hentet fra http://kidsakoder.no/

Kidsakoder. (u.å). Kodeklubbens oppgaver. 11/10, 2016,Hentet fra http://kodeklubben.github.io/

kidsakoder. (u.å). Lær Kidsa Koding. 08/05, 2016,Hentet fra http://kidsakoder.no/

Kim, B. (2001). Social constructivism. *Emerging Perspectives on Learning, Teaching, and Technology, 1*(1), 16.

Knobelsdorf, M., & Romeike, R. (2008). Creativity as a pathway to computer science. *ACM SIGCSE Bulletin, , 40*(3) 286-290.

Kostøl, K. B., & Remmen, K. B. (2016). Lektor 2: Læring gjennom oppdrag. 11/23, 2016,Hentet fra http://www.lektor2.no/c1336836/nyhet/vis.html?tid=2160488

Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM, 50*(4), 36-42.

Krumsvik, R. J. (2014). *Forskingsdesign og kvalitativ metode : ei innføring*. Bergen: Fagbokforlaget.

Kunnskapsdepartementet. (2016). Meld. St. 28 (2015–2016). Fag – Fordypning – Forståelse En fornyelse av Kunnskapsløftet. 08/10, 2016,Hentet fra https://www.regjeringen.no/contentassets/e8e1f41732ca4a64b003fca213ae663b/no/pdfs/stm201520160028000dddpdfs.pdf

Kvale, S. & Brinkmann, S. (2015). I Brinkmann S., Anderssen T. M. and Rygge J. (Red.). *Det kvalitative forskningsintervju* (3. utg., 2. oppl. utg.). . Oslo: Gyldendal akademisk.

Kynigos, C. (2008). Black-and-white-box perspectives to distributed control and constructionism in learning with robotics. *Proceedings of SIMPAR Workshops,* 1-9.

Lang-Ree, H. L. (2016). " Vi må tenke og ikke bare tegne": En kvalitativ studie om bruk av programmering som verktøy i arbeid med matematikk.

Lee, T. Y., Mauriello, M. L., Ahn, J. & Bederson, B. B. (2014). CTArcade: Computational thinking with games in school age children. *International Journal of Child-Computer Interaction, 2*(1), 26-33.

Livingstone, I., & Hope, A. (2011). Next Gen. 3/24, 2015,Hentet fra http://www.education.gov.uk/ta-assets/~/media/get_into_teaching/resources/subjects_age_groups/cs_next_generation.pdf

Maxwell, J. W. (2006). Re-Situating Constructionism. *The international handbook of virtual learning environments* (s. 279-298) Springer.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? *American Psychologist, 59*(1), 14.

McDowell, C., Werner, L., Bullock, H. & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. *ACM SIGCSE Bulletin, 34*(1), 38-42.

Merriam-Webster. (u.å). Learning - definition of learning. 08/01, 2016,Hentet fra http://www.merriam-webster.com/dictionary/learning

Molnar, A. (1997). Computers in Education. 02/08, 2016,Hentet fra https://thejournal.com/Articles/1997/06/01/Computers-in-Education-A-Brief-History.aspx?Page=3

Mozelius, P. (2014). Game based learning–a way to stimulate intrinsic motivation. *International Conference on E-Learning ICEL 2014. Academic Conferences Publishing,* 272-278.

NOU 2015:8. (2015). *Fremtidens skole : Fornyelse av fag og kompetanser*. Oslo: Kunnskapsdepartementet.

P21.org. (2015). Framework for 21st century learning. 11/29, 2015,Hentet fra http://www.p21.org/about-us/p21-framework

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas* Basic Books, Inc.

Papert, S. (1987). Computer criticism vs. technocentric thinking. *Educational Researcher, 16*(1), 22-30.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95-123.

Papert, S., & Harel, I. (1991). Situating Constructionism. 3/24, 2015,Hentet fra http://www.papert.org/articles/SituatingConstructionism.html

Pea, R. D. & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137-168.

Postholm, M. B. (2010). *Kvalitativ metode: en innføring med fokus på fenomenologi, etnografi og kasusstudier* (2 utg utg.). . Oslo: Universitetsforl.

