Høgskulen på Vestlandet

Masteroppgave

MASIKT-OPG

Predefinert informasjon

Startdato:	27-05-2019 09:00	**Termin:**	2019 VÅR
Sluttdato:	03-06-2019 14:00	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	MasIKT-opg: Masteroppgave		
SIS-kode:	203 MASIKT-OPG 1 OM-1 2019 VÅR stord		
Intern sensor:	Per Ivar Kjærgård		

Deltaker

Navn:	Christina Hemnes
Kandidatnr.:	109
HVL-id:	095652@hvl.no

Informasjon fra deltaker

Tittel *:	Programmering og problemløsning i småskolen
Antall ord *:	29856
Navn på veileder *:	Anders Grov Nilsen

Egenerklæring *: Ja

Jeg bekrefter at jeg har registrert oppgavetittelen på norsk og engelsk i StudentWeb og vet at denne vil stå på vitnemålet mitt *: Ja

Jeg godkjenner avtalen om publisering av masteroppgaven min *

Ja

Er masteroppgaven skrevet som del av et større forskningsprosjekt ved HVL? *

Nei

Høgskulen på Vestlandet

MASTEROPPGAVE

Programmering og problemløsning i småskolen

Programming and problem-solving in early primary school (K-4)

Christina Hemnes

Master i IKT i læring
Institutt for pedagogikk, religion og samfunnsfag
Veileder: Anders Grov Nilsen
Innlevert dato: 03.06.2019

Forord

Å skrive forordet til denne masteroppgaven kjennes rart. To år med hardt arbeid, lesing av forskningslitteratur og teori, oppgaveskriving og faglige diskusjoner nærmer seg slutten. Jeg er takknemlig for muligheten jeg har fått til å fordype meg i fagområdet jeg har skrevet oppgaven min om. På høgskolen på Stord har jeg blitt kjent med en herlig bukett med dyktige forelesere og faglærere. Jeg må spesielt takke min supre veileder Anders Grov Nilsen for uvurderlig og konstruktiv veiledning! Du har gjort veien mot masterinnleveringen så mye enklere å gå, og jeg er takknemlig for alle Skype-møter vi har hatt og alle sendte mail med gode råd og nyttige tips for hvordan jeg skulle ro dette prosjektet i land.

Jeg må også takke medstudentene mine! Dere har vært så gode samtalepartnere å ha, og gjort studiesamlingene på Stord så trivelige. Disse årene på masterstudie har bidratt til ny og forbedret kunnskap om programmering og IKT i skolen som jeg kan ta med meg rett inn i klassen min, og dele med flotte kollegaer! Jeg vil takke min arbeidsgiver, Fjell kommune, og min skoles ledelse for at de har vært tålmodige og fleksible med mine søknader om studiepermisjoner. Takk for at dere har lagt til rette for at jeg kunne levere denne oppgaven på normert tid!

Min kjære mann, Øyvind, må berømmes for sin tålmodighet og omtenksomhet dette siste året. Du har stilt ekstra opp på hjemmefronten og som mitt personlige støttende stillas! Jeg skylder deg en del timer med kinodater, ski- og fisketurer!

Sammendrag

Denne masteroppgaven handler om hvordan programmering i småskolen kan bidra til at elever utvikler problemløsningsferdigheter. Programmering og algoritmisk tankegang er i vinden i norsk læreplanarbeid, og mange skoler har også innført kodetimer for klassene sine. For at skolen skal undervise i og benytte seg av programmeringsaktiviteter for at elever skal tilegne seg algoritmisk tankegang, trenger vi forskning som problematiserer og vurderer hvordan denne opplæringen skal skje i norske klasserom, med de rammefaktorene og forholdene som finnes der. Problemstillingen er derfor formulert slik: Hva kjennetegner programmeringsøkter i småskolen der barn får utvikle problemløsningsferdigheter? Sammen med problemstillingen står to forskningsspørsmål. F1: Hva mener lærere fremmer og hemmer småskoleelevers muligheter for å utvikle problemløsningsferdigheter sammen med andre når de jobber med programmering? F2: Hva kjennetegner situasjoner der barn får mulighet til utvikling av problemløsning når de jobber med testing og feilsøking? For å finne svar på problemstillingen har forskningsdesignet vært utformet som et casestudie. Metodene som er benyttet er intervjuer og elevobservasjoner. Informanter i intervjuene er skolens avdelingsleder og lærere (kontaktlærer og andrepedagog) tilhørende et av småskoletrinnene på skolen. En klasse på småskoletrinnet er observert i to undervisningsøkter. Sosiokulturelle læringsteorier ligger som et bakteppe for å forstå hvordan elevene utvikler problemløsningsferdigheter gjennom programmering. Oppgaven presenterer også teori om problemløsning i en sosial kontekst (TASC) og teori om *computational thinking.* Analysene og drøftingene i etterkant av datainnsamlingen peker på hvor viktig interaksjonene mellom barn, håndgripelige programmeringsverktøy og voksne er for at elever skal utvikle problemløsningsferdigheter. Lærerens uvurderlige støtte og veiledning fungerer som et støttende stillas for barna, og læreren avgrenser og tydeliggjør veien videre i programmeringen for barna. Konklusjonene handler om at støttende stillas er uvurderlig for barn som strever med problemløsning i programmering. Fysiske programmeringsverktøy som brukes med digital flate konkretiserer testing og feilsøking for barna. Det pekes også på sammenhenger mellom muligheter for å utvikle problemløsningsferdigheter og programmeringsoppgaver. Tydelige mål med programmeringen vil føre til at testing og feilsøking faller mer naturlig for barn når de bruker fysiske programmeringsverktøy.

Abstract

This master's thesis examines how programming can help early primary students (K-4) develop problem-solving skills. The main research question is: What characterizes programming sessions in early primary schools where children develop problem-solving skills? Along with the main issue, two research questions are proposed. Q1: What do teachers think promotes and inhibits the ability of early primary school students to develop problem-solving skills with others when working with programming? Q2: What characterizes situations where children have the opportunity to develop problem solving when they are working with testing and debugging? In order to find answers to the problem, the research design has been constructed as a case study. The case has been centred around a single case, a school that has implemented programming in all its classes and plans at the school. The methods used are interviews and student observations. Informants in the interviews are the school's deputy headteacher and two teachers (contact teacher and other educator) belonging to one of the early primary classes at the school. An early primary class is observed in two teaching sessions. Socio-cultural learning theories are chosen as a theoretical framework for understanding how students develop problem-solving skills through programming. The thesis also presents theory of problem solving in a social context (TASC) and theory of computational thinking.

The findings and discussions after the data collection discuss how important the interactions between children, tangible programming tools and adults are for students to develop problem solving skills. The interactions between the children and the teacher's invaluable support and guidance serves as a supportive scaffold, and the teacher delimits and clarifies the way forward in the programming activities for the children. The master thesis concludes that supportive scaffolding is invaluable to children struggling with problem solving in programming. Tangible programming tools used with digital surfaces concretise testing and debugging. The conclusions also points to relationships between opportunities for developing problem solving skills and programming tasks. Clear programming goals will make testing and debugging more natural for children to use, when they are using tangible programming tools.

Innholdsfortegnelse

Forord 1

Sammendrag 2

Abstract 3

1. Innledning 6

1.1. Bakgrunn for valg av tema 6

1.2. Problemstilling og forskningsspørsmål 9

1.3. Begrepsavklaring: Computational thinking 9

1.4. Tidligere forskning knyttet til tema 12

2. Teori knyttet til programmering og problemløsning i skolen 21

2.1. Sosiokulturell læringsteori 21

2.1.1. Vygotskys syn på barns utvikling og den proksimale utviklingssonen 21

2.1.2. Scaffolding 22

2.1.3. Medierende artefakter 24

2.2. Problemløsning 24

2.3. Computational thinking 30

2.4. Hvilke konsekvenser har teoriene ovenfor for mitt forskningsprosjekt? 33

3. Forskningsdesign og metode 35

3.1. Forskningsdesign 35

3.2. Mitt epistemologiske utgangspunkt 36

3.3. Kvalitativ og kvantitativ tilnærming 36

3.4. Casestudiedesign 37

3.5. Metoder 39

1.2. Analyse 46

1.3. Gyldighet og pålitelighet 48

1.4. Etikk 50

2. Analyser av datamaterialet 51

4.1. Analyse av intervju 51

2.2. Analyse av observasjon 65
3. Diskusjon 73
3.1. Det støttende stillasets betydning 74
3.2. Programmering og problemløsning 79
3.3. Fysiske, visuelle programmeringsverktøy 85
4. Avslutning 87
4.1. Konklusjoner 87
4.2. Avgrensninger 90
4.3. Fremtidig forskning 90
5. Kilder/referanser 91
6. Vedlegg 95
Vedlegg 1: Intervjuguide for lærere og avdelingsleder 96
Vedlegg 2: Oversikt over koder og kategorier i analysearbeidet 97
Vedlegg 3: Kvittering for godkjent NSD-søknad 98
Vedlegg 4: Informasjonsskriv og samtykkeskjema til lærere 100
Vedlegg 5: Informasjonsskriv og samtykke til observasjon 102
Vedlegg 6: Feltnotater, mal for observasjonsnotat 104

1. Innledning

1.1. Bakgrunn for valg av tema

1.1.1. Personlige mål for prosjektet

I 2009 begynte jeg på allmennlærerutdanningen. Jeg fordypet meg i realfag, og bygget opp lærerutdanningen blant annet med en årsenhet i matematikk og halvårsenhet i naturfag. Som elev i grunnskolen og student har jeg alltid likt realfagene på grunn av at det gjerne finnes faste regler, algoritmer og systemer for å løse oppgaver. Da jeg kom ut i skolen som lærer, så jeg etter hvert at en del av elevene mine opplevde disse fagene som utfordrende. Ofte gav elevene uttrykk for at matte var vanskelig eller kjedelig, fordi de ikke fant det rette svaret eller var redd for å gjøre feil. Som lærer så jeg at en del av elevene manglet redskaper eller tvilte på sine egne strategier for problemløsning. I stedet for å angripe et problem med utforskende, lekende eller nysgjerrige tilnærminger, prøvde elevene å slippe unna problemet med å *spørre etter riktig svar* på oppgaven; lettkjøpte løsninger. Disse erfaringene gjorde at jeg ble opptatt av problemløsningsferdigheter i skolen.

I 2017 startet jeg masterstudiet i IKT og læring. Det kom signaler fra skoleledere, fagforbund og politiske sentrale styresmakter om økt fokus på læring på, av og gjennom digitale flater, og jeg ønsket kompetanse på området for å bli en (profesjonsfaglig) digital kompetent lærer. Jeg kunne ingenting om programmering før jeg startet på HVL, er for ung til å huske eller ha kjennskap til Commodore 64, og det var egentlig en veldig heldig feil at jeg meldte meg på en teknologisk (og ikke allmenndidaktisk) profil for masteren. Takket være lærere på HVL som sterkt antydet at jeg i alle fall burde prøve profilen, ble jeg værende her. Jeg lærte om programmeringen sin historie i skolen, om lett tilgjengelige verktøy og programmeringsspråk som kunne brukes i klasserommet. På samme tid var jeg lærer for en 2. klasse, og sammen lærte og testet vi ut verktøy og programmeringsoppgaver. Jeg så hvor ivrig elevene arbeidet med programmering, og deres entusiasme over å kunne hjelpe hverandre med å løse problemer i programmeringen var inspirerende. På den tiden hadde jeg kolleger og venner som naturlig nok var litt skeptiske til hvorfor og hvordan programmeringen skulle inn i skolen (og særlig småskolen), og jeg bestemte meg derfor å forske på hvordan jeg kunne forsvare programmeringen sin plass i barneskolen.

1.1.2. Programmering i skolen

NOU 2015:8: Fremtidens skole- fornyelse av fag og kompetanser

NOU-meldingen *Fremtidens skole – fornyelse av fag og kompetanser* (NOU 2015:8, 2015) hadde som mål å kartlegge hvilke kompetanser man så behov for i fremtiden. NOU-meldingen presenterer fire kompetanseområder utvalget, som i mediene blir referert til som Ludvigsenutvalget, mente norske elever burde tilegne seg for fremtiden.

Den første er *fagspesifikk kompetanse*, og den er knyttet til fagområder som matematikk, naturfag og teknologi, språk, praktiske og estetiske fag, og samfunns- og etikkfag. Dette er kompetanser som er viktige i forbindelse med spesifikke yrkes- eller utdanningsvalg, men også allmenndanning. Dessuten blir problemløsning og kritisk tenking trukket frem under fagspesifikk kompetanse, og da blir det naturlig å se programmering i sammenheng med dette: «vitenskapelige metoder og tenkemåter er relevante for fremtiden, og dette ses i sammenheng med behovet for å kunne tenke kritisk og løse problemer» (NOU 2015:8).

Det neste kompetanseområdet er *å kunne lære*, og da er metakognisjon og selvregulert læring nødvendig for videre læring. Utvalget understreker at disse områdene utvikles i samspill med andre (NOU 2015:8). Her trekkes det frem at elevene skal opplæres til selv å kunne videreutvikle sine ferdigheter og kunnskap i løpet av livet.

> Metakognisjon handler om å kunne reflektere over egen tenkning og læring. I læringssammenheng handler det om at elevene reflekterer over hvorfor de lærer, hva de har lært, og hvordan de lærer. Metakognisjon betyr også å kunne bruke tenkemåter og læringsstrategier aktivt og målrettet for å fremme egen læring. (NOU 2015:8)

Også her kan programmering knyttes opp til innholdet som ligger i kompetanseområdet. Metakognisjon og selvregulering kan stimuleres i programmeringsaktiviteter i skolen, dersom rammefaktorene og miljøet rundt eleven legger til rette for det.

Det tredje kompetanseområdet er *å kunne kommunisere, samhandle og delta*. I dette kompetanseområdet ligger muntlige ferdigheter, men også ferdigheter innen lesing og skriving. Utvalget trekker frem evnen til å argumentere, diskutere og samarbeide som viktige evner for fremtida. Programmeringsaktiviteter i et klassefellesskap kan være med på å trene opp slike ferdigheter, der barna skal argumentere for valg og løsninger, hjelpe hverandre, spørre og formulere seg korrekt.

Å kunne utforske og skape er det siste kompetanseområdet til Ludvigsen-utvalget, og utvalget «anbefaler at kreativitet, innovasjon, kritisk tenkning og problemløsning er kompetanser skolen bør bidra til at elevene utvikler» (NOU 2015:8). De argumenterer med at arbeids- og samfunnslivet trenger skapende mennesker, både nasjonalt og internasjonalt. Mennesker som er trent på å finne løsninger på problemer i skole- og arbeidssammenheng, kan muligens bruke de samme strategiene i andre deler av livet sitt. Her er en kobling til programmering naturlig, fordi det å skape egne programmer krever at man kan finne nye, innovative løsninger på problemer. «Nysgjerrighet, utholdenhet, åpenhet for å se ting på nye måter og evne til å ta initiativ er viktige sider ved kompetansene. Unge mennesker er av natur undrende og utforskende, men nysgjerrighet må stimuleres for å utvikles» (NOU 2015:8).

Læreplanarbeid

I flere land, som England, Frankrike, Finland, Irland og Malta, har programmering og *computational thinking* fått utvikle seg til å bli en nøkkelprioritet i obligatorisk undervisning og skolegang (Bocconi, S., Chioccariello, A. and Earp, J., 2018). I 2018 ble kjerneelementene for den nye *norske* læreplanen presentert. Her har algoritmisk tankegang og programmering kommet inn som eget tema (Utdanningsdirektoratet, 2018). Kjerneelementene står uavhengig klassetrinn, og de skal ramme inn det viktigste i faget og beskrive det elevene må lære for å mestre faget (Utdanningsdirektoratet, 2018). Ett av kjerneelementene er *Utforsking og problemløsning*. Her er det trukket frem at elevene skal fokusere mer på strategier og fremgangsmåter, enn på løsninger. «Algoritmisk tenking er viktig i prosessen med å utvikle strategiar og framgangsmåtar for å løyse problem» (Utdanningsdirektoratet, 2019). Det neste kjerneelementet som trekkes frem er *Abstraksjon og generalisering*. Også her finner vi elementer som kan kobles sammen med programmeringsaktiviteter, fordi elevene skal oppdage sammenhenger og strukturer og ikke bli presenterte for en ferdig løsning (Utdanningsdirektoratet, 2019).

Ifølge utkastet for de nye læreplanene (pr. 18. mars 2019) skal algoritmisk tankegang være en del av matematikkopplæringen for grunnskolen (1.-10. klasse fellesfag). I høringsutkastet til de nye læreplanene i matematikk skriver Utdanningsdirektoratet i en innledning:

> Vi har tatt algoritmisk tenkning og programmering inn i faget. Vi har vektlagt algoritmisk tenkning fordi dette er en viktig problemløsningsstrategi. Når

> elevene bruker programmering til å utforske og løse problemer, kan det være et godt verktøy for å utvikle matematisk forståelse. (Utdanningsdirektoratet, 2019)

Det er verdt å merke seg at de skriver at programmering *kan* være et godt verktøy for å utvikle problemløsningsstrategier. Det indikerer at andre faktorer, f.eks. valg av verktøy, arbeidsmåter, organisering, læringsmiljø, osv., har mye å si for utbyttet elevene får ut av å arbeide med programmering. Forskning fra land som allerede har innført programmering i skolen er ikke nødvendigvis overførbart til norske forhold. Man kan tenke seg at rammefaktorene i norske klasserom skiller seg fra klasserom fra England, Malta og Frankrike, som nevnt over (Bocconi, et al., 2018). Derfor er det viktig at det forskes på programmering i norske klasserom og at vurdering av rammefaktorene også blir tatt med som en del av disse studiene.

1.2.Problemstilling og forskningsspørsmål

En rekke forskere på feltet etterlyser forskning på hvordan barna blir støttet og veiledet i problemløsning når de programmerer (Fessakis, Gouli & Mavroudi, 2013; Lye & Koh, 2014). Det er også behov for skildringer som forteller noe om programmering og problemløsning i norske klasserom. Etter en vurdering av ulike innfallsvinkler endte jeg derfor opp med:

Problemstilling: Hva kjennetegner programmeringsøkter i småskolen der barn får utvikle problemløsningsferdigheter sammen med andre?

F1: Hva mener lærere fremmer og hemmer småskoleelevers muligheter for å utvikle problemløsningsferdigheter sammen med andre når de jobber med programmering?

F2: Hva kjennetegner situasjoner der barn får mulighet til utvikling av problemløsning når de jobber med testing og feilsøking?

Ved å stille disse forskningsspørsmålene, kan jeg undersøke om det lærerne har intensjoner om, planlegger eller har vurdert som viktig for problemløsningsutviklingen til elevene, viser seg i praksis i klasserommet.

1.3.Begrepsavklaring: Computational thinking

Sentralt i den forskningen som handler om programmering og problemløsning står begrepet *computational thinking*. I dette delkapittelet vil jeg begrunne hvorfor jeg forholder meg til begrepet *computational thinking* (og ikke algoritmisk tankegang)

videre i masteroppgaven. Dette er et sentralt begrep som går igjen allerede fra kapittelet om tidligere forskning og til avslutningskapittelet i denne avhandlingen.

I 2008 formulerte Jeanette Wing (2008) en definisjon av *computational thinking* som er referert til i flere forskningsartikler siden (Fessakis, et al., 2013; Kafai, et al., 2014; Palmèr, 2017). I følge Wing er *computational thinking*: «design systems, solve problems, and understand human behavior, by drawing on the concepts fundamental to computer science» (Wing, 2008, s.3717). Begrepet problemløsning kan knyttes til algoritmisk tankegang, fordi Wing (2008) definerer en algoritme som «an abstraction of a step-by-step procedure for taking input and producing some desired output» (Wing, 2008, s. 3718). Steg-for-steg-prosedyren for å finne en ønsket løsning på et problem.

Teorien som er valgt i dette masterprosjektet legger vekt på elementer og prinsipper som kan defineres under paraplyen *computational thinking (CT).* I teorikapittelet nedenfor blir dimensjonene ved CT presentert, men jeg nevner dem her for å begrunne hvorfor jeg benytter CT som begrep. Brennan & Resnick (2012) deler computational thinking inn i *computational concepts, computational practices* og *computational perspectives.* Det er særlig *computational practices* og *perspectives* som er relevant for problemløsning. På grunn av denne inndelingen forstår jeg CT som et mer dekkende begrep for programmeringsundervisning med fokus på problemløsning sammen med andre, enn algoritmisk tankegang. Dimensjonene ved CT gir mulighet for en bred forståelse av hvilken kunnskap og hvilke ferdigheter vi her er ute etter, noe som blir nyttig når forskningsdata skal analyseres og drøftes.

I Norge brukes begrepet algoritmisk tenking om elementene i *computational thinking*. UDIR (2019) publiserte 27.03.2019 en artikkel, der de beskriver *den algoritmiske tenkeren* og kjennetegn på algoritmisk tankegang. Figuren de beskriver er delt inn i nøkkelbegreper for algoritmisk tenking og arbeidsmåter som brukes for å løse problemer. Arbeidsmåtene til høyre i figuren kan sammenlignes med *comptuational practices* (se teorikapittelet om CT).

Figur 1: Den algoritmiske tenkeren. Nøkkelbegrep og arbeidsmåter knyttet til algoritmisk tenking. Figur hentet med tillatelse fra UDIR (2019).

I det videre arbeidet velger jeg å forholde meg til begrepet *computational thinking.* Ved å fokusere på dimensjonene ved CT i teorikapittelet, får jeg en ryddigere presentasjon av begrepet. Det blir enklere å forholde seg til fenomenet programmering i klasserommet når jeg kan være tydelig på hvilken dimensjon det fokuseres på. Når det er sagt, er arbeidsmåtene til høyre i figuren over konkretisert og brutt ned til *gjenkjennelige tegn* på problemløsning, noe som er en fordel når jeg skal lete kjennetegn på et fenomen (jf. F2). Jeg kan ikke finne noen motsetninger mot elementene i figuren og innholdet i dimensjonene om CT. Men ved å velge CT som overordnet begrep, deles begrepet inn i tre dimensjoner, noe som gjør det lettere å få en oversikt over fenomenet og dermed analysere funnene opp mot teori og tidligere forskning.

1.4.Tidligere forskning knyttet til tema

I arbeidet med selve masteroppgaven, ble det utført et litteratursøk med en påfølgende litteraturgjennomgang, basert på tidligere forskning på problemløsning og programmering i småskolen, fra 1.-4. klasse i det norske skolesystemet. For å treffe best internasjonalt har jeg brukt k-4, som skal begrense søket til og med elever opp til 4. klasse. Søkeordene er valgt ut for å spisse litteratursøket inn mot forskning fra småskolen og problemløsning.

Litteratursøket i tabellform:		
Tema	**Inkludert**	**Ekskludert**
Database	Eric, Google scholar, Oria, Science Direct	Alle andre databaser.
Tid	Siste seks år	Alt annet. Trenger nyere forsking å vise til
Fokus	Vitenskapelige artikler, bokkapitler og rapporter om programmering i grunnskolen (eller tilsvarende i andre land)	Publikasjoner om programmering knyttet til videregående, fagskoler, universitet og høyskole.
Type aktivitet	Publikasjoner om programmering som sier noe om problemløsning knyttet til programmering i småskolen, 1.-4. klasse (eller tilsvarende i andre land, K-4).	Alt som ikke er relatert til programmering og læring i småskolen og barneskolen.
Språk	Norsk, svensk, dansk, engelsk	Alt annet.
Søkeord	«Computational thinking» + «problem solving» + «primary school» OR preschool OR «early primary school»+ programming «problemløsning» + «programmering» OR «koding» + «småskole» OR «førskole»	
Metode	Kvalitative metoder. Kvantitative metoder.	Ingen

Tabell 1: Søketabell for litteratursøk

Det finnes mye internasjonal forskning på programmering i skolen, så det var viktig å filtrere bort artikler som ikke passet i forhold til problemstillingen. Nøye utvalgte forskningsartikler ble lest for å belyse hva som er forsket på innenfor problemløsning og programmering i småskolen. Ovenfor vises en tabell over litteratursøket.

Etter å ha arbeidet meg gjennom forskningslitteraturen på feltet, har jeg valgt å sortere denne i fire kategorier. Alle disse feltene gir viktig bakgrunnsinformasjon til mitt eget prosjekt, og har vært med på å utforme dette. De fire feltene er: 1) Programmering og

problemløsning 2) Veilederens rolle 3) Fysiske, visuelle programmeringsobjekter 4) Kvantitativ tilnærming til programmeringsforskning.

1.4.1. Programmering og problemløsning

Programmering med fokus på problemløsning har vært forsket på i flere prosjekt. Lye & Koh (2014) publiserte en litteraturstudie der de undersøkte 27 forskningsprosjekt om *computational thinking* (CT). De tok utgangspunkt i de tre dimensjonene om CT: *computational concepts, practices and perspectives* (Brennan & Resnick, 2012). Bare åtte av 27 studier fokuserte på *computational practices* eller *perspectives*. Siden det er *computational practices* og *computational perspectives* som spesielt utfordrer elevenes problemløsningsferdigheter (se teorikapittelet om CT), er det uheldig at det ikke finnes flere studier som undersøker disse dimensjonene (Lye & Koh, 2014). Videre peker Lye & Koh (2014) på at noen av studiene som havnet under lupen i litteraturstudien deres har antydninger om at den lærende tilegner seg ferdigheter innen problemløsning av seg selv. Her trekkes Fessakis, et. al (2013) frem som et eksempel, og dette er en av forskningsartiklene som har vært med i litteraturgjennomgangen til dette masterprosjektet. Dessuten bygger mine forskningsspørsmål og problemstilling på implikasjoner og funn hos Fessakis, et al. (2013):

> (…) problem solving approaches of the children are concerned, it is interesting to investigate whether children plan their solution or employ trial and error method. Most of the children followed a step by step approach, according to the following pattern: repeat until the problem is solved[choose the right command/execute it/see the result/make corrections if needed]. This pattern is analogous to the trial and error method which utilizes a "guess and check" procedure. (Fessakis, et.al, 2013, s. 96)

Her beskriver Fessakis, et al. (2013) i hvilken grad elevene tilnærmer seg en løsning på et problem ved planlegging i forkant, eller om de bruker en prøve og feile-metode. I Fessakis sin studie brukte de fleste barna en *steg for steg*-tilnærming til problemet. Denne tilnærmingen kan beskrives ved at barna velger en input, utfører den, ser om den virker og justerer ved behov. I min egen forskning vil jeg undersøke hvordan elevene i småskolen bruker denne tilnærmingen når de programmerer, eller om de planlegger i større grad enn man forventer ved en prøve og feile-metode. Hva elevene og veilederen

gjør når virkningen av sekvensen ikke blir som forventet og elevene ikke finner en løsning, står også sentralt i min forskning.

Ifølge Lye & Koh (2014) er det nødvendig å undersøke og grundig gjennomgå antakelsen om at elevene lærer problemløsningsferdigheter av seg selv gjennom programmering. De utdyper at om ikke kognitive prosesser blir støttet og veiledet av andre eller at barna ikke får bekreftelse på valg av løsninger og utførelse av oppgaver, kan disse aktivitetene lede til et begrenset læringsutbytte eller bli *non-educative*, fordi elevene ikke aktivt reflekterer over erfaringene de har gjort seg (Lye & Koh, 2014, s. 58). Prøving og feiling brukes gjerne som metode, men kognitive tankerekker eller prosesser stimuleres ikke.

Sipitakiat & Nusen (2012) konkluderer i sin studie med 52 barn mellom 8-9 år, med at *feilsøking* er en av de viktigste aktivitetene barna kan arbeide med i programmering for å utvikle logisk tenking, problemløsning og interaksjon mellom deltakerne i programmeringsaktivitetene (Sipitakiat & Nusen, 2012). Deltakerne kan være andre barn eller voksne i konteksten. I denne studien fant forskerne også at barna viste en signifikant økning i *analytisk* tenking etter å ha gjennomført programmeringsopplegget.

Dersom fokuset flyttes mer over på aktiviteter i et naturlige kontekster i klasserommet, vil pedagoger i skolen kunne benytte seg av kunnskap som kan brukes i virkelighetsnære klasseromssituasjoner (Lye & Koh, 2014). Lye & Koh (2014) argumenterer med at mesteparten av forskningen som er gjort på feltet har informanter som enten er plukket eller hentet fra frivillige forskningsprosjekt. For denne masteroppgaven, blir det viktig å poengtere behovet for skildringer fra *computational practices* i norske klasserom. Dessuten etterlyser Lye & Koh (2014) behovet for forskning på *rich data*. «The term rich data describes the notion that qualitative data and their subsequent representation in text should reveal the complexities and the richness of what is being studied» (Given, 2008). Med *rich data* ønsker Lye & Koh (2014) at det fokuseres mer på kompleksitet og verdifulle tolkninger av programmeringsaktiviteter i klasserommet. De argumenterer med at dersom barn bruker det verbale språket når uttrykker kognitive prosesser i forbindelse med programmering, får forskeren et mer verdifullt innblikk i læringsprosessene (Lye & Koh, 2014).

Yu & Roque (2019) gjennomgikk programmeringsverktøy som var passende for barn i 7-årsalderen. Verktøyene de forsket på delte de inn håndgripelige, virtuelle og

hybridverktøy. Hybridverktøy er verktøy der barnet bruker både en virtuell, digital flate og et håndgripelig verktøy, der blant annet verktøyene LEGO WeDo og Blue-bot hører til (se kap. 1.4.3.). Yu og Roque (2019) ønsket å belyse:

1) verktøyenes design og hvilke funksjoner de tilbyr
2) hvilke *computational practices and concepts* barna kan utforske gjennom disse verktøyene
3) hvilket utvalg av aktiviteter barna kunne aktivisere seg i
4) hvilke kunnskaper og ferdigheter barna kunne lære i tillegg til computational thinking.

Det argumenteres for at alle verktøyene som er med i studien gir barna mulighet til å utforske *computational practices*. Her må det nevnes at et av verktøyene som er med i denne studien er Bee-boten, som også er et sentralt programmeringsverktøy i empirien i denne masteroppgaven. De forskjellige verktøyene og programvarene tillater elevene å gjøre feil, og begrenser ikke barnas interaksjoner med verktøyet, i form av et gitt antall forsøk eller handlinger (Yu & Roque, 2019). Testing og feilsøking er også nevnt som en *computational practice*, og dette er prosesser som er gjennomførbare i flesteparten av verktøyene som er med i studien. Samtidig trekkes det frem at «step-by-step instructions or specific challenges can help to scaffold children's learning experiences, more open-ended systems can enable children to express ideas, develop their own goals, and pursue more personally meaningful prosjects» (Yu & Roque, 2019, s. 11). På den måten ser vi fordeler og ulemper med oppskriftspregede oppgaver og instruksjoner. Oppskriftspregede oppgaver og verktøy støtter elevene som et stillas. På den andre siden utfordrer åpne utfordringer, eller *open-ended systems*, barna til kreativ og meningsfull problemløsning. Yu & Roque (2019) konkluderer:

> kits often use blocks or puzzle-pieces to represent code that controls robots or virtual sprites. The most common computational concepts that children could explore are sequences, loops, data, and conditionals. (…) Some (kits) are constrained to specific challenges such as moving a robot or sprite through a maze, while some provide some scaffolds such as a narrative to structure the coding experience. Relatively few kits enable more open-ended engagement and creation. (Yu & Roque, 2019, s. 15)

For denne masteroppgavens del er det verdt å merke seg at det konkluderes med at det brukes blokker eller puslebiter som skal representere koder som styrer verktøyet. I tillegg fant Yu og Roque at i flere av verktøyene struktureres programmeringsaktivitetene i form av en fortelling eller en labyrint. Da blir målet klart definert for barna; de skal komme seg ut av labyrinten. Verktøy som er *åpne*, vil utfordre elevene kreativt og til selv å finne mening med programmeringen sin. Det poengteres at det er få verktøy som er designet slik. LEGO WeDo er ikke tatt med hos Yu & Roque (2019).

1.4.2. Veilederens rolle

Fordi problemstillingen i dette prosjektet skal belyse frasen «sammen med andre» i et småskoleperspektiv, er det naturlig å se etter forskning der veilederen (enten det er lærer, assistent, medelever, osv.) kommer i fokus. I flere forskningsartikler (Fessakis, et al., 2013; Lye & Koh, 2014; Sipitakiat & Nusen, 2012) settes det fokus på lærerens eller den voksnes rolle når barn programmerer. En av konklusjonene i casestudien til Fessakis, et al. (2013) er nettopp lærerens interaksjoner og samhandling mellom elever, elevgruppe og voksne når barna programmerer. Relasjon, samspill og interaksjoner mellom barn-barn og barn-voksne har stor betydning for læringssituasjonen, også når det gjelder hvordan barna tilegner seg problemløsningsferdigheter gjennom programmering. Læreren oppmuntrer og motiverer, samtidig som hun støtter og veileder når barna står i et problem de har vansker med. De trekker også frem viktigheten av lærerens ferdigheter og evne til klasseledelse når barn arbeider med problemløsning (Fessakis, et al.,2013). Organisering av elevene, klasserommet og læringsmiljøet påvirker hvordan elevene arbeider med programmering.

Sipitakiat & Nusen (2012) er også opptatt av den voksnes rolle når barn programmerer. I deres funn, blir det klart at *early primary children* (småskoleelever) gjerne mister konsentrasjon, oppmerksomhet og fokus dersom de står fast i et problem og ikke kommer noen vei i problemene sine. På grunn av dette, blir den voksnes støtte og veiledning uvurderlig når barnet skal utvikle problemløsningsstrategier gjennom programmering. (Sipitakiat & Nusen, 2012).

1.4.3. Fysiske, visuelle programmeringsobjekter

Palmèr (2017) gjennomførte en studie der målet var å undersøke programmering som en del av matematikkopplæringen for førskolebarn. Forskningen er relevant for denne masteroppgaven fordi flere av funnene og drøftingene handler om elementer som

knyttes til problemløsningsferdigheter. Programmeringsverktøyet som ble brukt var Bee-bot (figur 2), som er et *fysisk programmeringsobjekt*, dvs. at barna ikke nødvendigvis trenger en digital flate å programmere gjennom. Bee-boten er en robot, en bie, utformet med et design som appellerer til barn. Den har klare, skarpe farger og andre visuelle uttrykk som fanger barnas oppmerksomhet og interesse.

Figur 2: Bee-bot. Bildet hentet fra produktutvikler (www.bee-bot.us)

Figur 3: Blue-bot. Bildet hentet fra produktutvikler (www.bee-bot.us)

Barna programmerer ved hjelp av knapper på toppen av Bee-boten. Blue-bot (figur 2) er en videreutvikling av Bee-boten, og Blue-bot gir mulighet for å programmere via en app. Da kan barna se og følge med på sekvensene sine etter hvert som det fysiske verktøyet forflytter seg. Kelleher & Pausch (2005) fant at fysiske programmeringsobjekter gjorde det mulig for barn å kunne fokusere på logisk resonnering og strukturer i sekvensene sine. En sekvens er beskrevet som «Like a recipe, a sequence of programming instructions specifies the behavior or action that should be produced» (Brennan & Resnick, 2012, s. 3).

Det finnes studier som konkluderer med at fysiske programmeringsobjekt gjør problemløsningsprosesser mer håndgripelig for barn (Fessakis, et.al, 2013). I Palmèr

(2017) sin studie designet barna egne kart. Deretter skulle Bee-boten programmeres til å følge kartet som barnet hadde produsert. Barna brukte et verbalt språk for å uttrykke inputene og sekvensene de satte sammen, og det bør nevnes at studien hadde en sosiokulturell tilnærming. Input er definert som informasjon eller data som er sendt til en computer for prosessering (Computer Hope, 2018). Palmèr (2017) fant at alle barna i studien utførte oppgaven med å produsere et eget kart, og hun hevder at barna viste evner til å dele opp oppgaven sin i flere deloppgaver, *decomposition*.

I 2017 utførte Di Lieto, et al. (2017) en studie med mål om å undersøke kortsiktige virkninger av det de definerte som ER-økter (resultatene fra studien blir kort oppsummert i avsnittet nedenfor). «Educational Robotic (ER) is a broad term used to indicate a branch of knowledge requiring students to program robots actions or even to design, create and assembly them» (Di Lieto, et al., 2017). Ut ifra denne definisjonen kan vi forstå ER som et paraplybegrep for kunnskaper og ferdigheter for å designe og programmere roboter. Bee-bot (som nevnt over) er et eksempel på en slik robot. Di Lieto, et al. (2017) trekker frem STEM-fag som fagområder der programmering skal ha vist en *læringseffekt*, og bruker andre studier for å underbygge påstanden. I Norge brukes begrepet realfag om STEM-fag (fagene som tilhører STEM-begrepet er *science, technology, engineering og mathematics)*. Bers, Flannery, Kazakoff & Sullivan (2014) utførte og forsket på et ER-kurs for som fikk navnet «The Tangible K Robotics Program». I den studien ble en av konklusjonene at førskolebarns læringsforutsetninger generelt kan bli styrket gjennom programmering, i tillegg til at deres *computational thinking* ble forbedret etter at de hadde gjennomført et konstruert opplegg designet for studiet (Bers, et al., 2014). Innholdet i opplegget handlet blant annet om kreativt robot-design og løsninger av feil i programmer (feilsøking og testing) og sekvenstenking.

Jeg velger å introdusere LEGO WeDo-konseptet under overskrifte fysiske programmeringsverktøy, *tangible robots* (Beers et al., 2014), fordi dette verktøyet var programmeringsverktøyet som ble brukt når data fra feltet skulle samles i denne masteroppgaven. LEGO WeDo 2.0. blir levert som verktøyskrin, der LEGO-brikkene ligger sortert. Sammen med skrinet bruker elevene en app, der programmeringen foregår. Her ligger også oppskrifter for ulike produkter barna kan designe.

Figur 4: Innhold i LEGO WeDo. Appen gir også tilgang til oppskrifter på ulike produkter som kan designes. Bildet er hentet fra produsent: https://education.LEGO.com/en-us/elementary/shop/wedo-2

1.4.4. Kvantitativ tilnærming til programmeringsforskning

I nesten alle studiene som er nevnt over, har forskningsdesignet og metodene vært knyttet til kvalitative data. Den eneste studien som er presentert med kvantitative data, er studien til Di Lieto et al. (2017). Det er valgt å ta med denne kvantitative forskningsartikkelen ut ifra etiske retningslinjer for forskning (NESH) og forskerens redelighet. Det forsøkes ikke å «pynte på resultatet» ved å la være å ta med slike studier som den som er nevnt her. Artikkelen kan ikke utelates fordi den retter kritikk mot kvalitativ tilnærming på programmeringsundervisning i skolen. Når det er sagt foreslås det implikasjoner om *rich data* i andre artikler (Lye & Koh, 2014), og der kan kvalitative tilnærminger være passende fordi man søker konstruksjon av forståelse og kunnskap nært på informantene. Uansett er Di Lieto et al. (2017) kritiske til overvekten av studier med kvalitative metoder når programmeringsfenomenet skulle forskes på i skolesammenheng. Dessuten påpekte de at kontrollgrupper ofte manglet. Derfor er deres studie planlagt og gjennomført med evidensbasert tilnærming og kvantitative data. Målet var å studere effekten av kortvarige ER-økter, og man ønsket å undersøke eventuell endring i kognitive funksjoner hos førskolebarn etter hvert som barna arbeidet seg gjennom ER-øktene. Flere yrkesgrupper var satt inn for å overvåke elevaktiviteten i studien, og de hentet inn både lærere, ingeniører, nevropsykologer og psykologer. Resultatene fra studien viser en signifikant forbedring av visuell-spatiale arbeidsminnet og evnen til å sortere informasjon etter hva som er viktig og nødvendig for å løse

oppgaven (Di Lieto et al., 2017). Dessuten konkluderer de med at ER-øktene må være korte og intensive.

Implikasjoner fra tidligere forskning

Den forskningen som er presentert over er hentet fra internasjonale studier. Det er naturlig å tenke at rammefaktorer, blant digital infrastruktur, bruk av digitale verktøy og utforming av læringsareal og klasserom i Norge skiller seg fra skoler i andre land. I arbeidet mot ny læreplan (2020), har algoritmisk tenking og programmering fått en rolle å spille. Selv om forskningen som er presentert over er hentet fra førskole og småskoletrinn, er det behov for forskning som forteller noe om hvordan programmering foregår i norske småskoleklasser.

Programmering som fenomen er godt dokumentert, men mye av forskningen er begrenset til *computational concepts* (Lye & Koh, 2014), altså begreper knyttet til programmering. Siden det nettopp er *computational practices og perspectives* som utvikler elevenes problemløsningsferdigheter, er det nødvendig at disse blir satt mer i fokus når det snakkes om problemløsning og programmering. Over har jeg også pekt på implikasjoner fra Lye & Koh (2014) der det hevdes at det finnes en antakelse om at problemløsningsferdigheter utvikles «av seg selv». Derfor er det behov for og fornuftig å fokusere på dette i norske klasserom, og særlig småskoleklasserom. I litteratursøket mitt fant jeg lite norsk forskning som handlet om programmering i småskolen.

Tidligere forskning (Fessakis, et al., 2013) bekrefter at det må fokuseres mer på den voksnes støtte og elevens samspill med voksne i programmeringsaktiviteter. Da blir det fornuftig og hensiktsmessig å forske på programmering i klasserommet ut fra en tilnærming der man kan forstå barnets interaksjoner med andre barn og voksne, og også interaksjoner mellom barn og verktøy. En sosiokulturell tilnærming vil derfor være passende. Som nevnt peker Lye & Koh (2014) på at det er få studier som har fokusert på *computational practices and perspectives*, der testing og feilsøkingsaktiviteter står sentralt. I slike aktiviteter er det interessant å se på interaksjoner mellom barn-barn og verktøy-barn. En sosiokulturell tilnærming muliggjør studier av disse interaksjonene, selv om fenomenet har lange tradisjoner innen konstruksjonisme som læringsteori (Papert & Harel, 1991). Det påpeker også Lye & Koh (2014).