Processing.org. (u.å). 3/08, 2016,Hentet fra https://processing.org/overview/

Regjeringen. (2015). Koding blir valgfag på ungdomstrinnet. 11/26, 2015,Hentet fra https://www.regjeringen.no/no/aktuelt/koding-blir-valgfag-pa-ungdomsskolen/id2435271/

Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition,* 1-6.

Resnick, M. & Rosenbaum, E. (2013). Designing for tinkerability. *Design, make, Play: Growing the Next Generation of STEM Innovators,* , 163-181.

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. *Proceedings of the 2005 Conference on Interaction Design and Children,* 117-122.

Rushkoff, D. (2010). *Program or be programmed: Ten commands for a digital age* Or Books.

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., . . . Voll, L. O. (2016). Teknologi og programmering for alle - En faggjennomgang med forslag til endringer i grunnopplæringen- august 2016. 10/24, 2016,Hentet fra http://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf

Scaffidi, C. & Chambers, C. (2012). Skill progression demonstrated by users in the Scratch animation environment. *International Journal of Human-Computer Interaction, 28*(6), 383-398.

Schunk, D. H. (1996). Learning theories. *Printice Hall Inc., New Jersey,*

Sevik, K. (2015). Koding i skolen. 04/18, 2016,Hentet fra https://iktsenteret.no/sites/iktsenteret.no/files/attachments/koding_i_skolen_-_sikt-notat_nr_2_-_temakonferanse_2015.pdf

Smith, M. (2016). Computer Science For All. 05/19, 2016,Hentet fra https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

Smith, N., Sutcliffe, C., & Sandvik, L. (2014). Code club: Bringing programming to uk primary schools through scratch. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education,* 517-522.

Sommerfeldt, S. (2013). Nødvendigheten av god digital kunnskap blant barn. 10/24, 2016,Hentet fra https://docs.google.com/document/d/1rwbwiaNRNOsudMh6bb4mp4-60D-8JP__86zQ-g3TCTE/edit

Turkle, S. & Papert, S. (1992). Epistemological pluralism and the revaluation of the concrete. *Journal of Mathematical Behavior, 11*(1), 3-33.

UDIR. (2016). Forsøk med programmering som valgfag. 04/09, 2016,Hentet fra http://www.udir.no/Lareplaner/Forsok-og-pagaende-arbeid/forsok-med-programmering-som-valgfag/

UDIR. (u.åa). Læreplan i informasjonsteknologi - programfag i utdanningsprogram for studiespesialisering (INF1-01). 08/10, 2016,Hentet fra http://www.udir.no/kl06/INF1-01

UDIR. (u.åb). Veiledninger til Kunnskapsløftet. 08/10, 2016,Hentet fra www.udir.no

Vos, N., Van Der Meijden, H. & Denessen, E. (2011). Effects of constructing versus playing an educational game on student motivation and deep learning strategy use. *Computers & Education, 56*(1), 127-137.

Wikipedia. (2016). Minecraft Mods. 07/22, 2016,Hentet fra https://en.wikipedia.org/wiki/Minecraft_mods

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725. doi: 10.1098/rsta.2008.0118

Vedlegg 1 - intervjuguide

Intervjuguide

Bakgrunnsinformasjon:

Kjønn: Mann/Kvinne

Navn:

Alder:

Utdanning/Bakgrunn:

Problemstilling:

> Hvordan kan erfaringer fra kodeklubbkonseptet være med å utvikle elevenes programmeringskompetanse i skolen?

Forskningsspørsmål:

> Hvilke erfaringer har man fra kodeklubbkonseptet?
>
> Hvilken programmeringskompetanse trenger elevene?

Generelle spørsmål:

> Fortell litt om deg selv, hvordan du kom i gang som kodeklubbveileder, hvilke erfaringer du selv har fra programmering/datateknologi/undervisning osv. Hva er motivasjonen for å fortsette?
>
> Kan du beskrive bakgrunnen og interessen du og dine medveiledere har for å drive kodeklubb?

Erfaringer

> Kan du fortelle litt om hvordan aktiviteten er organisert hos dere? Hva er ressurssituasjonen?
>
> Hvordan opplever du kjønnsfordeling og kjønnsforskjeller i kodeklubben du er med i?
>
> Kan du fortelle litt om undervisningsoppleggene dere mest bruker, hvor har dere hentet dem, hva tenker du om den pedagogiske tilnærmingen osv.?