2. Teori knyttet til programmering og problemløsning i skolen

2.1. Sosiokulturell læringsteori

2.1.1. Vygotskys syn på barns utvikling og den proksimale utviklingssonen

Ut fra et sosiokulturelt syn på læring står interaksjoner, samarbeid og samspill med andre sentralt for barns kognitive utvikling. Den russiske psykologen Lev Vygotsky er en av teoretikerne som står sentralt bak noen av grunntankene som ligger i de sosiokulturelle læringsteoriene. Vygotsky utviklet en teori om kulturell utvikling. Den går ut på at et barn først mestrer sammen med andre gjennom interaksjoner, og deretter vil barnet mestre alene på et indre plan.

> Every function in the child`s cultural development appears twice: first, on the social level, and later, on the individual level; first, between people (interpsychological), and then inside the child (intrapsychological). This applies equally to voluntary attention, to logical memory, and to the formation of concepts. (Vygotsky, 1978, s. 57)

Dermed blir det lettere å se hvordan språk og interaksjoner blir avgjørende for læring. Først skal barnet mestre f.eks. begreper sammen med andre, deretter skal barnet mestre det for seg selv, på et indre plan. Vygotsky hevdet at språket blir et artefakt for læring. Dersom elevene skal lære noe, må de først bruke *språket* sammen med andre (i en kultur), som blir en sosial prosess. Deretter skjer det en indre prosess, med en indre tale.

I *Mind in Society* (1978) beskriver Vygotsky teorien om den nærmeste utviklingssone, *the zone of proximal development.* Denne teorien kan brukes som et bakteppe for å forstå barns utvikling, sett gjennom et sosiokulturelt perspektiv på læring. Vygotsky skiller mellom barnets nåværende utviklingsnivå, og barnets potensielle utviklingsnivå. Det potensielle utviklingsnivået kan barnet nå ved hjelp av støtte, hjelp og veiledning fra mer *capable peers.*

> the distance between actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers. (Vygotsky, 1978, s. 86)

Her ser vi at Vygotsky skiller mellom problemløsning som barnet klarer alene og uten støtte, og problemløsningsoppgaver der barnet gjennomfører oppgaven med veiledning

fra voksne eller i en samarbeidssituasjon. I samarbeidet må gruppemedlemmene være mer kapable enn barnet selv, for at barnet skal kunne strekke seg inn i den proksimale utviklingssonen. Denne teorien blir ofte visualisert med tre sirkler, der det lærende barnet befinner seg i sentrum av den innerste sirkelen. I sirkelen nærmest barnet ligger de oppgavene barnet klarer selv. Den ytterste sirkelen skal illustrere barnets proksimale utviklingssone, der barnet får hjelp/støtte fra voksne eller samarbeider med andre.

Mestring og appropriering

Det er ikke alle som har vært fullstendig enig i Vygotskys teorier, og det finnes flere eksempler på at hans teorier er videreutviklet. En av disse er James Wertsch. I sammenheng med den proksimale utviklingssonen løfter Wertsch (1998) frem begrepene *mastery and appropriation.* Å mestre noe, *mastery*, blir i denne sammenhengen synonymt med å kunne noe, i dette prosjektet blir det nærliggende å tenke på å *mestre et verktøy* som eksempel på mestring. For at barna kan kunne utvikle seg kognitivt er appropriering nødvendig. Det betyr at lærestoffet, det barnet skal mestre, må bli en del av barnet selv, barnet må gjøre det til sitt eget.

I dette forskningsprosjektet kan barna f.eks. mestre et programmeringsverktøy, de vet gjerne hvilke funksjoner knappene på Bee-boten har og kan sette sammen tilfeldige sekvenser eller kjeder av kommandoer. Om et barn som programmerer kopierer en mer kompetent medelev sine handlinger, kan barnet mestre tekniske ferdigheter i et program (f.eks. å kjenne igjen forskjellige fargegrupper for koder i et verktøy). Dette vil ikke føre til appropriering. Dersom barnet i stedet klarer å sette sammen kommandoer og sekvenser for å produsere egne meningsfulle program, avanserte kjeder av handlinger, vil dette føre til kognitiv appropriering.

2.1.2. Scaffolding

En annen forsker som har forsøkt å nyansere Vygotskys ideer er sosialpsykologen Jerome Bruner. Bruner er mest kjent for sitt begrep *scaffolding*, en videreføring av teorien om den proksimale utviklingssone. Med *scaffolding* refererer Bruner til apparatet rundt læringsprosessen. Wood, Bruner & Ross (1976) beskriver hvordan begrepet *scaffolding*, eller stillasbygging passer inn i et sosiokulturelt syn på læring. De tar utgangspunkt i et barn eller en novise som skal løse et problem, nå et mål eller gjennomføre en oppgave. Den voksne eller mer kompetente vil kontrollere og avgrense elementene i oppgaven som er utenfor den lærende sin læringskapasitet eller forutsetninger for å lære. Det vil føre til at den lærende kan konsentrere seg om de

elementene som er innenfor elevens muligheter for å lære (Wood, Bruner & Ross, 1976, s. 90). Den voksnes oppgave blir altså å legge til rette for læring, ved å luke vekk elementer fra oppgaven som blir for vanskelige for barnet.

Stillasbygging er en sosial prosess, der læreren støtter elevene emosjonelt og kognitivt (Meyer & Turner, 2002). Gjennom stillasbygging har læreren mulighet til å støtte elevens motivasjon og selvregulering på tre måter; gjennom kompetansebygging av elevens forståelse, gjennom å støtte elevens emosjonelle behov og samtidig engasjere eleven til læring, og gjennom å trene opp elevens selvstendighet/autonomi. Det er viktig at *the scaffolder and the scaffoldee* har en felles forståelse av hva som vil være en vel utført oppgave, «an understanding of what successful performance of the target task would look like that was shared between the scaffolder and the scaffoldee» (Belland, 2017, s. 18; Wood, et al, 1976). Da har eleven mulighet til selv å forstå når oppgaven er løst på en tilfredsstillende måte, og forutsetninger for å klare en tilsvarende oppgave selv i fremtiden.

Wood, Bruner & Ross (1976) gjennomførte en undersøkelse med førskolebarn, der de skulle bruke byggeklosser og bygge en pyramide. Barna var mellom 3 og 5 år gamle. I etterkant av undersøkelsen skisserte de en instruksjonsprosess, som tar utgangspunkt i teorien om scaffolding. Stillasets (lærerens) oppgave er å fange elevens oppmerksomhet og «verve» eleven til oppgaven. Deretter skal læreren gi eleven en «reduction in degrees of freedom» (Wood, et al., 1976, s. 98). Det innebærer at problemet må forenkles og konkretiseres, slik at oppgaven er gjennomførbar for eleven. Videre skal lærerens rolle være å holde eleven på rett vei, *direction maintenance*. Her handler det delvis om å motivere eleven til å holde frem og fokusere på oppgaven, men også at en veileder støtter og synliggjør veien videre i problemet, slik at eleven opplever en verdifull mening med å holde kontinuitet i oppgaven eller risikerer et steg videre i problemløsningsprosessen (Wood, et al, 1976, s.98). Her kan vi si at elevene må trenes på å gå ut av komfortsonen sin, at de ikke blir værende i en «behagelig» del av oppgaven der de ikke kommer videre mot en løsning. Veilederens rolle blir så å fremheve kritiske deler av valgene som den lærende tar. Det er avgjørende for en god løsning, der læreren, veilederen eller stillaset tolker avvik som kan danne misoppfatninger eller ikke føre frem til en god løsning. Det er viktig at veilederen/læreren er bevisst på at eleven ikke blir for avhengig av tilbakemeldinger,

støtte og ros fra stillaset. Samtidig skal stillaset være en plass der man får støtte når man har behov for det. Det siste momentet i denne instruksjonsprosessen er *demonstrasjon*:

> Demonstrating or modelling solutions to a task, when closely observed, involves considerably more than simply performing in the presence of the tutee. It often involves an "idealization" of the act to be performed and it may involve completion or even explication of a solution already partially executed by the tutee himself. (Wood, et al., 1976, s. 98)

Wood et al. (1976) argumenterer altså for at dersom stillaset modellerer en løsning, er hensikten at den lærende vil etterligne løsningen som er modellert i en mer hensiktsmessig og passende form. Å modellere vil ikke bare si å gjennomføre oppgaven når den lærende ser på, men i tillegg kan løsningen og utførelsen som modelleres og observeres av den lærende, være delvis utført av den lærende allerede.

2.1.3. Medierende artefakter

Säljö er også en av de som arbeider videre i tradisjonen etter Vygotsky (Säljö & Moen, 2001). Säljö argumenterer for mennesker er skapt for samspill med andre. Vi beskriver, observerer og tolker våre medmennesker, slik at de hjelper oss til å forstå verden (Säljö & Moen, 2001). Fysiske, psykologiske eller intellektuelle redskaper som hjelper oss med denne forståelsen kalles *medierende artefakter.* I et klasseromsperspektiv kan medierende artefakter være alt fra hjelp, støtte og veiledning fra andre elever eller voksne, digitale hjelpemidler, læringsressurser og kalkulatorer, til innhold i pennal eller plakater og plansjer på veggene (Säljö & Moen, 2001).

Sentrale artefakter i en programmeringssammenheng i klasserommet er interaksjoner og språk. Samspill mellom elevene i samarbeid og hjelp, støtte og veiledning fra voksne er naturlig å fokusere på. Dessuten blir selve programmeringsverktøyet et viktig artefakt. Verktøyet gir en tilbakemelding til den lærende på om og hvordan problemet er løst, eller om det finnes feil i programmet som må løses. I programmeringsaktiviteter hvor det benyttes *oppskrifter (*f.eks. LEGO WeDo), vil også oppskriftene være et sentralt artefakt.

2.2. Problemløsning

Når problemstillingen i denne masteroppgaven handler om hvordan barn utvikler problemløsningsferdigheter gjennom programmering, blir det viktig å definere

problemløsning. Problemløsning er et begrep som ofte blir knyttet til utviklingen av *computational thinking* (Fessakis, et al., 2013; Kafai, et.al, 2014; Wing, 2006). I skolesammenheng er også problemløsning en ferdighet som har fått stor betydning, blant annet i Ludvigsentuvalget (NOU 2015:8, 2015) og i 21th Century skills (Ananiadou & Claro, 2009; Dede, 2009, s. 3). Dede (2009) argumenter for at måten man tenkte problemløsning i skolen i *20th century instructions*, ikke ville gi en god overføringsverdi til *real world situations*. Det er fordi man tenkte på problemløsning som et abstrakt fenomen «removed from their application to knowledge» (Dede, 2009, s.3). Dede (2009) hevdet derfor at den ultimate tilnærmingen til å lære elevene problemløsningsferdigheter, er at de lærer en *problemløsningsrutine*.

Begrepet problemløsning er et komplekst begrep, og det finnes flere definisjoner som forsøker å forklare hva problemløsning er (Mayer & Wittrock, 2006; Jonassen, 2010). Kim & Hannafin (2011) definerer problemløsning som «situated, deliberate, learner-directed, activity-oriented efforts to seek divergent solutions to authentic problems through multiple interactions amongst problem solver, tools, and other resources» (Kim & Hannafin, 2011, s. 405). Ved å vektlegge elevaktivitet og interaksjoner mellom problemløser og artefakter, passer disse elementene godt inn i et sosiokulturelt perspektiv.

I mitt forskningsprosjekt har jeg valgt en modell som lar seg koble sammen med og kan settes i sammenheng med sosiokulturell læringsteori og *computational thinking*, og som stemmer med definisjonen til Kim & Hannafin (2011). Med tanke på at problemstillingen min fokuserer på småskolebarn, må jeg også finne en måte å forstå problemløsning på som passer inn i metodene og arbeidsmåtene som man kan finne i småskolen.

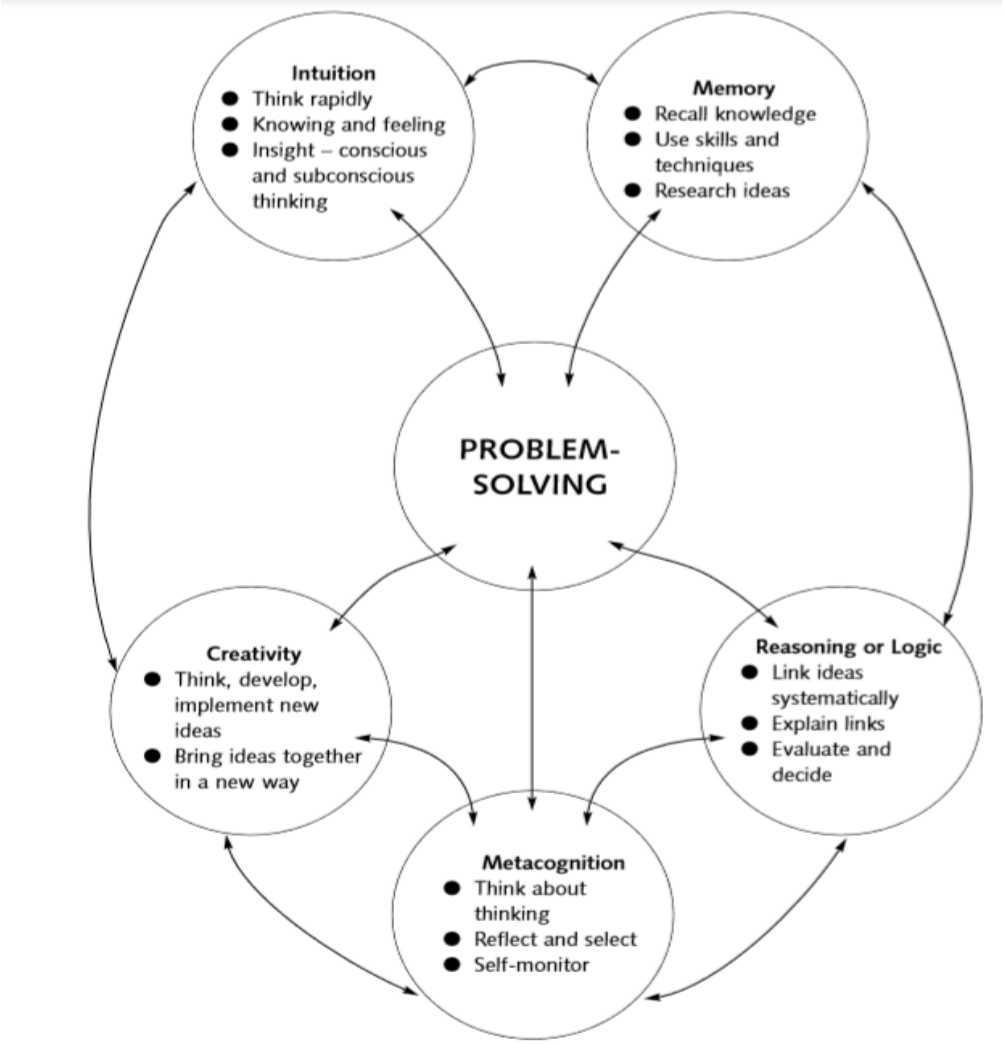
2.2.1. Wallace & Makers problemløsningsmodell

De sosiokulturelle teoriene som er nevnt over, er sentrert rundt at barn lærer i interaksjon med voksne, andre barn og gjennom, med og av artefakter. I 1993 publiserte Wallace & Adams et rammeverk for å forstå problemløsning i en sosial kontekst (Wallace & Adams, 1993). Dette rammeverket fikk navnet TASC, *Thinking Actively in a Social Context.* Flere år senere ble dette rammeverket revidert og nyansert av Wallace & Maker (Wallace, B., Maker, J., Cave, D., & Chandler, S., 2005). De var blant annet påvirket av Vygotsky, Bruner og Bandura (Wallace, et al., 2005, s. 31-33). Særlig har et

sett av *vital learning capacities* fått påvirke TASC-modellen. Resultatet ble en ny utgave av TASC-modellen, som heretter vil bli omtalt som Wallace & Makers problemløsningshjul (TASC *Problem Solving Wheel*). I avsnittene nedenfor vil først de vitale læringskapasitetene bli presentert, deretter problemløsningshjulet og mot slutten av delkapittelet vil en vurdering av modellen bli gitt. Vurderingen er basert på forskningsartikler som har brukt Wallace et al. (2005) som teoretisk rammeverk. Det er nødvendig å argumentere for modellens gyldighet, i og med at modellen ikke hører til de store og anerkjente læringsteoriene og teoretikerne.

Vital learning capacities

Maker og Wallace har presentert en oversikt (Wallace, et al., 2005) som forklarer hvordan ulike ferdigheter (de bruker begrepet *kapasiteter* om disse ferdighetene) spiller inn på barns evne til problemløsning. Ferdighetene er gjensidig avhengig av hverandre,



Figur 5: Vital learning capacities

og står i et interaktivt og dynamisk forhold til hverandre i problemløsningsprosessen. De ulike ferdighetene eller kapasitetene som kategoriseres i modellen er intuisjon, minne, kreativitet, metakognisjon og argumentasjon eller logikk (Wallace, et al., 2005). Se Figur 5.

Intuisjon

Intuisjon er evnen til å umiddelbart vite noe. Tenkingen går hurtig for seg, og kan være logisk og kreativ; «rapid understanding and insight that uses elements of both subconscious and conscious thinking» (Wallace, et al., 2005, s. 6). Wallace, et al. (2005) hevder at barn har en betydelig kapasitet for intuitiv forståelse, de både forstår og føler før de utvikler språket som seinere vil uttrykke kunnskap og følelser. Her ser vi hvordan denne delen av modellen kan knyttes opp mot sosiokulturell læringsteori. I følge Vygotsky utvikler barn først det indre språket og deretter det ytre språket. Wallace, et al. (2005) formulerer også disse argumentene i fremstillingen sin, her ved referanse til et første og andre språk: «We could refer to their intuitive understanding as their 'first language' and call the sounds and symbols they learn to express ideas their 'second language'» (Wallace, et al., 2005, s. 6). For dette forskningsprosjektet sin del, er det verdt å merke seg at Wallace, et al. (2005) påpeker at mange barn prosesserer den intuitive kunnskapen så hurtig at de får problemer med å formulere sine tanker og argumenter for andre.

Minne

Den neste kategorien i modellen er *minne*. Wallace, et al. (2005) argumenterer med at «everyone (…) needs knowledge that is specific to a particular ability,for example wide language vocabulary, maths sequences, decoding and spelling(…)»(Wallace, et al., 2005). De viderefører dette argumentet med at når barn jobber med problemløsning, trenger de både erfarings- og kompetansespesifikk kunnskap. De hevder tilslutt at dersom læringen skjer i en virkelighetsnær kontekst (*real life-context*) har barnet stor minnekapasitet.

Kreativitet

Den neste kategorien av dynamiske kapasiteter er kreativitet. *Kreativitet* er en essensiell del av problemløsning, og blir her definert som «general capacity to think of, develop,and implement new or appropriate ideas and solutions,or to bring unusual ideas together in a new way» (Wallace, et al, 2005, s. 6).

Argumentasjon eller logikk

Argumentasjon eller logikk blir i denne sammenhengen definert som kapasitet til å lenke ideer sammen systematisk, der man kan forklare eller rettferdiggjøre valgene man tar (Wallace, et al., 2005, s.6). I sammenheng med problemløsning blir det nødvendig med argumentasjon der barnet skal vurdere valg, ideer og løsninger. Det er viktig at man skiller mellom ulike typer av logikk og argumentasjon. For å løse et matematisk problem trenger vi en annen fremgangsmåte enn dersom man skal rettferdiggjøre og argumentere for en påstand om en person i en fiktiv novelle.

Metakognisjon

Til slutt i denne modellen ligger metakognisjon, som «uses images, words and thoughts and constitutes the important overall 'monitor' of the use of human capacities» (Wallace, et al., 2005, s. 7). Ifølge Wallace, et al. (2005) involverer metakognisjon selve problemløsningsprosessene, og metakognitive refleksjoner gjør barnet i stand til å vurdere og velge ut hvilke av de nevnte kapasitetene som er passende å bruke i en pågående problemløsningsprosess: bruke erfarings- eller kompetansebasert kunnskap, stole på sin intuisjon, tenke kreativt, logisk osv.

Problem solving wheel

Problemløsningshjulet til Wallace, et al. (2005) er delt inn i åtte stadier, men å følge stadiene slavisk er ikke alltid hensiktsmessig; «sometimes going back to a stage to clarify the initial thinking is necessary. Learners also need to verbalise the stages of their thinking in order to highlight and crystallise the skills they are practising» (Wallace, et al., s. 37). I likhet med ferdighetene presentert i forrige avsnitt, ser vi at dette hjulet også kan fungere dynamisk. Problemløsningshjulet er delt inn slik (med mine norske oversettelser):

1. Samle og organisere

   Her finner den lærende ut hva han vet om problemet fra før, og systematiserer førkunnskaper. Wallace, et al. (2005) påpeker at på dette stadiet er bruk av og forbedring av minnet en viktig del av problemløsningsprosessen.

2. Identifisere problemet

   Å tydeliggjøre problemet krevet både analytisk og kreativ tenking (Wallace, et al., 2005). Mange elever (og voksne) mister oversikten, tar inn for mange og unødvendige detaljer eller blir distrahert av uviktige detaljer når de skal løse et

problem. Aktuelle spørsmål å stille seg i denne prosessen kan være: Hva må vi gjøre? Hva prøver vi å gjøre? Hvorfor vil ikke dette skje?

3. Generere ideer

   Dette stadiet krever et høyt nivå av kreativ tenking. Ofte velger den lærende den første ideen som dukker opp, og velger derfor en enkel løsning. Her blir det viktig å trene elevene på utholdenhet. Å stå i et problem til de får tenkt seg godt om blir nødvendig: Hvor mange mulige løsninger finner du på problemet? Hvordan kan vi finne alternative løsninger?

4. Beslutningstaking

   Dominerende tenkeverktøy på dette stadiet er analytiske og det er viktig med evne til å evaluere en løsning (Wallace, et al., 2005). Når man foretar beslutninger, må man planlegge, prioritere og grunngi valg man gjør, i stedet for å foreta impulsive beslutninger. Samtidig mener Wallace, et al. (2005) at mulige løsninger må bli prøvd ut.

5. Implementering

   Når elevene befinner seg på dette stadiet, mener Wallace, et al. (2005) at den lærende må stille seg spørsmål om løsningen fungerer slik man ønsket, om man burde gjøre noen endringer, eller hva man skal gjøre etterpå. Ved å stille disse spørsmålene, ser det ut til at den lærende *overvåker* løsningen og *vurderer* endringer underveis.

6. Evaluering

   I denne modellen handler evalueringsfasen om å se seg tilbake, og vurdere om man har utført oppgaven på ordentlig vis. Den lærende skal vurdere om han eller hun er fornøyd med løsningen og hvordan man kunne funnet en enda bedre løsning.

7. Kommunikasjon

   Å fortelle andre om arbeidet og løsningene man har valgt skal føles meningsfullt for den lærende. Derfor er det nødvendig med et «ekte publikum» for å fortelle og dele ideer og resultater med andre (Wallace, et al., 2005). Et godt eksempel å stille seg i denne delen av prosessen er: Hvordan kan jeg forklare løsningen/resultatet for andre?

Frasen «sammen med andre» er brukt i både problemstilling og i et av forskningsspørsmålene. Derfor må teorivalget kunne brukes i et sosiokulturelt perspektiv på læring. Modellen til Wallace & Maker (Wallace et al., 2005) har vært brukt som teorigrunnlag i flere forskningsartikler, blant annet hos Clavio & Fajardo (2008). Denne studien hadde som formål å undersøke leker brukt som «instructional tools in the development of problem-solving skills among preschoolers» (Clavio & Fajardo, 2008, s. 87). Her ble modellen til Wallace & Maker (Wallace et al., 2005) brukt som et grunnlag for å designe læringsøkter for førskolebarn. Forskerne konkluderer blant annet studien støtter «and agrees with the effectiveness of Wallace and Maker's instructional design in developing problem-solving skills among young learners» (Clavio & Fajardo, 2008, s. 98). Teorien om problemløsning i en sosial kontekst som her er presentert, er ikke spesifikk for programmering i skolen.

2.3. Computational thinking

Masteroppgavens problemstilling retter søkelys på programmering og problemløsning:

Problemstilling: Hva kjennetegner programmeringsøkter i småskolen der barn får utvikle problemløsningsferdigheter sammen med andre?

F1*: Hva mener lærere fremmer og hemmer småskoleelevers muligheter for å utvikle problemløsningsferdigheter sammen med andre når de jobber med programmering?*

F2*: Hva kjennetegner situasjoner der barn får mulighet til utvikling av problemløsning når de jobber med testing og feilsøking?*

Teorien om problemløsning som er presentert over, kan anses som teori som passer inn i flere sammenhenger. Den er ikke spesifikk rettet mot programmeringsøkter i skolen. Derfor er det viktig å trekke frem spesifikk teori om programmering og *computational thinking.*

Brennan and Resnick (2012) har formulert et rammeverk bestående av tre dimensjoner for å forstå begrepet *computational thinking* (CT). De har tatt utgangspunkt i programmeringsaktiviteter for barn og unge, og bygget opp dimensjonene sine utfra det. De tre dimensjonene for å forstå CT er *computational concepts, practices* og *perspectives*. Ifølge Lye & Koh (2014) er disse dimensjonene nyttige for å forstå

hvordan grunnskoleelever (k-12) tilnærmer seg programmering og *computational thinking*.

Computational concepts handler om begreper som er knyttet til programmering. Brennan & Resnick (2012) tok utgangspunkt i det visuelle blokkbaserte programmeringsspråket Scratch når de utviklet dimensjonene. Samtidig hevdet de at begrepene de tar utgangspunkt i kan overføres «to other programming (and non-programming) contexts: sequences, loops, parallelism, events, conditionals, operators, and data» (Brennan & Resnick, 2012, s. 3). Her vises det også til at disse begrepene kan gjenkjennes i andre programmeringsspråk. I mitt forskningsspørsmål F1 har jeg tatt utgangspunkt i begrepene testing og feilsøking, begrep som kategoriseres som *computational concepts*.

Computational practices sier noe om prosessene som ligger bak programmeringsbegrepene når barn programmerer. Strategier og prosesser for læring og tenking står sentralt her. Brennan og Resnick (2012) hevdet at de kunne trekke frem fire praksiser som gikk igjen når barn designet interaktive medier: «being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing» (Brennan & Resnick, 2012, s. 7). Det er denne dimensjonen som står sentralt når jeg skal drøfte det andre forskningsspørsmålet mitt, jeg har fokus på praksiser i klasserommet som passer inn i modellen jeg har presentert om problemløsning. Lye & Koh (2014) gjengir og kategoriserer nemlig denne dimensjonen som problemløsende praksiser som oppstår under programmeringen. Begrepene feilsøking (*debugging)* og testing er trukket frem som slike problemløsende praksiser. Det er fornuftig å se på hvordan elevene tester programmene sine, hva som skjer når de oppdager en feil eller at programmet ikke responderer slik barna ønsker. Samtidig kan *being incremental* oversettes til å kunne bryte ned et problem til mindre småproblemer, og dermed kan man undersøke om barna evner å gjøre det i sine programmeringsaktiviteter.

Computational perspectives handler om elevenes forståelse av seg selv, om samspillet og interaksjonene med andre og teknologien rundt dem (Lye & Koh, 2014; Brennan & Resnick, 2012). Kjennetegn som kan observeres er hvordan elevene uttrykker seg, og stiller spørsmål om teknologien som omgir dem. Teknologien fungerer som en lærende partner, og barnet er i interaksjon med verktøyet. Begrepet *medierende artefakt* står

sentralt i sosiokulturelle læringsteorier, og programmeringsverktøyet (f.eks. Bee-bot og de tilhørende koordinatene til den) vil fungere som et slikt artefakt. Nettopp derfor synes jeg det er nyttig å forske på programmering fra et sosiokulturelt perspektiv. Interaksjonene mellom barn og verktøy blir satt i fokus.

Wing (2006) argumenterte med at dersom vi trener på å forstå programmering i datasammenheng, oppnår vi en overføringsverdi til den virkelige verden. Vi blir bedre til å forstå problemer, og ved hjelp av dataprogrammering blir vi bedre problemløsere. Computational thinking er «understood as extending computer science principles to other disciplines in order to help break down the elements of any problem» (Kafai & Burke, 2013, s. 62).

Wing (2006) argumenterer for hvordan algoritmisk tankegang er gjeldende i hverdagslivet og i barns lek. Hun tar for seg et eksempel med LEGO-brikker, og en situasjon der barn skal rydde opp og sortere disse brikkene. Da kan man tenke seg at man har et system, der «alle rektangelformede brikker skal i en boks», uavhengig eller avhengig av brikkens farge. En programmerer vil kalle denne prosessen for *hashing*, en prosess der man gjør store data om til små data. Det er nok sannsynlig i en vanlig lekesituasjon at barnet eller den voksne hadde kastet alle LEGO-brikkene i en boks. Men dersom barnet planla et større LEGO-prosjekt, der målet var å konstruere en avansert figur eller et bygg med et gitt system, sett med regler eller mønster, ville denne sorteringen vært en av de forberedende tiltakene barnet ville hatt fordel av i problemløsningen.

2.3.1. Å fikle med et problem

Det å *produsere* et produkt står sentralt i flere av de sentrale læringsteoriene som står igjen i dag. Dette finner vi igjen i sosiokulturelle læringsteorier, blant annet i Deweys *progressivisme* (Dewey, 1938), og i Seymour Papert sine formuleringer om *konstruksjonisme* (Papert & Harel, 1991). Begge disse teoriene er eksempler der den lærende får en eksperimentell, utforskende tilnærming til læring. Elevaktivitet står sentralt. *Fikling* er også nevnt i UDIR sin modell om den algoritmiske tenkeren (Figur 1). Resnick & Rosenbaum (2013) bruker begrepet *tinkering* som en «valid and valuable style of working, characterized by a playful, exploratory, iterative style of engaging with a problem or project» (Resnick & Rosenbaum, 2013, s. 164). Slik kan *tinkering* knyttes opp til computational practices, som jeg har skrevet om over.

På norsk kan *tinkering* oversettes til fikling, å fikle med et problem. Med fikling mener Resnick & Rosenbaum (2013) at den lærende har en utforskende tilnærming til å finne løsningen på problemet. Man tester stadig ut nye løsninger, justerer og eksperimenterer med nye muligheter, gjerne om og om igjen. Resnick & Rosenbaum (2013) hevder at fikling er en rotete prosess, og i artikkelen skriver forfatterne at mange mener planlegging ut fra et mål (ovenfra-ned) er en motsatt prosess av fikling. Har man et ovenfra og ned-perspektiv blir målet klart definert, og man planlegger ut ifra målet. Med en fiklende tilnærming til problemet blir ikke målet definert, og man utforsker, eksperimenterer og leker seg frem til problemet.

I følge Resnick & Rosenbaum (2013) blir fikling en særlig viktig del av den verden vi lever i, for at barna skal bli kreative problemløsere. De hevder at «success in the future will depend not on what you know, or how much you know, but on your ability to think and act creatively—on your ability to come up with innovative solutions to unexpected situations and unanticipated problems» (Resnick & Rosenbaum, 2013, s. 4). Man ønsker altså å utdanne problemløsere som kan håndtere uventede problemer med innovative og kreative løsninger. Denne måten å forstå fikling på, harmonerer med det Seymour Papert argumenterte med da han utviklet konstruksjonisme som læringsteori (Papert & Harel, 1991). Han tok utgangspunkt i at elevene arbeidet med et materiale over lang tid. Læringsaktivitetene dreide seg om design av systemer, og barna skulle arbeide, endre, rive ned og bygge opp igjen sine egne produkt.

2.4.Hvilke konsekvenser har teoriene ovenfor for mitt forskningsprosjekt?

Ut ifra et sosiokulturelt læringsperspektiv er lærerens rolle vesentlig for hvordan elevene utvikler problemløsningsferdigheter. Den voksne skal motivere og veilede for at eleven skal komme videre på veien mot en løsning på problemet. Det finnes flere måter å hjelpe eleven på, jf. Wood, et.al (1976). I intervjuguiden (vedlegg 1) og observasjonsnotatene (vedlegg 6) for dette forskningsprosjektet har lærerens rolle i disse situasjonene fått en stor plass. Hvilke spørsmål stilles, hvordan tilpasses oppgaven elevens motivasjon og evne til problemløsning er interessante momenter for å finne svar på problemstillingen.

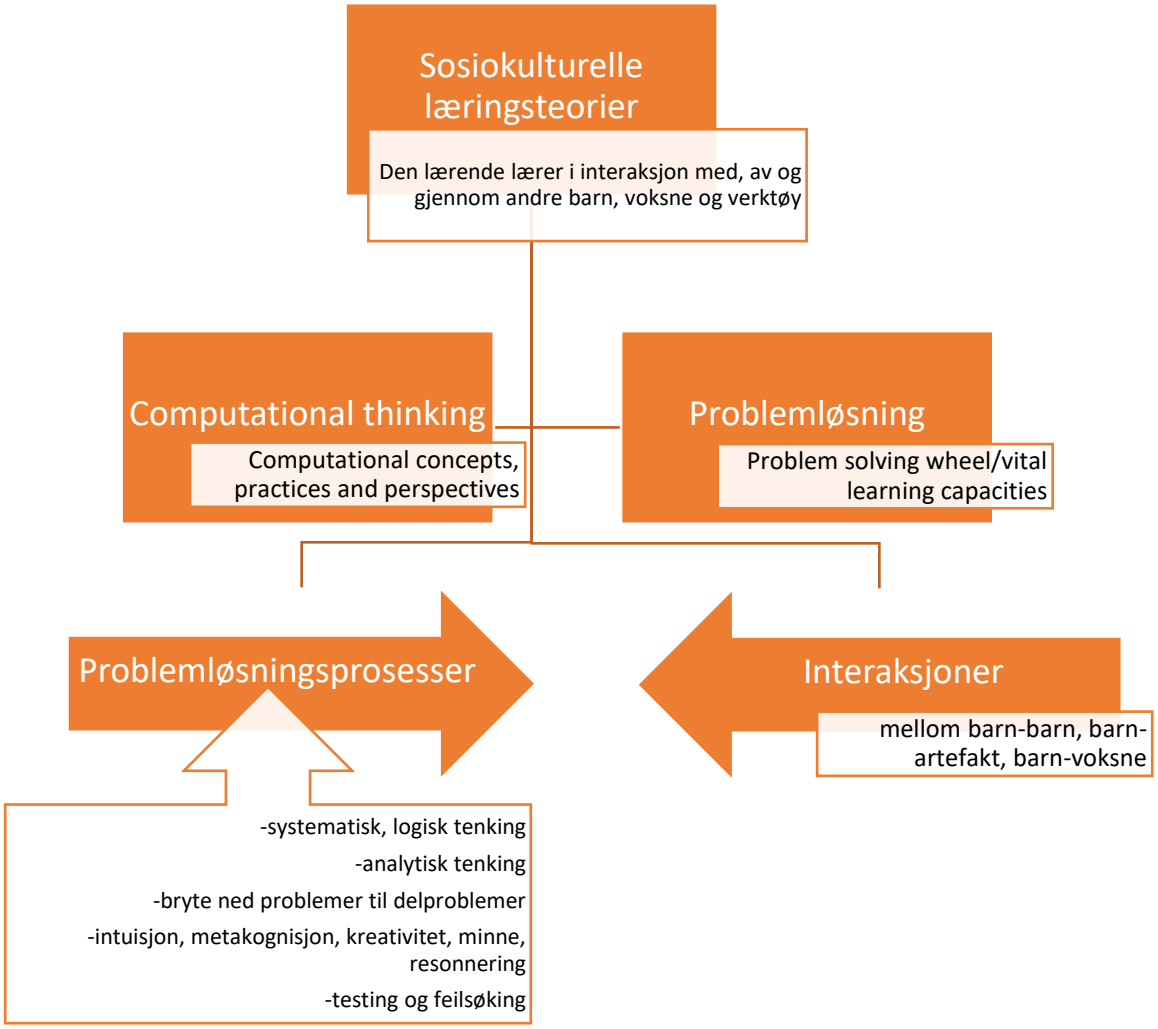
Ved å bruke modellen til Wallace, et al. (2005) får jeg delt det komplekse begrepet problemløsning til noe mer håndfast. Samtidig får jeg koplet sammen elementer fra modellen sammen med elementer i begrepet *computational thinking*. Særlig blir

computational practices (jf. Brennan & Resnick, 2012) relevant. I datainnsamlingsprosessen er det nødvendig å lete etter tegn på at elevene jobber med problemløsning. For å svare på problemstillingen er det nødvendig å se etter tegn på feilsøking og testing, men også om elevene evner å dele inn hovedproblemet inn i mindre deler. Ved å observere interaksjoner mellom barn-barn, barn-voksne og barn-artefakt, ønsker jeg å finne holdepunkter for å drøfte hvordan interaksjoner påvirker problemløsningsprosessene i programmeringsundervisningen. Modellen og læringsferdighetene til Wallace, et al. (2005) fokuserer også på selvregulering; hvilke strategier tar elevene i bruk dersom de opplever motgang i arbeidet sitt? Hvordan henter de seg inn igjen når utholdenheten og det å stå i problemet blir for vanskelig? Elevens læringsmiljø blir derfor relevant i datainnsamlingen. Barnas opplevelse av programmeringsmiljøet som trygt og pålitelig, og vilje og evne til å løfte frem sine problemer eller løsninger for andre er momenter som bør fokuseres på i datainnsamlingen. Dette er en del av både *computational perspectives* (å være i samspill med andre og teknologien underveis i programmeringen) og *computational practices* (problemløsningsprosesser).

Det jeg opplever som en noe utfordrende konsekvens av teorivalget mitt, er at jeg har valgt en sosiokulturell tilnærming til programmeringsaktiviteter, som har lange tradisjoner fra konstruksjonisme og Papert. Grunnen til at sosiokulturelle læringsteorier er et nyttig og relevant perspektiv å legge til grunn i dette prosjektet, er at man da kan lete etter synlige tegn (verbalt språk og interaksjoner) på læring. Dessuten kan man bruke det sosiokulturelle begrepet *medierende artefakt* om programmeringsverktøyene som blir brukt. Det tredje, og kanskje viktigste argumentet for å velge en sosiokulturell tilnærming er det jeg har vært inne på tidligere i teorikapittelet: at læring skjer i interaksjon mellom barn, lærere/støttende stillas og artefakter/verktøy.

Som figur 6 illustrerer, har sosiokulturelle læringsteorier lagt grunnlaget for analysene. Denne læringsteorien, som kan sees på som en av de store, *generelle* læringsteoriene, kobles deretter opp mot mer spesielle teorier om problemløsning og *computational thinking*. Derfor ligger disse to teoriene under sosiokulturelle læringsteorier. Teoriene om problemløsning og *computational thinking* blir videre konkretisert til problemløsningsprosesser, som er brutt ned til prosesser og kjennetegn som kan finnes igjen i klasserommet. Disse prosessene går gjerne på tvers av generell problemløsning og *computational thinking,* og elementene og prosessene som ligger der flyter litt over i

hverandre. Et eksempel er at jeg, i analysene mine, forventer å finne igjen læringskapasitetene til Wallace, et al. (2005) i *computational practices* (f.eks. intuisjon, kreativitet, metakognisjon) og muligens også *computational perspectives*. Når elevene arbeider med disse prosessene, får de hele tiden tilbakemeldinger, støtte eller veiledning (interaksjoner) på hvordan de utfører oppgaven. Interaksjonene skjer mellom mennesker og mellom mennesker og verktøy.



Figur 6: Gjennom interaksjoner mellom elever, voksne og verktøy får den lærende støtte, veiledning og tilbakemeldinger på hvordan hun løser et problem.

3. Forskningsdesign og metode

3.1.Forskningsdesign

Prosjektets problemstilling og teorivalg krever at forskningsdesignet er utformet slik at metodene som velges, åpner for å forstå hvordan barn arbeider med problemløsning og

programmering gjennom *interaksjoner*. Metodekapittelet innledes med en kort presentasjon av epistemologi og en gjennomgang av kvalitative og kvantitative metoder. I dette kapittelet blir det også redegjort for valg av forskningsdesign og metoder. Det innebærer en gjennomgang av benyttede metoder, intervju og observasjon, og redegjørelse for utvalg, gjennomføring av metoder og analysedelen av datainnsamlingen. Mot slutten av kapittelet drøftes forskningsprosjektets gyldighet og pålitelighet.

3.2.Mitt epistemologiske utgangspunkt

Postholm & Jacobsen (2018) forklarer *epistemologi* med hvordan man kan få vitenskapelig kunnskap om virkeligheten (Postholm & Jacobsen, 2018, s. 45). Det finnes forskjellige retninger som forklarer de ulike vitenskapsteoretiske perspektivene, og et av disse er *konstruksjonisme*. Gjennom dette perspektivet blir forståelse skapt gjennom kontinuerlig dialog mellom forsker og forskingsobjekt (Postholm & Jacobsen, 2018, s. 50). Like viktig erkjenner en konstruktivistisk forsker at man ikke kan lage faste regler eller retningslinjer om det man forsker på. Fordi kunnskap er i stadig endring, kan man bare gjengi en forståelse av det man ser, analyserer og tolker. Problemstillingen min søker etter å forstå interaksjoner mellom elever, lærere og programmeringsverktøy, og undersøke hvordan problemløsningsprosesser skjer i et programmeringsfellesskap. Konstruktivisme er det vitenskapsteoretiske perspektivet som passer oppgaven best, fordi gjennom disse «brillene» kan jeg skape forståelse gjennom interaksjon og samhandling mellom mennesker og andre artefakter. Innenfor konstruktivisme finnes det flere perspektiver, og man kan se på konstruktisme som et paraplybegrep. Sosiokulturell læringsteori, som er gjort rede for i kapittelet over, har et klart konstruksjonistisk utgangspunkt.