Hva mener du kjennetegner barnas motivasjon i kodeklubben?

Hvilken rolle tar du som veileder? Viser hvordan ting skal gjøres? Stiller undringsspørsmål? «Gjør» det for dem?

Hva tror du er grunnen til at dette er blitt populært?

Barnas programmeringsferdigheter

Hva tenker du er det viktigste barna lærer i kodeklubben?

Hvordan foregår en eventuell overgang fra de helt grafiske verktøyene til tekstbaserte verktøy?

Hva tenker du er det viktigste barna lærer om programmering?

Overføringsverdi til skolen

Hvordan ser du for deg at programmering i større grad kunne trekkes inn i skolen? I hvilke fag/aktiviteter kunne dette hatt sin plass?

Noen andre tanker om hva skolen kunne lære av kodeklubbene?

Koding kan være inngang til f.eks lesing og skriving.

Hva mener du eventuelt er gode argumenter for å innføre mer koding i skolen?

Har du noe å tillegge til slutt?

Vedlegg 2 – Kvittering fra NSD omkring behandling av personopplysninger

Norsk samfunnsvitenskapelig datatjeneste AS
NORWEGIAN SOCIAL SCIENCE DATA SERVICES

Harald Hårfagres gate 29
N-5007 Bergen
Norway
Tel: +47-55 58 21 17
Fax: +47-55 58 96 50
nsd@nsd.uib.no
www.nsd.uib.no
Org.nr. 985 321 884

Anders Grov Nilsen
Avdeling for lærerutdanning og kulturfag Høgskolen Stord/Haugesund
Klingenbergvegen 8
5414 STORD

Vår dato: 04.01.2016 Vår ref: 45780 / 3 / HIT Deres dato: Deres ref:

TILBAKEMELDING PÅ MELDING OM BEHANDLING AV PERSONOPPLYSNINGER

Vi viser til melding om behandling av personopplysninger, mottatt 23.11.2015. Meldingen gjelder prosjektet:

45780	*Hva kan skolen lære av kodeklubbene.* *Masterarbeid hvor jeg skal undersøke aktivitet i en kodeklubb i regi av "lær kidsa koding"* *http://www.kidsakoder.no/*
Behandlingsansvarlig	*Høgskolen Stord/Haugesund, ved institusjonens øverste leder*
Daglig ansvarlig	*Anders Grov Nilsen*
Student	*Egil Bjørnevoll*

Personvernombudet har vurdert prosjektet og finner at behandlingen av personopplysninger er meldepliktig i henhold til personopplysningsloven § 31. Behandlingen tilfredsstiller kravene i personopplysningsloven.

Personvernombudets vurdering forutsetter at prosjektet gjennomføres i tråd med opplysningene gitt i meldeskjemaet, korrespondanse med ombudet, ombudets kommentarer samt personopplysningsloven og helseregisterloven med forskrifter. Behandlingen av personopplysninger kan settes i gang.

Det gjøres oppmerksom på at det skal gis ny melding dersom behandlingen endres i forhold til de opplysninger som ligger til grunn for personvernombudets vurdering. Endringsmeldinger gis via et eget skjema, http://www.nsd.uib.no/personvern/meldeplikt/skjema.html. Det skal også gis melding etter tre år dersom prosjektet fortsatt pågår. Meldinger skal skje skriftlig til ombudet.

Personvernombudet har lagt ut opplysninger om prosjektet i en offentlig database, http://pvo.nsd.no/prosjekt.

Personvernombudet vil ved prosjektets avslutning, 31.12.2016, rette en henvendelse angående status for behandlingen av personopplysninger.

Vennlig hilsen

Katrine Utaaker Segadal

Hildur Thorarensen

Dokumentet er elektronisk produsert og godkjent ved NSDs rutiner for elektronisk godkjenning.