3.3.Kvalitativ og kvantitativ tilnærming

I et forskningsprosjekt kan man velge mellom kvalitative og kvantitative metoder for å hente inn grunnlag til å svare på problemstilling og forskningsspørsmål. Thagaard (2009) forklarer kvantitative metoder som metoder som fokuserer på utbredelse, hyppighet/frekvenser eller kvantitet (Thagaard, 2009, s. 17). Ved å velge kvantitativ tilnærming vil forskeren gjerne forhåndskategorisere begreper, og på grunn av dette kan forskeren standardisere informasjonen han henter inn ved hjelp av frekvenser, hyppighet og måling (Postholm & Jacobsen, 2018, s. 166). Dermed får man større avstand til

informantene sine ved å velge kvantitative metoder. Kvalitative metoder fokuserer derimot mer på *dybdeforståelse* og å konstruere kunnskap og forståelse i et fellesskap med forskningsdeltakere. I forhold til valgt forskningsdesign, epistemologiske utgangspunkt og teorivalg vil som sagt kvalitative metoder passe best. Det er fordi den kvalitative forskeren er opptatt av å undersøke «(…) sosiale prosesser i deres naturlige setting» (Postholm, 2010, s. 34). Som en kvalitativ forsker vil man oppsøke situerte aktiviteter, og dermed må man ut på feltet for å observere og undersøke fenomenet man er opptatt av. Problemstillingen og forskningsspørsmålene blir gjentatt for å vise hvorfor kvalitative metoder passer best:

Problemstilling: Hva kjennetegner programmeringsøkter i småskolen der barn får utvikle problemløsningsferdigheter?

F1: Hva mener lærere fremmer og hemmer elevenes muligheter for å utvikle problemløsningsferdigheter sammen med andre?

F2: Hva kjennetegner situasjoner der barn får mulighet til utvikling av problemløsningsferdigheter når de programmerer?

Disse spørsmålene retter fokus på kontekster som man kan finne igjen i småskolen. Jeg ønsker å forklare og forstå samspill og prosesser i forbindelse med læring, og derfor er en kvalitativ tilnærming mest hensiktsmessig.

3.4.Casestudiedesign

Yin (2009) definerer casestudie som en «empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident (Yin, 2009, s. 18). Han hevder at fordi det ofte ikke er mulig å skille fenomen og kontekst i «real life»-situasjoner, vil casestudiet støtte seg på flere beviskilder og triangulering. Samtidig vil forarbeidet med et teoretisk bakteppe styre hvor man skal fokusere på innsamling av data til gjenstand for analyse (Yin, 2009, s. 18). I dette forskningsprosjekt er det valgt en sosiokulturell tilnærming til forskningsfenomenet, og analysene skal derfor sentreres rundt interaksjoner mellom barn, voksne og verktøy for å forstå hvordan barna lærer. Fokuset på sosiokulturell læringsteori og *computational thinking* i *forkant* av datainnsamlingen er forenelig med Yin (2009) sine påstander om viktigheten av å ha et teoretisk bakteppe før man starter undersøkelsene. Da vet man hva man skal se etter.

Casestudie gir mulighet til å forske på hvordan den *konteksten* elevene befinner seg i, hemmer eller fremmer muligheten til å utvikle problemløsningsferdigheter gjennom programmering, ved bruk av flere kilder (triangulering). Dermed antyder jeg at utviklingen av problemløsningsferdigheter gjennom programmering helt eller delvis har sammenheng med konteksten rundt eleven i klasserommet. Konteksten er en naturlig setting fra barnas skolehverdag, der målet blir å undersøke menneskelige prosesser og interaksjoner. Dette er også i tråd med et konstruktivistisk vitenskapelig perspektiv og det Postholm (2010) mener argumenterer for en kvalitativ tilnærming. Dessuten finner vi noe av det samme hos Bent Flyvbjerg (2010). Han mener at casestudie gir kontekstavhengig kunnskap som «forskning i læring viser er nødvendig for, at mennesker kan udvikle sig fra regelbundne begyndere til helbefarne eksperter» (Flyvbjerg, 2010, s. 466). Han hevder at det nettopp er velvalgte kasus som hjelper oss å forstå og oppnå kompetanse om læring. I motsetning til f.eks. et post-positivistisk perspektiv der man mener at kunnskap og forståelsen er uavhengig av konteksten (Postholm & Jacobsen, 2018), løfter konstruktivismen frem den enkelte og unike konteksten for forskningen.

Fordi problemstillingen tar utgangspunkt i å undersøke hvordan elevene utvikler problemløsningsferdigheter gjennom programmering, passer dette prosjektet inn i en beskrivende casestudie. Jeg ønsker å finne ut hva som kjennetegner programmeringsøkter der barn får utvikle problemløsningsferdigheter. Problemstillingen er ikke formulert slik at jeg stiller spørsmål *om* elevene får utvikle problemløsning eller ikke når de programmerer, men hva som kjennetegner økter der barna får trene på problemløsning. I dette prosjektet ligger sosiokulturell læringsteori og teori om *computational thinking* som et bakteppe for å forstå hvordan barn kan utvikle problemløsningsferdigheter gjennom programmering. Tidligere forskning har dokumentert sammenheng mellom problemløsning og programmering (Fessakis, et al., 2013; Lye & Koh, 2014; Sipitakiat & Nusen, 2012), men det er behov for å forske på hvordan problemløsningsferdighetene på best mulig måte kan utvikles etter norske rammefaktorer i klasserommet (jf. delkapittelet om implikasjoner fra tidligere forskning).

F1 heter: «Hva mener lærere fremmer og hemmer elevenes muligheter for å utvikle problemløsningsferdigheter sammen med andre?» Her har jeg valgt intervju som metode for å belyse forskningsspørsmålet, og det valgt to forskjellige enheter: intervju

med klassens kontaktlærer og andrepedagog i kodetimene og skolens avdelingsleder. Skolens avdelingsleder har et spesielt lederansvar for skolens fokus på digital læring, og er derfor valgt som en egen enhet.

Forskningsspørsmål 2 handler om kjennetegn på situasjoner der barn får mulighet til å utvikle problemløsningsferdigheter. Da blir en småskoleklasse en naturlig enhet, og observasjon er valgt som passende metode for å lete etter kjennetegn. Samtidig kan også utsagn og analysene fra intervjuene bidra til svare på dette forskningsspørsmålet. Derfor kan vi kalle forskningsdesignet i dette prosjektet som et *embedded* singlecase. I følge Yin (2009) er det hensiktsmessig å velge en holistisk (og ikke embedded) studie når ingen logiske enheter, *subunits*, er naturlig eller logisk. I dette tilfelle finnes det naturlige og logiske enheter (lærer, avdelingsleder, elevgrupper) for å finne svar på problemstillingen, og da kan man ifølge Yin operere med en *embedded* case. Når jeg observerer, henter jeg informasjon om det mennesker faktisk gjør og interaksjonene som skjer, og gjennom intervjuene samler og analyserer jeg data der mennesker forteller hva de gjør (Tjora, 2010, s. 38). Dermed gir casestudie meg mulighet til å se på om det intervjuinformantene forteller (F1), faktisk skjer i praksis (F2). Kasuset er begrenset til en klasse og en skole, og derfor blir forskningsprosjektet definert som et *embedded singlecase.*

3.5.Metoder

3.5.1. Utvalg og enheter

Velvalgte casestudier og enheter hjelper forskeren med å oppnå kompetanse, og kontekstuavhengige datamateriale og regelsett vil holde forskeren på et nybegynnernivå (Flyvbjerg, 2010, s. 466). På denne måten blir fordelen med valget av casestudie som design en fordel. Her kan forskeren fordype seg i konteksten i det gitte kasuset.

Det viktigste kriteriet i utvelgelsen av informanter var en klasse (med tilhørende lærere) som arbeidet jevnlig med kodetimer eller skaperverksted i sin naturlige skolehverdag. Med det menes at klassen ikke bare utførte kodetimen som et isolert dagsprosjekt en gang i året, men at kodetimer og programmering var implementert i skolens timeplaner, gjerne også i planarbeid. Dette kriteriet var nødvendig for å sikre at elevene og lærerne var trygge på de arbeidsmåtene, aktivitetene og oppgavene/verktøyene klassen arbeidet med.

Dessuten var det nødvendig å finne en klasse som jevnlig *samarbeidet* om å programmere (jf. teoretisk bakteppe og problemstilling). For å finne svar på forskningsspørsmål 1, brukes primært enheten *lærere* (som underviser i kodetimer/skaperverksted). For å finne svar på forskningsspørsmål 2 er enheten primært observasjon av elever som deltar i en kodetime/skaperverksted. Derfor ble også lokalisering av skolen et av utvalgskriteriene.

For å finne frem til det mest passende utvalget, brukte jeg nettverket mitt og spurte kollegaer og lærervenner om de kjente til skoler som arbeidet systematisk med programmering. Det viste seg at flere skoler i mitt nærmiljø hadde koding som en del av skolehverdagen sin, men en skole skilte seg ut med at de hadde en detaljert årsplan i koding. Jeg tok kontakt med denne skolens ledelse, og fikk positive svar.

Deretter vurderte jeg hvilket alderstrinn jeg skulle velge. Mtp. at jeg skulle observere elevene, var det fornuftig å velge et småskoletrinn der jeg kunne forvente en aldersadekvat kommunikasjon mellom elevene når de samarbeidet med programmering, som var lett for en utenforstående å tolke og analysere. Jeg forventet ikke å finne et språk og programmeringssamarbeid i 1. klasse, der barna selvstendig kunne samtale om testing og feilsøking uten at voksne deltok i samtalen. Når det er sagt, mener jeg *ikke* at et slikt samarbeid ikke kan finnes i 1. klasse, men at sjansene for å finne samtaler som kunne svare på problemstillingen var høyere dersom barna hadde arbeidet med programmering over tid.

Enheten som skulle observeres ble derfor en klasse i småskolen som arbeidet ukentlig med kodetime. For å bevare klassens konfidensialitet nevner jeg ikke klassetrinn i fare for at klassen kan bli gjenkjent i lokalmiljøet. Klassen besto av 25 elever. Klassen har hatt kodetimer siden 1. klasse, og skolen følger en årsplan i koding. Klassen er godt kjent med fysiske programmeringsverktøy og alle elevene har sin egen ipad. Kodetimene er obligatoriske for alle klassene på skolen.

I utgangspunktet hadde jeg bare en intervjuavtale med klassens kontaktlærer, men kontaktlæreren ønsket at en faglærer/andrepedagog skulle være med i intervjuet, siden hun også hadde ansvar for kodetimene for klassen. Derfor ble denne læreren spurt om å delta i intervjuet noen minutter før intervjuet skulle starte. Hun fikk en mer perifer rolle i starten av intervjuet. Hun støttet gjerne opp under det kontaktlæreren sa. Andrepedagogen fortalte også om sine erfaringer, men særlig dersom de skilte seg fra

kontaktlærerens erfaringer. Dette skjedde hyppigere mot slutten av lærerintervjuet. Lærerintervjuet fikk noen av de samme fordelene som et *fokusgruppeintervju* gjerne har. Lærerne fikk samle seg rundt «spesifikke saker og konstruere kunnskap og strategier i et fellesskap for å forbedre situasjoner og forhold som de befinner seg i» (Postholm & Jacobsen, 2018).

Klassens lærere er intervjuet om planlegging, gjennomføring og evaluering av aktivitetene som foregår i kodetimene. Kontaktlærer er utdannet førskolelærer, og har ingen formell kompetanse i programmering. Hun har videreutdanning i enkelte fag, for å få undervisningskompetanse høyere i småskolen. Hun har hatt klassen siden 1. klasse. Fra nå av blir begrepet *kontaktlærer* brukt om denne læreren. Andrepedagogen (lærer) er også intervjuet med samme intervjuguide. Denne læreren har klassen i enkeltfag, og fungerer som en andrepedagog i klassen. De to lærerne bytter på å ha kodetimer med klassen. *Andrepedagog* blir fra nå av brukt når det refereres fra denne læreren, eller andrepedagogen omtales i kapitlene nedenfor.

Skolens avdelingsleder (som også underviser i kodetimer og skaperverksted) er intervjuet fordi hun har et lederansvar i forbindelse med skolens digitale satsing. Derfor blir også avdelingslederen en del av konteksten som casestudiet bygger på. Det er også relevant å nevne at denne personen også har undervisningsansvar i enkeltfag, og bruker programmering som en del av undervisningen. Avdelingsleder har allmennlærerbakgrunn, men ingen formell kompetanse i programmering. Når denne personen blir referert eller omtalt fra nå av, brukes ordet *avdelingsleder*.

I analysedelen er utvalget brutt ned og sortert til 3 enheter: intervju med avdelingsleder, intervju med kontaktlærer og andrepedagog, og elevobservasjon. Utvalget i dette prosjektet kan derfor kalles *strategisk* (Thagaard, 2009)*:* Utvalget har de egenskapene som problemstillingen ønsker å belyse, og utvalget er valgt fordi det er tilgjengelig for forskeren. Dette kan ha noe å si for generaliseringen eller overføringsverdien, og vil bli nærmere gjort rede for i kapittelet om *Gyldighet og Pålitelighet*.

3.5.2. Intervju

I tillegg til observasjon, var intervju min andre metode. I et forskningsintervju er «intensjonen å utvikle kunnskap knyttet til en bestemt tematikk, og det er vanligvis forskeren som leder an intervjuet med utgangspunkt i problemstillingen og forskningsspørsmål for sin studie» (Postholm & Jacobsen, 2018, s. 117). Jeg har valgt å

benytte semistrukturerte intervju i dette prosjektet. I et semistrukturert intervju forberedes en intervjuguide med forslag til noen spørsmål som skal være med å svare på problemstillingen (Kvale & Brinkmann, 2015). Det er også åpent for at informanten kan bringe inn tema som forskeren ikke hadde tenkt på i forberedelsene til intervjuet (Postholm & Jacobsen, 2018). Spørsmålene og temaene følger ikke nødvendigvis en bestemt struktur, men blir tatt inn i samtalene der det er naturlig. Det er også åpent for at forskeren kan ta opp spørsmål som ikke er forberedt på forhånd, og ber informanten om å utdype interessante elementer som fortelles om. Derfor foregår det en kontinuerlig analyse av informasjonen som samles inn.

Den åpenbare grunnen til at jeg valgte semistrukturerte intervju, var at intervjuenhetene hadde forskjellige roller i caset som er valgt ut. Samtidig skulle de være med på å svare på problemstillingen. Informantene kunne ha forskjellige perspektiver og meninger på elementer som er viktige for problemstillingen. Derfor er samme intervjuguiden brukt på alle informantene.

Intervjuet av avdelingslederen var den første delen av datainnsamlingsprosessen. En uke etter at intervjuet med avdelingslederen var gjennomført, fikk jeg intervjuavtale med klassens kontaktlærer og andrepedagog. Begge intervjuene ble gjennomført på skolen der informantene jobbet. Det ble tatt lydopptak av samtalene, og opptakene ble deretter transkribert av meg i vanlig tekstbehandlingsprogram.

Pilotintervju

Et pilotintervju er nyttig for å sikre at intervjuguiden fungerer slik forskeren har tenkt. Jeg ønsket å finne ut om spørsmålene jeg stilte var mulige å svare på for en barneskolelærer, og om programmeringsbegrepene som ble brukt der var gjenkjennelige for småskolelærere. Dessuten ønsket jeg å se om svarene jeg fikk fra spørsmålene var mulige å tolke og analysere, og om de så ut til å kunne være med på å svare på problemstillingen. Derfor gjennomførte jeg et pilotintervju i forkant av datainnsamlingen.

Her var informanten en av mine kollegaer som kan en del om programmering i skolen. Jeg gjorde ingen justeringer i intervjuguiden etter dette pilotintervjuet, for jeg opplevde å få presise, grundige og informative svar på spørsmålene i den semistrukturerte spørsmålsguiden. Jeg var klar over at selv om intervjuguiden fungerte godt på min

testinformant, kunne jeg ikke ta for gitt at den også ville fungere like bra for mine informanter som skulle bidra til datainnsamling i denne studien.

3.5.3. Observasjon

Observasjon ble valgt som metode for å se etter kjennetegn på situasjoner der barn utvikler problemløsningsferdigheter i klasserommet. Observasjon som forskningsmetode blir ofte beskrevet som en naturalistisk metode (Angrosino & Mays de Perez, 2000). Det er fordi fenomenet som forskes på blir observert i sin naturlige setting, der man får tilgang til den menneskelige aktiviteten eller settingen rundt fenomenet som forskes på (Postholm & Jacobsen, 2018, s. 114). Som observatør kan man innta forskjellige roller i observasjonen. Postholm & Jacobsen (2018) viser til Gold (1958) når de presenterer de ulike observatørrollene i en tabell. De skiller mellom:

- Deltaker-som-observatør
- Fullstendig observatør
- Fullstendig deltaker
- Observatør-som-deltaker

  (Gold, 1958 i Postholm & Jacobsen, 2018, s. 115).

Dersom man observerer som «*deltaker-som-observatør*», blir observatørrollen tydeligere definert enn når forskeren inntar en fullstendig deltaker. «I rollen som fullstendig observatør har forskeren ingen tilknytning til situasjonene som blir observert, og han eller hun vil på ingen måte samhandle med de som observeres» (Postholm & Jacobsen, 2018, s. 115). En fullstendig deltaker vil være en del av det som observeres. En observatør-som-deltaker deltar ikke i aktiviteten som observeres, og forskeren er mest observatør.

Erfaring fra småskolen har lært meg at barn er nysgjerrige på nye voksne som kommer inn i ei elevgruppe. Derfor planla jeg på forhånd at jeg skulle innta en *deltaker som observatør*-posisjon i observasjonsøktene. På den måten blandet jeg meg ikke inn i selve undervisningsopplegget som pågikk, men kunne kommunisere med barna og svare vennlig på spørsmål om hvem jeg var og hva jeg gjorde der.

Klassen ble presentert for meg første gang i forbindelse med den første observasjonsøkten. Da var allerede avdelingsleder og lærerne intervjuet. Jeg valgte å forklare at jeg skulle være med i klassens kodetime som *deltaker som observatør,* og forklarte hva det betydde for elevene. Denne presentasjonen gjorde at alle deltakerne i

kodetimen visste hva jeg gjorde der, og hvordan de kunne forholde seg til meg. Intensjonen min var at forklaringen på observatørrollen og presentasjonen kunne skape forutsigbarhet og trygghet for elevene, og sjansen ble derfor mindre for at elevene blir distrahert når de blir observert.

Denne klassen er observert i to undervisningsøkter. Av 25 samtykkeskjema som ble sendt ut ca. ei uke før observasjonene skulle starte, ble 15 sendt i retur med godkjenning fra foresatte om at barnet deres kunne observeres. Klassens kontaktlærer tok utgangspunkt i disse 15 elevene når hun planla gruppesammensetning i de øktene klassen skulle observeres. Jeg har ikke observert elevene som ikke har levert samtykke. Totalt fikk jeg observert 6 elevgrupper. To grupper jobbet med LEGO WeDO, to arbeidet med Blue-bot og to grupper arbeidet med Bee-bot. Alle disse gruppene befant seg på hver sine grupperom. Skolens skaperverksted ble ikke brukt i observasjonene. Men for konteksten og kasuset sin del er det fornuftig å nevne at skolen har et skaperverktsted, hvor de oppbevarer ulike programmeringsverktøy og kan bruke rommet som et verksted.

Observasjonsnotater (vedlegg 6) ble skrevet ned i en loggbok samtidig som observasjonene foregikk. Her var det allerede stikkord som skulle hjelpe meg å finne observasjoner som var relevante i forhold til problemstillingen, men jeg var selvsagt åpen for andre tema enn det som var foreslått på forhånd. Postholm (2010) og Postholm & Jacobsen (2018) beskriver hvordan feltnotater kan deles i to. På den ene siden av arket noteres det som skjer under observasjonene. Informasjon om kontekst og setting vil også stå her. På høyre side har forskeren mulighet til å notere analyser og tolkinger som umiddelbart faller inn. Dette ble gjort for å sikre at ikke glemte observasjoner som ville ha betydning for videre observasjonsøkter, oppfølgingsspørsmål til lærerne eller dataanalysen skulle forsvinne. Kort tid etter at observasjonene var gjennomført, ble det skrevet utfyllende beskrivelser fra det som var observert. Videoopptak og lydopptak ble ikke brukt i observasjonsøktene, og begrunnes med ønsket om å beholde elevenes konsentrasjon på programmeringen og samarbeidet.

For å ivareta den teoretiske sensitiviteten min i observasjonsarbeidet, valgte jeg å skrive ned noen momenter som kunne belyse problemstillingen med sosiokulturelle briller. Derfor er feltnotatene (vedlegg 6) skissert med forslag til elementer som er relevante for å forstå problemløsning og programmering ut fra et sosiokulturelt perspektiv. Samtidig

er det viktig å påpeke at selv om jeg på forhånd hadde skrevet ned hva jeg kunne se etter i observasjonene, var jeg åpen for nye faktorer og observasjoner som kunne belyse problemstillingen.

3.5.4 Sammenheng mellom metoder i datainnsamlingsperioden

I god tid før det ble skrevet ut observasjonsnotat og intervjuguide, satte jeg meg inn i teori om sosiokulturell læringsteori, problemløsning og CT. På bakgrunn av det utformet jeg en modell for å konkretisere kjennetegn på at elever arbeider med problemløsningsprosesser når de programmerer (se Figur 6). Disse ble tatt utgangspunkt i da jeg skrev observasjonsnotatet (vedlegg 6) og intervjuguiden (vedlegg 1), men som jeg kommer inn på i delkapittelet om analyse, var ikke disse kjennetegnene ferdig definert som koder før jeg startet datainnsamlingen. Som kvalitativ forsker må man likevel erkjenne at man tar med seg sin forforståelse inn i datainnsamlingen og analysen (Postholm, 2010). Min forforståelse har jeg hovedsakelig tilegnet meg gjennom lærererfaring (se kapittel 1), og etter timer med teori- og litteraturgjennomgang om fenomenet jeg skriver om.

De første forskningsdataene ble hentet inn i et intervju med skolens avdelingsleder. Der fikk jeg informasjon om konteksten og rammefaktorer (som hemmer og fremmer elevenes muligheter for å utvikle problemløsningsferdigheter) som var relevante for forskningsfenomenet mitt. I tillegg fortalte avdelingslederen om sin generelle forståelse av programmeringen sin rolle i skolesammenheng. Jeg fikk også innblikk i skolens årsplan for programmering, og innsyn i hvilke verktøy og arbeidsmåter skolen benyttet seg av. Dette intervjuet har nok hatt betydning for min førforståelse før observasjonene og lærerintervjuet ble gjennomført. Jeg hadde avdelingslederens oppfatninger og vurderinger av alt fra programmeringsverktøy til elevgrupper med meg før jeg kom inn i klasserommet og startet observasjonene. Dataene fra intervjuene og observasjonene skulle analyseres og sammenlignes i etterkant av datainnsamlingsprosessen, men på denne måten startet også analysedelen når jeg var ute i feltet. Pga. masteroppgavens begrensning i tid og intervjuinformantenes ledige tider for intervju, måtte jeg gjennomføre lærerintervjuet før elevobservasjonene startet.

I observasjonsøktene var det fornuftig å bli værende hos en elevgruppe over lang tid for å forstå hva som skjedde og hvorfor det skjedde. Et eksempel på dette var at avdelingsleder, andrepedagogen og kontaktlærer trakk frem elevenes evne til å «stå i et

problem» som en faktor for utvikling av problemløsningsferdigheter. Derfor ble også kodene lagt til i kodeboken før observasjonene startet. Da var det nødvendig å sitte lenge nok med elevene til å observere evnen til utholdenhet og/eller selvregulering. Konsekvensen av det ble begrenset tid til å observere andre elevgrupper.

Det ideelle hadde vært å gjennomføre et lærerintervju i forkant av observasjonene, og et i etterkant. Da kunne gjerne andre typer spørsmål eller tema vært fokus i lærerintervjuene, og dermed hadde jeg utnyttet metodetrianguleringen på en bedre måte. Alle intervjuene har vært gjennomført i forkant av observasjonene, og det har nok også vært med på å prege observasjonsarbeidet. Disse refleksjonene til tross, som Wolcott (2008) referert i Postholm & Jacobsen (2018) konstaterer, er subjektivitet alltid et element av forskningsprosessen. Wolcott (2008) poengterer at i observasjonen inngår også forskerens personlige erfaring, og at strenge objektivitetsprinsipper ikke er noe som bør etterstrebes i denne type forskningsprosjekteter (Postholm & Jacobsen, 2018, s. 130). Dette henger sammen med en konstruktivistisk epistemologi, der forståelse og kunnskap blir skapt av de som deltar i observasjonene i kodetimen.

3.6. Analyse

Hensikten med analysemetoder er å sortere datamaterialet som er samlet inn, for å gjøre materialet forståelig (Posholm & Jacobsen, 2018, s. 139). Når kvalitative metoder brukes, er analyseelementet tilstede i hele forskningsprosessen, også før man setter seg ned med materialet i etterkant av datainnsamlingen. Det var også tilfellet i denne masteroppgaven. Under semistrukturerte intervju ble oppfølgingsspørsmål stilt, og disse intervjuene hadde betydning for elevobservasjonene og hva jeg kunne forvente å se når jeg senere kom til klassen for å observere.

Den mer formelle delen av analyseprosessen begynner gjerne med deskriptiv analyse. Forskeren ønsker å kartlegge og definere mønstre som kan samles i koder, kategorier eller forskjellige tema. Hovedmålet med et kvalitativt forskningsprosjekt er imidlertid ikke bare å beskrive, men som Stake (1995) argumenterer for: finne mening i datamaterialet som er hentet inn og å utvikle en forståelse av kasuset som er studert. Forskeren streber etter å forstå individuelle situasjoner på den ene siden. På den andre siden vil forskeren forstå kasuset, slik at forskningen på gjeldende case får betydning for andre lignende kasus (Postholm & Jacobsen, 2018, s. 157).

Begrepet *teoretisk sensitivitet* viser til hvordan forskeren kan analysere, forstå og gi mening til datamaterialet som kommer inn (Postholm & Jacobsen, 2018, s. 142). Ved hjelp av begrep, prinsipper og innhold i valgt teori og tidligere forskning, blir forskeren sensitiv for hva elementene i datagrunnlaget betyr. «Både direkte analyse og kategorisk opphopning avhenger av letingen etter mønster. Noen ganger er disse kjent på forhånd, mens andre ganger vokser de frem i analysen» (Stake, 1995 i Postholm & Jacobsen, 2018, s. 158). Forskeren vil, ved å sette seg inn i teorien, gjøre seg noen antakelser som enten vil bli bekreftet eller avkreftet etter hvert som datamaterialet blir behandlet.

Analyseprosessen

Teoriene som er valgt, og de tidligere erfaringene som jeg har tilegnet meg gjennom observasjoner av elever som gjennomfører programmeringsaktiviteter, har gjort at det forelå noen antakelser om hvordan elevene trente på problemløsning gjennom programmering. Selv om disse antakelsene lå der før datainnsamlingen startet, var det viktig med et åpent og nyansert tolkningsperspektiv på datamaterialet. På den måten kan koder utvikles fra materialet som kommer inn.

Programmet HyperRESEARCH ble brukt for å definere og sortere datamaterialet. Etter at alle intervjuene og ei observasjonsøkt var gjennomført og transkribert, ble avdelingslederintervjuet analysert i programmet. Her ble en *kodebok (*vedlegg 2), som ble felles for alle analyseenhetene, opprettet. *Teoretisk sensitivitet* ble viktig for at kodene ble utviklet med prinsippene om sosiokulturell læringsteori, teori om CT og problemløsning i bakhodet etter hvert som det første intervjuet ble analysert. Samtidig hadde jeg forskningsspørsmålene koblet opp mot datagrunnlaget når jeg definerte kodene:

***F1**: Hva mener lærere fremmer og hemmer småskoleelevers muligheter for å utvikle problemløsningsferdigheter sammen med andre når de jobber med programmering?*

***F2**: Hva kjennetegner situasjoner der barn får mulighet til utvikling av problemløsning når de jobber med testing og feilsøking?*

Etter at avdelingslederintervjuet var analysert, bestod kodeboken av 18 koder. Det andre intervjuet bestod av data som kunne plasseres under de fleste kodene fra avdelingslederintervjuet, men noen flere koder ble også lagt til underveis i analysearbeidet fra lærerintervjuet. Da hadde jeg 25 koder klare (se vedlegg 2) på forhånd før analysene fra observasjonene begynte. Dette opplevdes utfordrende. Jeg

ønsket å være så åpen, nyansert og teoretisk sensitiv som mulig gjennom hele analysearbeidet. Samtidig var enhetene en del av samme kasuset. Derfor ble kodene som var opprettet fra lærerintervjuene brukt som utgangspunkt for analysene av observasjonene. Disse kodene stammer fra samme kasuset, som også gjør det naturlig å bruke de samme kodene i alle analyseenhetene:

> Forståelsen utviklet på bakgrunn av de enkelte delene (observasjon og intervju) vil kunne bidra til at forskeren får utviklet helhetsforståelsen for fokusområdet som studeres. Forskeren befinner seg her i den hermeneutiske spiralen, en prosess som bidrar til å skape forståelse og mening. (Postholm & Jacobsen, 2018, s. 130)

3.7.Gyldighet og pålitelighet

Begrensninger som er knyttet til egen forskning og hvordan forskeren kan ha påvirket de endelige resultatene gjennom sin egen forskning er viktig å reflektere over. Studiens *gyldighet* kan deles inn i to forskjellige typer: indre og ytre gyldighet. «Indre gyldighet går på om det vi har kommet frem til- de konklusjonene vi trekker- er gyldige for de eller det vi har studert (…) Ytre gyldighet eller overførbarhet relaterer seg til i hvor stor grad kan resultatene overføres til andre skoler?» (Postholm & Jacobsen, 2018, s. 223). Studiens pålitelighet knytter Postholm & Jacobsen (2018) til refleksjon over hvordan selve undersøkelsen og forskeren kan påvirke resultatet. Gjennom å reflektere over sin påvirkning og å gjøre forskningsprosessen synlig slik at andre også kan reflektere over den, blir forskningens pålitelighet fremmet (Postholm & Jacobsen, 2018, s. 224).

3.7.1. Indre gyldighet

Det er viktig å reflektere over hvor gyldige begrepene som er brukt i denne forskningen er. Når analysene blir presentert nedenfor, er overskriftene og de tykke beskrivelsene som blir presentert, valgt ut fordi de er hentet fra datamaterialet og kan være med på å svare på problemstillingen. Det er datamaterialet som er hentet fra intervjuer og observasjoner som presenteres i analysekapittelet, og koder og kategorier er utviklet fra dette materialet (vedlegg 2).

Når det er sagt, er forskerens forforståelse og teoretisk sensitivitet ikke uten betydning. Likevel har jeg analysert og observert så åpent og nyansert som mulig. På den måten har nye tanker, ideer og funn som er med på å svare på problemstillingen, men som jeg ikke har tenkt over, fått sin plass i analysekapittelet. Slik har jeg pendlet mellom induktiv og deduktiv tilnærming.

Da jeg utførte intervjuene, prøvde jeg etter beste evne å bruke begreper som jeg forventet at informantene kjente til, og å forklare begreper som jeg så at informantene ikke skjønte eller opplevde behov for å forklare for dem. Postholm & Jacobsen (2018) definerer dette som deltaker-validering. Deltakerne i intervjuene og observasjonene har fått tilbud om å lese gjennom transkriberte intervju og beskrivelsene fra observasjonene for å bekrefte eller avkrefte tolkningene mine, men jeg har ikke fått tilbakemeldinger fra dem. Jeg har etter beste evne forklart begreper som er sentrale for å forstå problemstillingen og forskningsspørsmålene. Et vesentlig poeng som må nevnes her, er at det var en av forskningsdeltakerne som først tok opp begrepet *feilsøking* i forbindelse med problemløsning da vi snakket sammen første gang. Det gav meg et trygghet om at deltakerne hadde en forståelse av sammenhengen mellom feilsøking og problemløsning.

I tillegg til å beskrive og forklare sammenhenger mellom problemløsning og programmeringsøkter i småskolen, har jeg hatt som mål å forklare hvorfor (eller hvorfor ikke) elevene utvikler problemløsningsferdigheter gjennom programmering. «Vi må være åpne for at det kan være tiltak som virker på tvers av ulike kontekster, selv om det ikke betyr at effektene vil være identiske i alle kontekster» (Postholm & Jacobsen, 2018, s. 233). Da er det viktig å påpeke at fordi en faktor virker fremmende i mitt kasus, er det ikke sikkert at denne faktoren vil virke fremmende i en annen.

3.7.2. Triangulering

Alle forskningsdesign og metoder har sine styrker og svakheter. «Kombinasjon av flere ulike forskere, forskningsdesign, datainnsamlingsmetoder og datakilder er en måte å styrke både pålitelighet og gyldighet på» (Postholm & Jacobsen, 2018, s. 236). Dette er et prinsipp jeg har benyttet når jeg valgte casestudiet som forskningsdesign, og jeg har inkludert flere informanter (voksne og barn) og metoder (intervju og observasjon) i datainnsamlingsgrunnlaget. Samtidig har jeg under litteratursøket vært åpen for både kvalitative og kvantitative kilder, flere nasjonaliteter, osv. Når det er sagt, påpeker Postholm & Jacobsen (2018) at masterstudenter skal passe seg for å gape for stort over feltet. Samtidig opplever jeg at intervju og observasjon av deltakere i det samme kasuset, ikke har vært for stort å gape over. Tvert imot, triangulering i casestudie har gitt meg dypere forståelse av fenomenet i den konteksten jeg har forsket på.

3.7.3. Ytre gyldighet: overførbarhet

I et kvalitativt forskningsprosjekt vil ytre gyldighet knyttes til prosjektets *overførbarhet.* «I et kvalitativt perspektiv vil overføring være knyttet til hvorvidt en beskrivelse er gjenkjennbar, altså om den som leser forskningen kan si: «Dette ligner mye på min situasjon!»» (Postholm & Jacobsen, 2018, s. 238). Ved å skrive slik at leseren opplever å bli invitert inn i forskningsprosessen rundt det aktuelle kasuset, styrkes overførbarheten (Postholm & Jacobsen, 2018, s. 238). Derfor er analyse- og metodekapittelet beskrevet med tykke beskrivelser, der konteksten rundt utvalget er beskrevet nokså detaljert, handlinger knyttet til observasjoner blir fyldig presentert og intervjuinformantenes meninger presentert og tolket gjennom kategorier og koder (vedlegg 2). Derfor har jeg gjengitt og beskrevet forskningsprosessen så detaljert jeg kan, slik at leseren selv kan vurdere hva som kan overføres til egen situasjon. Denne formen for overførbarhet kaller Postholm & Jacobsen (2018) for *naturalistisk generalisering*.

3.7.4. Pålitelighet

Et kvalitativt forskningsprosjekt er vanskelig å replikere fullt ut, fordi forskeren tar med seg sin subjektive, individuelle teori i forskningsprosessen (Posthom, 2010; Postholm & Jacobsen, 2018, s. 224). Dessuten er mennesker i utvikling. Når det er sagt, har jeg etter beste evne beskrevet utvalget og konteksten informantene befinner seg i, for å invitere leseren med i forskningsprosessen. Under prosessen med å skrive selve oppgaven har jeg også hatt som mål å skrive så transparent som mulig. Deltaker-valideringen har også bidratt til å styrke prosjektets pålitelighet, f.eks. mtp. begreper og ord som brukes i intervjuene. Skolen og informantene var strategisk valgt ut fordi jeg mente at dette kasuset kunne bidra til å svare på problemstillingen min. Samtidig er jeg klar over at dette er en skole som er kommet lenger enn mange andre skoler når det kommer til systematisk programmeringsopplæring.

3.8. Etikk

Gjennom hele forskningsprosjektet har jeg prøvd så godt jeg kan å holde meg innenfor NESH sine retningslinjer for forskningsetikk (NESH, 2006). Alle informantene i prosjektet har skrevet under på et samtykkeskjema, der de har fått grundig informasjon om studiens formål og innhold. De har fått tydelige instrukser for hva de skal gjøre dersom de ønsker å trekke seg, be om innsyn eller har spørsmål til studien. Det er barnas foresatte som har gitt samtykke for småskolebarna, men kontaktlærer har i tillegg spurt hvert enkelt barn om de selv ønsker å delta. Når samtykke forelå, passet

kontaktlæreren på at det bare var elever med samtykke som ble observert. NSD har gjennomgått mitt utkast til samtykkeskjema for lærere og foresatte (vedlegg 4 og 5) og intervjuguide (vedlegg 1), og har vurdert prosjektet som godkjent ut fra gjeldende lovverk om personopplysninger (vedlegg 3). Postholm & Jacobsen (2018) tydeliggjør nødvendigheten av at informantene får grundig kjennskap til prosjektets formål, innhold, datainnsamlingsstrategier og hvordan data skal publiseres. Dette redegjøres for i samtykkeskjemaet som barnas foresatte og de voksne intervjuinformantene har signert på. På samtykkeskjemaet for pedagogene er det også påpekt at lydopptak blir gjort, og hvordan lydopptaket blir behandlet i etterkant av intervjuet. Referanser er redegjort for, og jeg har gjort mitt beste for å opptre så redelig som jeg kan gjennom hele forskningsprosjektet.

4. Analyser av datamaterialet

Analysekapittelet er delt opp i to delkapittel. Det første delkapittelet handler om analyser fra intervjuene som var gjennomført i denne undersøkelsen, og primært skal denne metoden svare på F1. Det andre delkapittelet presenterer analyser og fortellinger fra observasjonene, og primært skal observasjonene være med på svare på F2.

4.1. Analyse av intervju

Her blir analysene presentert under utvalgte tema. Disse temaene er hentet ut fra datamaterialet. Totalt 3 kodegrupper ble opprettet etter datainnsamlingsprosessen, og 25 koder er fordelt innenfor disse kodegruppene (se vedlegg 2). Kodegruppene er

1) Planlegging og gjennomføring av programmeringsøkter

3) Problemløsning

2) Samarbeid, veiledning og støtte

Planlegging og gjennomføring av programmeringsøkter er valgt som kategori fordi avdelingsleder, kontaktlærer og andrepedagog trekker frem ulike didaktiske vurderinger de gjorde i *forkant* av kodetimene. Disse vurderingene og valgene er faktorer som kan fremme eller hemme problemløsningsferdigheter, og da er det fornuftig å sortere ut sitater og analyser fra datamaterialet som kan passe under denne kategorien. Det er presentert tidligere forskning i kap. 1.4 som belyser hvordan programmeringsverktøy og lærerens funksjon og rolle påvirker barns problemløsningsferdigheter i

programmeringsaktiviteter. Denne forskningen kan knyttes til didaktiske valg og begrunnelser som lærere gjør i forkant av programmeringsøkter.

Problemløsning er konstruert som kategori fordi det finnes en del utsagn som forteller noe om kjennetegn på problemløsning og hvordan barn øver på problemløsning gjennom programmering. Her kan *computational practices* settes i sammenheng med kjennetegn på problemløsning gjennom problemløsning (Brennan & Resnick, 2012). Funn og analyser som kan knyttes til en fiklende tilnærming til programmering (Resnick & Rosenbaum, 2013) er også kjennetegn på problemløsningsaktivitet.

Den siste kategorien, *Samarbeid, veiledning og støtte*, er valgt fordi lærerne trekker frem flere forhold rundt samspill mellom lærer-elev, elev-elev og lærer-elevgruppe som har betydning for hvordan elevene utvikler problemløsningsferdigheter gjennom programmering. Disse analysene blir forstått gjennom sosiokulturelle læringsteorier, som et bakteppe for å forstå læring gjennom interaksjoner mellom barn, voksne og verktøy. I alle tre kategoriene som er nevnt ligger også utsagn og analyser som viser hva som kan hemme utviklingen av problemløsningsferdigheter.

Planlegging av programmeringsøkter	Problemløsning	Samarbeid, veiledning og støtte
• Fysiske programmeringsverktøy • Gruppesammensetninger • Instruksjon til timen • Klassemiljø • Kontinuitet • Lek	• Dele i delproblemer • Fikling • Kreativitet • Logikk og systematikk • Mengdetrening • Metakognisjon • Selvregulering • Språk • Testing og feilsøking • Utholdenhet • Oppskrifter	• Avgrensning av oppgaven • Elevsamarbeid • Emosjonell støtte • Ledende spørsmål • Modellering • Proksimale utv.sone • Selvregulering • Språk

Tabell 2: Oversikt over kategorier og tema. Kodene er plassert under den kategorien hvor de hører til. Språk er plassert under problemløsning og samarbeid, veiledning og støtte (se eksempelet under.

Enkelte sitater og analyser passer inn i flere kategorier. F.eks. viser et sitat fra kontaktlærer hvordan elevene kommuniserte med lærer når de hadde lokalisert en feil,

og hva de elevene gjorde for å prøve å fikse feilen sin. Dette handler om kommunikasjon mellom lærer-elev, og kan dermed plasseres under 4.1.2. *Samarbeid, veiledning og støtte*. Jeg har valgt å plassere sitatet under dette temaet fordi sitatet gir et eksempel på hvor vanskelig der kan være for et barn å sette ord på problemet sitt. Det er likevel viktig å huske på at det er en strategi barnet bruker for å lokalisere og rette opp i feil og problemer i programmeringen.