Avdelingskontorer / *District Offices:*
OSLO: NSD. Universitetet i Oslo, Postboks 1055 Blindern, 0316 Oslo. Tel: +47-22 85 52 11. nsd@uio.no
TRONDHEIM: NSD. Norges teknisk-naturvitenskapelige universitet, 7491 Trondheim. Tel: +47-73 59 19 07. kyrre.svarva@svt.ntnu.no
TROMSØ: NSD. SVF, Universitetet i Tromsø, 9037 Tromsø. Tel: +47-77 64 43 36. nsdmaa@sv.uit.no

Kontaktperson: Hildur Thorarensen tlf: 55 58 26 54
Vedlegg: Prosjektvurdering
Kopi: Egil Bjørnevoll egil.bjornevoll@ski.kommune.no

Vedlegg 3 – Informasjonsskriv til deltakere

Forespørsel om deltakelse i forskningsprosjektet

" Hva kan skolen lære av kodeklubben?»

Bakgrunn og formål

Gjennom mine studier i IKT og læring på Høgskulen i Stord/Haugesund har jeg blitt interessert i programmeringsaktiviteter for barn/unge, og skriver en masteroppgave om dette.

Hovedproblemstilling:

Hvordan kan erfaringer fra kodeklubbkonseptet være med å utvikle elevenes programmeringskompetanse i skolen?

Forskningsspørsmål:
1. *Hvilke erfaringer har man fra kodeklubbkonseptet?*
2. *Hvilken programmeringskompetanse trenger elevene?*

Jeg vil altså fokusere på veiledernes opplevelser og erfaringer med arbeidet i kodeklubben og vil finne ut om hva du anser som spesielt viktig at barn/unge lærer om programmering.

Hva innebærer deltakelse i studien?

Studien er kvalitativ, dvs. jeg søker å gå i dybden på et avgrenset felt, og finne ut så mye jeg kan om opplevelsen og erfaringen til de som deltar i studien.

Intervjuet vil ta omtrent en time. Jeg kommer til å benytte lydopptak i intervjuet, men kommer til å anonymisere alle personopplysninger og slette alle opptak ved prosjektslutt. Hvis intervjuet gjøres digitalt ønsker jeg videopptak av møtet. Dette vil også bli slettet etter prosjektslutt.

Hva skjer med informasjonen om deg?
Alle personopplysninger vil bli behandlet konfidensielt. Kun min veileder og jeg selv vil ha tilgang til informasjonen jeg samler inn. Koblingsnøkkel, dvs. dokumentet hvor man kan lese ut hvem som snakker på den enkelte lydopptakene vil kun oppbevares på en ekstern harddisk, mens lydopptakene er lagret på en privat PC uten noen ekstern tilgang fra andre maskiner i mitt hjemmenettverk. Alle slike dokumenter skal også være passordbeskyttet.

I min ferdige masteroppgave vil ikke deltakere, eller kodeklubbene det er snakk om, kunne gjenkjennes.

Prosjektet skal etter planen avsluttes 31.12.2016.

Frivillig deltakelse
Det er frivillig å delta i studien, og du kan når som helst trekke ditt samtykke uten å oppgi noen grunn. Dersom du trekker deg, vil alle opplysninger om deg bli slettet.

Har du spørsmål om studien, ta kontakt med Egil Bjørnevoll på telefon 92 45 81 78. Min veileder er Anders Grov Nilsen, tlf. 48 24 05 76 eller epost anders.nilsen@hsh.no, hvis du har spørsmål til ham.

Studien er meldt til Personvernombudet for forskning, Norsk samfunnsvitenskapelig datatjeneste AS.

Samtykke til deltakelse i studien

Jeg har mottatt informasjon om studien, og er villig til å delta

---

(Signert av prosjektdeltaker, dato)

Vedlegg 4 – Matrise for litteratursøk

Søkeord	Antall resultater	Relevante artikler	Gjentakelser
Oria			
Programmering + læring	75	3	
Programmering + undervisning	55	7	
Programmering + konstruksjonisme	126	6	3
Code club + programming	1	1	1
Programming + computational thinking	56	20	1
Academic Search premier			
Programmering + læring	25	6	1
Programmering + undervisning	31	7	1
Programmering + konstruksjonisme	11	5	1
Code club + programming	1	0	0
Programming + computational thinking	14	9	0
ERIC			
Programmering + læring	111	7	5
Programmering + undervisning	54	6	4
Programmering + konstruksjonisme	5	1	1
Code club + programming	2	0	0
Programming + computational thinking	29	12	0