4.1.1. Planlegging og gjennomføring av programmeringsøkter

Gruppesammensetninger

Planleggingsarbeidet i forkant av kodetimer og skaperverksted ble sett på som en viktig faktor for at barna skal utvikle problemløsningsferdigheter gjennom programmering. Avdelingsleder, kontaktlærer og andrepedagog oppga forskjellige didaktiske vurderinger som de gjorde i forkant av øktene. De mente disse hadde betydning for om og hvordan barna utvikler problemløsningsferdigheter gjennom programmering sammen med andre. Det var enkelte forskjeller i meningene deres, blant annet i synet på gruppesammensetninger:

Begge lærerne og avdelingslederen som er intervjuet trakk frem *gruppearbeid* og *gruppesammensetninger* som en faktor som kan fremme, men også hemme elevenes muligheter for å utvikle problemløsningsferdigheter når de programmerer. Derfor har gruppearbeid og gruppesammensetninger blitt utviklet til en kode i analysearbeidet. Avdelingsleder sa:

> det er noe med å sette sammen gode grupper. Jeg tenker litt på den inndelingen. Før satte vi sammen (…) hele spekteret. Også var det en som styrte, og en som bare var med, men nå setter vi sammen litt mer homogene grupper, sånn at de kan støtte hverandre. I stedet for at det er en som alltid kan svaret, og en som tenker at «jeg gidder ikke være med». Da hjelper de hverandre med å utvikle seg, i stedet for at noen gir opp, fordi at de andre kan det fra før av. (Avdelingsleder)

Med *før* mente hun da skolen prøvde ut verktøy, der de gjorde seg *erfaringer* med didaktiske valg og verktøy. Dette var da de startet med digitaliseringen av enkelte klassetrinn. I tillegg hentet de etter hvert inn kompetanse om programmering i skolen. Kompetansen ble hentet inn fra ulike kanaler, blant annet fra nasjonale arrangement om læring på digitale flater.

Avdelingslederen trakk også frem at læreren må kjenne elevene for å sette sammen gode grupper, vite hvilke styrker de har. Noe av det samme finnes også hos lærerne i

observasjonsklassen. Her fortalte kontaktlærer om en klasse som sjelden sitter en og en, og at elevene er fortrolige med å samarbeide med hverandre. Samtidig er kontaktlærer tydelig på at gruppene kan være fleksible (i motsetning til avdelingsleder, som foretrekker homogene grupper), at hun ikke deler inn etter ferdigheter, kjønn eller andre variabler. Hun er også klar på at dette ikke er spesielt for kodetimene, men at det gjelder de fleste timer hvor gruppearbeid foregår.

> Jeg har pleid å dele de opp i grupper, slik at de er nødt til å være aktive alle sammen. (…) Koding er kanskje det faget der det skille kanskje minst, sant. Fordi at der… der er det andre ferdigheter som kreves. Men det er så… det spiller ingen rolle om de jobber med jenter eller gutter der inne. Det er ikke noe… jenter-jenter, gutter-jenter, det spiller ikke noen rolle. Spiller ingen rolle om de er sterk faglig. De er ganske jevne i koding. (Kontaktlærer)

Dette betyr at kontaktlærer mener det er viktig at alle elevene er aktive, at ingen sitter passive i aktivitetene. I utdraget fra det transkriberte intervjuet med kontaktlærer og andrepedagog, ser vi også at kontaktlærer mente at skillet mellom faglig sterke og svake elever ikke betyr så mye i kodetimene. Noe av det samme fortalte også avdelingsleder om:

> Det er kanskje ikke alltid de som er gode på programmering som ikke nødvendigvis vi tenke på som gode til å lese eller skriveoppgaver. (Avdelingsleder)

Avdelingsleder skildret også hvordan elevene ubevisst delte inn ulike funksjoner i gruppa si. En satt gjerne med ipaden og la inn kodene og var teknisk anlagt, en annen passet på det fysiske verktøyet og var gjerne mer kreativt anlagt. Kontaktlærer forklarte dette:

> At den ene styrer den ene veien, og den andre styrer den andre veien. At en følger med på (den fysiske) ballen, og en styrer appen. I LEGO har gjerne èn ansvar for å følge instruksjon og en bygger, fordeler litt. (Kontaktlærer)

Alle intervjuinformantene gjentok flere ganger at man måtte kjenne elevene sine som individer, men også som elevgruppe, for å vurdere hvordan gruppesammensetningen skulle være.

Fysiske programmeringsverktøy

Avdelingslederen trakk frem fysisk programmering og fysiske programmeringsverktøy i begynneropplæringen, som en introduksjon til *computational thinking* og

programmering. Temaet er knyttet til didaktiske valg og planlegging av undervisning i forkant av undervisningsøktene. Avdelingsleder forklarte:

> Jeg har også troa på å starte den problemløsingsbiten uten maskin. Det er litt mer på første trinn. Da vi begynte, var det jo litt mer at vi bare hoppet uti det. (Avdelingsleder)

Andrepedagogen mener også frem noe av det samme. Hun fortalte om programmeringsaktiviteter som kan minne om *språkleker* for begynneropplæringen:

> Vi har hatt noen kodetimer der de har programmert hverandre. Der de har tegnet opp firkanter med hva de skal gjøre, sant, og hvor mange ganger de skal gjøre det. Filmet hverandre. Det klarer de fint. (Andrepedagog)

Kontaktlærer og andrepedagog hadde erfaringer og snakket om Bee-bot som et nyttig fysisk programmeringsverktøy, og at de stort sett brukte fysiske programmeringsverktøy. Her argumenterte andrepedagog med at Bee-boten:

> «går uansett på en måte. Så da er det jo bare å prøve igjen. Det er alltid koordinater.» (Andrepedagog)

Det betyr at fysiske programmeringsverktøy er mer konkret, visuelt og håndgripelig enn programmeringsaktiviteter som kun foregår på en digital flate. Samtidig fortalte hun at elevene av og til strevde med å vite hva de har trykket på. Bee-boten har bare knapper på ryggen, ingen visuell støtte som tydeliggjør og konkretiserer sekvensen barna har produsert. Hun har testet ut Blue-boten, en videreutvikling av Bee-boten, og forteller om fordeler med den:

> Blue-boten kan du også koble til ipad, da ser du lettere hva du har trykket. Hva gikk feil nå? (Andrepedagog)

Klassemiljø

Kontaktlærer trekker frem elevenes trygghet som en faktor for hvordan de arbeider med problemløsning. Når kontaktlærer og andrepedagog kjente til klassemiljøet i klassen, kunne de lettere planlegge *hvordan* elevene skulle arbeide med verktøyene. Avdelingsleder påpekte også at man måtte kjenne elevene sine, for å planlegge gruppearbeidet på best mulig måte.

Kontaktlærer til småskoleklassen fortalte om en klasse der elevene sjelden eller aldri sitter alene og som ofte arbeider i grupper og samarbeider, en klasse der elevene er trent på å hjelpe hverandre og der elevene leker fint med hverandre i friminuttene. Dette er tegn på et godt klassemiljø, og når andrepedagog og kontaktlærer fikk spørsmål om

hvordan klassemiljøet er, svarte de at det var bra. Kontaktlæreren trakk også frem at elevene må føle trygghet i klassen for å utvikle seg mot å bli kreative problemløsere:

> Så tror jeg tryggheten har veldig mye å si. Hvis de ikke de er trygge på hverandre inni klasserommet, og hvis ikke du som den voksne lar de få lov til å være trygge og får lov til at vi er forskjellige, så tror ikke jeg at det er helt hadde… Akseptere at vi er ulike, tenke på forskjellige måter. Løse problemene på forskjellige måter. (Kontaktlærer)

Kontaktlærer mente altså at dersom ikke barna opplever trygghet i problemløsningen, blir det vanskelig å være kreative og nysgjerrige problemløsere. For å oppnå dette var lærerne tett på elevene i oppgaveløsningen, og grep inn dersom de observerte problemer som eskalerte, eller stagnert problemløsning som var på vei mot en elevkonflikt. Kontaktlæreren i klassen forteller også dette om klassemiljøet:

> De er veldig ivrig på å vise hva de får til. Veldig ivrig. Og når de får noen ting til, så er de… dette… da har de lyst til å lære det til noen andre med en gang. Det er vel egentlig på alt som de gjør… De klarer ikke å holde det inni seg selv, de må ut å fortelle. Og det gjelder jo egentlig i alle fag generelt. Hvis de har funnet ut av noen ting, så må de fortelle. (Kontaktlærer)

Andrepedagog supplerte:

> Da subber de bort, og noen ganger tar de den der «nei, men han har hjulpet henne, så du kan gå til han, for han har allerede hjulpet henne, så gå bort dit. Der er det allerede løst. Da er det bare å subbe bort dit og få seg hjelp. (Andrepedagog)

Avdelingsleder fortalte om sammenheng mellom klassemiljø og kodetimer, men hun vinklet det slik at programmeringsaktiviteter kan fremme *klassemiljøet*, fordi klassen blir sammensveiset når de løser problemer sammen. Kontaktlæreren til småskoleklassen beskrev også hvor generelt nysgjerrig barna er på hverandre:

> De er så nysgjerrige på hverandre, hvor langt er den kommet? Hvor langt er den kommet? Hvilken oppgave er det den sitter med? (Kontaktlærer)

Strukturering og progresjon i programmeringsopplæring

Klassen som er observert har hatt kodetimer siden de begynte på skolen. Både kontaktlærer, andrepedagog og avdelingsleder fortalte om en kodeplan (årsplan i koding) som alle klassene på skolen følger. Skolen har gjort seg erfaringer om hvordan programmeringsopplæringen bør foregå. Avdelingsleder sa:

> 1. klasse har jo disse Ruby-bøkene. Der de lærer disse ordene om programmering. Du lærer hva er en variabel er, hva en løkke er. Det er også bøker og praktiske oppgaver der du skal lage det i klasserommet, uten å bruke maskin. Da er det litt mer på begrepsforståelsen, som vi kanskje hoppet over og så begynte med når elevene var eldre. Da de startet, brukte vi ikke tid på det. Men det ser man i etterkant, at ja, det var jo lurt å bruke tid på. (Avdelingsleder)

Kontaktlæreren til observasjonsklassen fortalte om språket barna bruker når de programmerer. Hun beskrev at elevene brukte et hverdagslig, veldig enkelt språk, og at de unngikk bruken av programmeringsbegreper fordi det fremdeles er for vanskelig for småskolebarna. Språket er, ifølge klassens kontaktlærer:

> Helt enkelt, og så prøver vi å bruke begreper. Men det synes de er vanskelig. For de skal bruke hverdagsspråket. Det ser vi også spesielt i matematikktimene, og vi prøver å få inn faguttrykk, men det klarer de ikke. De skal bruke det helt enkle. (Kontaktlærer)

Her ser vi at lærerne beskriver elevenes språk som hverdagslig og enkelt. Samtidig er både avdelingsleder og kontaktlærer tydelige på at de ønsker at elevene skal bruke programmeringsbegreper, men at det er vanskelig for barna å internalisere dem.

Oppgavetyper

Avdelingsleder og kontaktlærer trakk frem at klassene arbeider mye programmeringsoppgaver hvor det finnes en gitt oppskrift for hvordan man kan bygge et produkt, f.eks. oppgavene i LEGO WeDo. Avdelingslederen mente at elevene blir gode på å følge oppskrifter, noe som etter hvert kan hemme utviklingen av *kreative* problemløsningsferdigheter. Hun argumenterte for påstanden slik:

> Det vi tenker nå er at vi er veldig gode på å følge oppskrifter. Og det… men det hjelper jo ikke, sant.. da blir man god på å følge oppskrifter. Vi må få de til å lage ting selv. Så det er jo den. Fra å gå fra oppskriftsferdige ting til å utvikle det selv, for det er jo først da man får utvikle seg.. (Avdelingsleder)

Videre pekte hun på at grunnen til at disse oppgavetypene oftest blir brukt, er lærerens *kompetanse*. Hun argumenterte med at når læreren ikke har programmeringskompetanse, er det naturlig at læreren velger oppgaver og verktøy som er oppskriftspreget. Det er gjennomførbart og åpenbart, fordi voksne bruker også oppskrifter når de skal gjøre noe de ikke kan. Hun plasserte kompetanse- og videreutdanningsbehovet inn i en hypotetisk sammenheng, hvor elevene etter hvert har nådd et visst faglig nivå og progresjon i programmeringen:

> De første klassene våre hoppet uti det, enkle programmeringsverktøy var nok. Men nå etter hvert får vi elever som kommer på 7. trinn og har kodet i 7 år, da trenger man kompetanse. Vi må ha kompetanse, det er bare sånn det er. (Avdelingsleder)

Avdelingsleder mente at elevene lærer mer om programmering etter hvert som de får erfaring med det, og utvikler ferdigheter og kompetanse som fremmer problemløsning. For å holde ved like denne progresjonen må lærerne utvikle seg, eller ha kunnskaper om verktøy, arbeidsmåter og andre rammefaktorer som påvirker hvordan elevenes progresjon blir. Det argumenteres altså for behovet for lærere med programmeringskompetanse, gjerne gjennom videreutdanning. Kompetanse blir også avgjørende for å planlegge programmeringsøkter med fornuftige, profesjonsfaglige didaktiske vurderinger.

4.1.2. Samarbeid, veiledning og støtte

Tre faktorer som alle intervjuinformantene mente fremmer utviklingen av problemløsningsferdigheter når elevene arbeider med programmering sammen med andre, var lærerens veiledning, motivasjon og støtte til når de står i problemer. Like viktig er elevenes evner til samarbeid.

Lærerens veiledning og støtte

Både avdelingsleder og lærerne fortalte om sine tanker om den voksnes rolle når barn programmerer. Dette temaet handler om interaksjoner som skjer i kodetimene. Alle intervjuinformantene mente at den voksne var en faktor som kunne fremme elevene sine forutsetninger for å bli bedre problemløsere. Kontaktlærer brukte uttrykket «trykke på noe» for å forklare hvordan hun gjerne gav elevene en ledetråd for å komme videre. Andre måter det støttende stillaset kom til syne var ved hjelp av avgrensninger (ledende spørsmål til elevene) eller emosjonell støtte. Kontaktlærer og andrepedagog var også bevisst på hvordan de skulle opptre som en undrende rollemodell for barna. Kontaktlæreren til klassen beskrev hvordan klassen reagerte når lærerne ikke umiddelbart hadde en løsning på kodingen deres:

> Og da får vi engasjementet til ungene mye mer frem, de blir jo kjempeivrig når ikke vi får det til. Og hvis vi kan spille på at vi ikke får det til, så skal i alle fall de få det til. Sant, så det blir en konkurranse i det og. (Kontaktlærer)

Når barna ser hvordan de voksne modellerer problemløsningsprosessen og ferdigheter som kreves for å bli en god problemløser, er sjansen til stede for at de selv tilegner seg slike problemløsningsferdigheter. Kontaktlærer sa:

> Jeg er av den oppfatning av at du som voksen skal være litt sånn undrende og spørrende. At du skal sammen med elevene prøve å finne ut av svaret. Jeg liker ikke at jeg alltid skal ha svaret, for da mister du litt ut av den gnisten selv, synes jeg. (Kontaktlærer)

Avdelingsleder var også inne på noe av det samme:

> Jeg ser nytten med programmering i klasserommet, å undre seg sammen med elevene. Men jeg er jo ikke teknisk sjøl. Jeg synes det er gøy når jeg får det til. Men når jeg skal programmere Micro:bit, så ser jeg ikke alltid logikken i alt. Men jeg skjønner hvorfor de må lære det. Alle skal ikke bli programmerere, men alle skal vite hva er en datamaskin er, hvordan teknologi virker og så ikke minst problemløsning og det å overføre til det virkelige liv. Det er jo det som er så spennende. (…) Ja, det er alltid noen som synes det er kjekt å hjelpe. De visste etter hvert hvem de skulle spørre. Jeg visste hvem jeg skulle spørre. (Avdelingsleder)

Her kommer den ydmyke holdningen avdelingslederen har til programmeringsferdigheter til syne. Hun erkjenner at hun får utfordringer med programmeringen til tider, men hun ser ressursene hos elevene og spør elevene om løsninger som hun ikke har. Andre elever som observerer at læreren deres spør elevene om tips, vil da erfare at det ikke er farlig å ikke ha en umiddelbar løsning.

Kontaktlæreren til klassen var tydelig på at de voksne prøver så langt det er mulig å unngå å gi direkte løsninger på problemene som barna møter i kodingen sin. I stedet prøver de å:

> Motivere de, prøver å liksom veilede de videre. Hjelper med en oppgave, en tankegang. Gi de et skritt fremover.. Ja, sant… og «prøv nå»! Det er vel egentlig det vi gjør (…) Det er ikke alltid godt å si hvordan vi gjør det! Ikke alltid vi kan hjelpe de. Hjelpe de litt videre med noen ledetråder eller flere spørsmål. Kanskje jeg kan føre noen sånne halvledende retninger eller stiller litt ledende spørsmål. Gir de ikke svaret. Det blir for dumt. (…) Så begynner jeg gjerne å trykke på noen ting. OK, nå går jeg sånn og sånn, også hva gjør jeg videre. Prøver å veilede… Ofte vet jeg det ikke selv altså! Så kan vi det ikke, så må vi finne det sammen. Fordi jeg aner ikke selv. (Kontaktlærer)

Disse tre siste sitatene forteller at lærerne ikke ønsket å gi elevene for mye hjelp, men at elevene fikk ledetråder og støtte, slik at de kunne gjennomføre resten av oppgaven på egenhånd. Avdelingslederen fortalte også om en klasse hun hadde vært i, der:

> **Avd.leder**: elevene rullet rundt på gulvet i frustrasjon fordi at bilen (LEGO Mindstorm) ikke gikk den veien de ville.
>
> **Meg**: Hva gjorde de voksne da?
>
> **Avd.leder**: Nei, da sa vi, her… det er noe i koden din, du må gå tilbake og lete. Og når da bilen går rett vei, så er det utrulig gøy. Eller når de innså at det ikke var nok å «stikke ledningene borti», vi måtte lodde de også, det funka når det funka. Der mestringen er når vi klarte det.

Alle intervjuinformantene beskrev hvordan de mente lærerens måte å støtte og veilede på kunne være med på å fremme elevenes problemløsningsferdigheter. Alle tre var nokså like i sitt syn på at de ikke ønsket å gi elevene svar, men heller støtte dem emosjonelt, gi dem ledetråder eller ledende spørsmål som kunne få elevene videre i arbeidet.

Avdelingsleder var også opptatt av at elevene måtte trene på hvordan de skulle møte motgang. Hun mente det var viktig å utruste elevene, gi dem verktøy for å trene på å møte motgang. Samtidig mente hun at lærere måtte gi elevene oppgaver som læreren vet at elevene kan mestre for å utvikle problemløsningsferdigheter.

> Å gi de mestringsopplevelser så gjør at de øver seg på å tenke at «går jeg på en smell, så kan jeg prøve på noe annet. Det er ikke fordi jeg er dum, men det er bare fordi at dette gikk ikke». Så… det er jo ikke uten grunn at livsmestring er i den nye planen. (Avdelingsleder)

Et av temaene som ble tatt opp i alle intervjuene var hvordan lærerne støttet og veiledet de barna, som kan beskrives slik avdelingslederen gjorde: «elever som ruller seg på gulvet i frustrasjon» når de ikke får til et problem. Lærerne fortalte om ulike måter å støtte elevene på. Som presentert over, var de enige om at å gi elevene ledetråder, «et steg videre» eller ledende spørsmål.

Elevsamarbeid

Klassen som ble observert, var en klasse der samarbeid var innøvd som arbeidsmetode. Elevene satt sammen i grupper, og lærerne deres fortalte om en klasse som ikke har hatt adskilte pulter, eller sittet en og en, siden de startet på skolen. Andrepedagogen sa:

> Elevene bruker nesten mer hverandre enn de bruker lærer når de spør om hjelp. Alltid hverandre først, og hvis det da ikke går, og gjerne flere i klasserommet, før de går til oss etter hjelp. (Andrepedagog)

Kontaktlæreren utdypet:

> Men det anbefaler og oppmoder vi til at de skal gjøre. Hjelpe hverandre før de kommer til oss. Prøv å løs problemet før dere spør. Hvis det ikke går, så kan dere komme til oss. Det er ikke sikkert vi får det til heller. Og da er det enda gøyere. (Kontaktlærer)

Klassens lærere hevdet også at det er viktig at elevene trener på å samarbeide, at det blir en innøvd arbeidsmetode. Lærerne mente dette var viktig for elevenes trygghetsfølelse. Da kunne elevene våge å dele sine utfordringer, og oppleve klassekameratene som meningsfulle lærevenner. Det er nærliggende å tenke at denne tryggheten kommer av et godt klassemiljø, og at samarbeid på tvers av kjønn og vennskap derfor er en arbeidsmåte som fungerer. Andrepedagogen beskrev miljøet for samarbeid og at elevene hjalp hverandre:

> Og gjerne da, da subber de bort, og noen ganger tar de den der «nei, men han har hjulpet henne, så du kan gå til han, for han har allerede hjulpet henne, så gå bort dit. Der er det allerede løst. Da er det bare å subbe bort dit og få seg hjelp». Og dette gjør de vel i alle fag. Og det er helt vanlig. (Andrepedagog)

Samtidig sitter barna for seg selv i timene også. Kontaktlærer forklarte:

> Men det er når de får det veldig godt til.. da sitter de veldig fokusert med sin koding, enser nesten ikke andre. Men sånn, hvis noen begynner å streve, da er det opp og begynne å se: hva er det andre gjør for å løse det? (Kontaktlærer)

Her ser vi hvordan elevene først prøver å løse problemet selv, før de løfter blikket og ser seg rundt etter hjelp fra andre. Kontaktlærer beskriver at det først er når elevene strever, at de ser opp fra eget arbeid. Det kan tyde på at andre problemløsningsstrategier i forkant, og at elevene velger å søke støtte fra andre når utholdenheten begynner å ta slutt.

> Noen ganger jobber de en og en på ulike apper for seg selv, men det har en tendens til at de liksom sklir sammen. Og så begynner de å diskutere, og så finner de løsninger sammen likevel. (Kontaktlærer)

Kontaktlæreren forklarte hvordan språket elevene bruker, er med på å fortelle noe om hvordan de finner feilene i kodene sine. Hun viste til at ofte så hun elever som *peker* på

et problem, og at de ikke klarer å fortelle på en fullstendig og forståelig måte hva problemet er:

> Hvis de er veldig ivrig er det veldig korte ord og setninger. Dersom det er noe de ikke helt får til, er det noen lange, uforklarlige setninger som selv ikke vi alltid skjønner. Også peker de bare, også vet ikke vi alltid hva de vil frem til. (Kontaktlærer)

Dermed ser vi hvorfor lærerne foretrekker fysiske programmeringsverktøy, hvor problemløsningsprosesser som testing og feilsøking blir konkrete og visuelle. Hun mente også at evnen til å formulere seg kom an på hvor engasjerte barna var. Dersom barna var svært ivrige gikk det ofte i setningsfragmenter og ufullstendige setninger. Da kunne de gjerne vise at de var utålmodige i atferden sin, og nappe ipaden med sekvensene eller produktet ut fra hendene til gruppemedlemmene. Noen ganger ble dette en kilde til konflikt mellom barna, og barna trengte voksenstøtte for å løse konflikten. Læreren mente at dette også hang sammen med elevtyper, hvordan elevene er som person.

4.1.3. Problemløsning

Kreativitet

Alle intervjuinformantene var inne på viktigheten av at elevene får utfolde seg kreativt i skaperverkstedet og kodetimene. Avdelingsleder har et perspektiv på teknologi og design som hun setter i en større sammenheng, når hun snakker om et prosjekt om *Ubrukelige roboter* (IKTipraksis.no, 2018), der man bruker Micro:bit som programmeringsverktøy:

> Vi kan ikke løse det store problemet, robotene våre kan bevege en arm. Men vi kan være med i en prosess. (…) Men tankeprosessen, et problem som skal løses, hvordan skal vi løse det. Det er jo det livet handler om. Vi vil møte problem, men vi må ha verktøy til å løse de. Bryte de opp. Sortere dem. (…) Og da tenker jeg at algoritmisk tankegang faktisk er med på å… og ikke minst alle de andre ferdighetene (…) Samarbeid, lære å lære, stå i problemet, kognitiv kondis, innovasjon, se nye muligheter, metakognisjon og selvregulering. Ja… du kan jo egentlig ta hele lista med 21. århundrets ferdigheter og putte de inn i en kodetime. (Avdelingsleder)

Hun knyttet også praktisk estetiske fag, der mulighetene for kreativ problemløsning er store, med programmeringsaktiviteter og mulighetene barna får der:

> Og på sløyd sant, det er masse gøye ting vi har laga før, men nå kan vi lage de enda gøyere. (…) Bamsene fikk lysende øyne, som skiftet farge.. med Micro:biten. (Avdelingsleder)

Feilsøking

Alle intervjuinformantene trakk frem mulighetene for at elevene får arbeide med feilsøking som en av faktorene som fremmer problemløsningsferdigheter når barna programmerer. Lærerne (i småskoleklassen) poengterte at barna var små i de laveste klassetrinnene, og at det derfor ikke var avansert feilsøking de jobbet med. De beskrev likevel hvordan barna stiller hverandre eller lærer spørsmål, som tegn på at de feilsøker programmene sine. I tillegg fortalte lærerne om hvordan elevene «plukker ut enkeltdeler av en kode, og bytter den ut med en annen del». Avdelingsleder fortalte at elevene gjerne spør om hjelp i disse fasene av programmeringen:

> **Avdelingsleder**: Elevene sier gjerne «det skjer ikke», eller «hvorfor skjer ikke det? Eller at de mener at de har gjort den rette koden, og likevel hopper katten ut i vannet. «Hvorfor hopper han ut i sjøen i stedet for å gjøre det?» spør de gjerne lærer eller hverandre. Hvis det er på skjerm da. De tror at de har gjort rett, også gjør ikke de det. Også hvorfor… feilsøking.
>
> **Meg**: Hvordan takler de den situasjonen?
>
> **Avdelingsleder**: De blir jo mye bedre på det. Og noen er jo veldig god på det å faktisk sitte i problemet. Og andre lærer jo det å heller bruke ekstra tid første gangen. For da… da går det litt fortere. At man… det nytter ikke å «skynte seg», fordi at du må tilbake. De har jo skjønt at de kommer seg ingen vei hvis de ikke feilsøker. Vi sier ikke «det er greit, gå videre». Du får ikke ny oppgave før du faktisk har klart det.

I utdraget fra det transkriberte intervjuet over er det gjengitt en del av samtalene vi hadde om feilsøking. I eksemplene avdelingsleder gav, ble det tydelig at elevene hadde sett for seg et resultat som ikke ble slik de hadde tenkt. Eksemplene viser at elevene stiller spørsmål om det som skjer i programmet.

Fikling

Alle intervjuinformantene mente at elevene benytter seg av *prøve og feile*- metoden når de programmerer, men at dette er noe barna må øve på, og at de blir bedre til det etter hvert. Dessuten trakk kontaktlæreren frem viktigheten av at lærere også etterstreber å bruke denne metoden når de skal hjelpe elevene. Når barna ser at voksne også strever, lærer de at det er greit å ikke få til noe med en gang. Lærerne mente at ved å ikke skrive opp typiske, tradisjonelle læringsmål mål på tavla, får man elevene mer med på den

fiklende, lekende tilnærmingen til programmering. Da får elevene undre seg, og produsere noe som utfordrer deres egen kreativitet. Det blir en balansegang mellom å gi instrukser og gi elevene rom for kreative løsninger, som eksempelet der elevene skulle produsere noe som kunne spre pollen.

Både kontaktlærer og andrepedagog var opptatt av at kodetimene skulle inneholde mye lek og lekpregede aktiviteter. De grunngav ikke hvorfor denne tilnærmingen til programmering var viktig. De fremhevet at elevene koste seg med programmeringsøktene, og at det ikke var noen elever som opplevde frustrasjon i form av manglende selvkontroll i disse timene:

> **Meg**: Opplever dere noe frustrasjon?
>
> **Andreped.**: Ja, men ikke noe sånn usunn holdt på si. Bare sånn «eg kommer ikkje meg videre»-frustrasjon eller «dette får ikke jeg til! Hvem kan hjelpe?»
>
> **Meg**: Klarer de å løse den frustrasjonen selv eller må de ha hjelp fra dere?
>
> **Andreped.**: Nei, de trenger ikke hjelp fra oss til det.
>
> **Meg**: Hva velger de da, for å løse det problemet når de står fast?
>
> **Kontaktlærer**: da bruker de hverandre!
>
> **Meg**: Helt av seg selv, naturlig å velge andre?
>
> **Begge lærerne**: ja, ja, ja!

Sitatene over tyder på at lærerne mente at elevene i den aktuelle klassen har selvkontroll og er løsningsorienterte nok til å finne gode løsninger, uten lærernes hjelp. Elevene skal trene på utholdenhet i et problem, å stå i et problem over tid, slik at de trener på å finne nye kreative løsninger. Dersom de løper for raskt til en medelev for hjelp, vil de ikke trene på utholdenhet, og å stå i et problem. Det kan være vanskelig for lærerne å følge med på.

Dele inn i delproblemer

Kontaktlærer beskrev elevenes evner til å kunne dele et problem til mindre deler, *decomposition*. Hun poengterte at det var viktig at elevene hadde trent på å arbeide slik:

> Hvis det er noe de har gjort flere ganger før, da er det lettere. Vi har brukt Bee-boten en del, da er det lettere. Men de første gangene klarte ikke de å «nå skal du gå to frem og så skal du snu til venstre eller snu til høyre». Da klarte ikke de det. Men nå begynner de så smått å ta inn det da.. (Kontaktlærer)

4.2. Analyse av observasjon

I utgangspunktet brukte jeg kodeboken fra intervjuene til analyse av observasjoner (s. 52). Ikke alle kodene fra intervjuene ble funnet igjen i observasjonene. Etter at funn fra observasjonene var kodet, fant jeg det fornuftig å presentere data fra observasjoner under to deloverskrifter. Elevene arbeidet på forskjellige måter med testing og feilsøking da de brukte disse verktøyene. Det kan se ut til at det er enkelte verktøy som egner seg bedre til å arbeide med testing og feilsøking enn andre, derav delkapittelet *Testing og feilsøking med fysiske programmeringsverktøy.* Under observasjonene gjorde jeg umiddelbare analyser av viktigheten av lærere og medelevers samspill og veiledning for at elevene skulle utvikle problemløsningsferdigheter. Derfor heter det andre delkapittelet *Veiledning, samarbeid og støtte*. Under *Veiledning, samarbeid og støtte* har jeg konstruert nok en deloverskrift som handler om *elevenes felles forståelse av oppgaven*. Jeg fant nemlig en sammenheng mellom elevenes testing- og feilsøkingsaktiviteter og hvor vidt elevene hadde en felles forståelse av oppgaven, noe som igjen har en sammenheng med hvordan elevsamarbeidet fungerer og *lærerens instruksjon og veiledning*.

I alle observasjonsøktene med programmeringsverktøy ble små grupperom eller gang mellom grupperom og klasserom brukt som læringsareal. Skolens eget skaperverksted, hvor skolen oppbevarer ulike programmeringsverktøy, ble ikke benyttet i løpet av observasjonsøktene. I løpet av observasjonsperioden (se 3.5.4.) brukte klassen tre ulike verktøy, Bee-bot (figur 2), Blue-bot (figur 3) og LEGO WeDo (figur 4). Elevene har brukt Bee-bot siden første klasse. Det er første gang elevene blir presentert for Blue-bot, og læreren har planlagt at elevene selv skal finne ut hvordan den virker. I den korte instruksjonen hun gav til klassen, sa hun: «Dere må finne ut hvordan den virker, jeg vet ikke!»

4.2.1. Testing og feilsøking med fysiske programmeringsverktøy

I en undervisningsøkt arbeidet to elevgrupper med Blue-bot, to andre grupper arbeidet med Bee-bot. Resten av klassen arbeidet med oppgaver på en pedagogisk nettressurs i klasserommet.

Gruppestørrelsen var 2 elever i hver gruppe. De fire gruppene hadde hver sine kart. Kartet viste ei øy i havet, og Bee-boten og Blue-boten skulle finne en skatt som var på øya. Skatten var et bilde som elevene la på en av koordinatene. Nedenfor blir interessante fortellinger fra denne kodetimen presentert. Elevene har fått fiktive navn.

Figur 7: Kart over øya hvor skatten er gjemt. Bilde hentet fra lekolar.no

Blue-bot:

Blue-boten ligger på E1. Ina trykker på Arne sin ipad. Hun lager sekvens med to frem og en til venstre. Sekvensen virker tilfeldig og intuitiv, hun trykker *hurtig* på blokkene på ipaden. Tester programmet sitt.

> Arne: nei!! Du må se… han har gått… bare to fremover. Han må gå en til… (Arne lener seg over mot Ina og peker på et sted inni sekvensen, som han studerer på appen. Så griper han ipaden og river den ut av hendene på Ina. Arne snur seg og ipaden litt bort fra Ina) Arne: Sånn… du må legge et skritt til der. (Arne gir ipaden tilbake til Ina). Ina drar en ny input inn i sekvensen. Begge ser bort på Blue-boten. Løfter Blue-boten tilbake til sjørøverskipet. De tester sekvensen. Ina følger programmet på ipaden (som visualiserer hvor langt programmet er kommet i sekvensen i form av fete markerte ruter over inputen som utføres). Blue-boten kjører fra sjørøverskipet og inn kartet og stopper i E4. Blue-boten er vendt fremover, slik at i starten av neste sekvens må elevene først snu den mot venstre, før de kjører den fremover. Begge barna fester blikket på ipaden. De blar seg gjennom ulike kodegrupper, og velger til slutt en blokk som

tydelig viser en stor «snupil» mot venstre. De sletter ikke koden, men bytter bare ut en blokk i sekvensen med en annen.

I fortellingen med Blue-boten legger barna inn sekvenser på ipaden. De må veksle mellom å følge med på roboten på gulvet, og sekvensen de har plottet inn på ipaden. Når objektet på gulvet gjør en uventet handling, feilsøker elevene programmet på ipaden. De lokaliserer hvor feilen er, og korrigerer den. De har arbeidet med stor iver og entusiasme, og begge ønsket å løse problemet. Her er det ikke en som dominerer over den andre, selv om Arne velger uheldige løsninger som kunne ført til elevkonflikt: han roper ut på en måte som kunne gjøre Ina irritert, i tillegg til at han river ipaden ut av hendene på henne uten å forklare hvorfor. Ina reagerer ikke på dette, det er mulig hun kjenner guttens karakter. Hun sitter tålmodig og venter på at Arne skal gi henne ipaden tilbake.

Bee-bot:

Nils og Nina sitter på gulvet med Bee-boten. De har kartet og koordinatene til Bee-boten foran seg. Kartet viser ei øy i havet (figur 7), og Bee-boten skal finne en skatt som er på øya. Skatten er et bilde som elevene legger på en av koordinatene.

De setter Bee-bot på startpunktet på kartet (sjørøverskipet). Denne gangen er skatten ved siden av vulkanen. Oppgaven er å programmere roboten til skatten. Nina sitter ved siden av en boks LEGO-brikker. Hun tar ut to brikker. Ser ned på kartet og legger den ene LEGO-biten på C1. Min vurdering er at hun gjør dette for å definere og avgrense oppgaven sin.

Nina: Her skal den svømme i land…

Nils: Han kan jo slåss med haiene først da!

Nils bøyer seg over Bee-boten og lager en sekvens med fire piler rett frem. Tester sekvensen. Bee-boten begynner å kjøre mot blekkspruten (feil retning). Nils løfter opp roboten før sekvensen er utført. Nina sitter passivt og ser på.

Nils: ups…

Nina: du må jo snu han først!

Nils setter ned Bee-boten når den er ferdig med sekvensen.

Nils… nope. Jeg kan bare snu han!

> Han vender Bee-boten fysisk, slik at han slipper å programmere retning. Så trykker han samme sekvens som tidligere, og Bee-boten kjører til haiene. Nina protesterer ikke på denne løsningen.

For å rette opp i feilen vendte Nils Bee-boten slik at sekvensen han hatt satt sammen, fungerte optimalt neste gang han testet. Dette er en enkel løsning. Den involverer ikke programmering. Nils slipper unna feilsøking og testing. En alternativ løsning kunne være å programmere roboten til å snu seg.

LEGO WeDo og kreative utfoldelser

I en observasjonsøkt ble LEGO WeDo benyttet som verktøy. Samme organisering som ble benyttet under økten med Bee- og Blue-bot ble brukt, men gruppene var nye. Klassen arbeidet i grupper på 3 eller 4 elever, og at grupperom og gang ble benyttet som læringsareal. Elevene skulle designe et produkt som kunne samle nektar, og de fikk beskjed fra lærer om å tenke kreativt og smart. Enkelte grupper ble ikke ferdig med designet, og fikk ikke startet kodingen. De brukte hele økta, ca. 50 min., på å designe produktet sitt. Denne oppgaven skulle utfordre elevenes kreativitet i designfasen, fordi elevene skulle produsere et produkt som kunne samle nektar. Programmeringsaktivitetene i etterkant av designet virket heller tilfeldige og intuitive. Kreativiteten i oppgaven ble også kraftig begrenset av at elevene fulgte bruksanvisninger og oppskrifter i LEGO WeDo-appen når de bygget produktet sitt. Likevel var det rom for kreative innslag. Disse kreative innslagene var ikke et nødvendig krav for å utføre oppgaven, men enkelte grupper evnet å knytte inn kreative innslag i oppskriften:

> Gunnar: Jeg skal kode han…
>
> (Han programmerer roboten til å kjøre sakte fremover)
>
> Gunnar: Jeg vet ikke hvordan han kan gå fortere…

Gunnar har ipaden og har tatt ansvar for kodingen uten at noen har sagt noe om dette, Liv og Silje bygger bilen. Gunnar ser ned på ipaden. Han virker konsentrert og blar gjennom ulike blokker for å øke hastigheten på roboten. Han legger til blokker i sekvensen og bytter de ut igjen når de ikke gir ønsket effekt. Gunnar blir ikke synlig frustrert, og viser evne til å holde ut i et problem over tid. Han leker seg med de forskjellige blokkene. Til slutt gir han opp å få få roboten til å gå raskere. I stedet begynner han å fikle med å legge inn ulike lyder i sekvensen sin. Gunnar leker seg med

forskjellige lyder som han legger inn i sekvensene sine. Jentene reagerer ikke på at han gir opp målet med å få roboten til å gå raskere.

> Liv: Hvilken vei kjører bilen?
>
> (Gunnar snur ipaden mot henne og viser sekvensen sin)
>
> Gunnar: Sånn… Hvorfor kan du ikke feste lys på hengeren, sånn vi får lys på bilen?
>
> Silje: Ja!

Sammen fant elevene kreative løsninger, og det er gjennom dialogen at kreativiteten vokser utover de begrensningene for kreative valg som oppskriften i appen gir. Gunnar viste det tydelig da han spurte gruppen om de skulle feste lys på hengeren. Dette var ikke en del av oppskriften.

4.2.2. Veiledning, samarbeid og støtte

Hvor viktig lærerens støttende stillas er i programmeringsøktene kom sterkt til syne under en programmeringsøkt med Blue-bot.

> Elevene arbeidet i par, Lise og Lars skulle arbeide sammen. De brukte tid på å finne ut hvordan de skulle programmere Blue-boten i appen, for dette var første gang barna brukte Blue-bot. Læreren gav dem bare instruksjonen: «Dere må finne ut hvordan den virker, for jeg vet ikke». Før økta startet forklarte læreren meg at hun bevisst ville legge frem verktøyet til barna på denne måten. Etter hvert fikk de bevegelse på Blue-boten på gulvet, og det har tatt litt tid. Ansiktsuttrykket på Lise sa at hun begynner å bli lei oppgaven. Lars sleit med å holde ut og fokusere på problemet på dette tidspunktet. Han ble distrahert av en boks LEGO-klosser, og etter hvert snudde han seg vekk fra programmeringsverktøy og Lise, og begynte å bygge LEGO i stedet for å fokusere på oppgaven. Lise viste tydelig at hun var oppgitt over Lars. Hun ropte navnet hans flere ganger, og kommenterte alderen hans. Lars tok seg sammen og satte seg ved siden av Lise. Hun gav ham ipaden, han laget en tilfeldig sekvens. Sekvensen hadde ingen effekt på roboten på gulvet. Lars gikk ut av appen og startet «Spør Siri» i stedet. Lise så oppgitt ut.
>
> Ved siden av elevene satt et annet læringspar. Den ene gutten så hva som skjedde og spurte: «Lars, hvorfor vil du ikke hjelpe?» Lars overhørte dette. Da kom læreren: «Hva skjer her da? Må jeg finne noen som kan ta over ipaden

din?» Hun satte seg på gulvet, ganske nært elevene og kartet. Likevel krøp Lars og Lise så tett inntil henne som de kunne, på hver sin side av lærere. Lærer spør elevene: «Skal han først til skogen? Hvordan skal vi få ham til skogen?»

Det at læreren kom inn i rommet, og satte seg ned med elevene hadde stor effekt på Lars sin selvregulering der og da. Han hentet seg inn igjen og satte seg ved siden av læreren sin. Lars klarte å sette sammen en korrekt sekvens som førte roboten til skogen, og han viste også større engasjement for oppgaven når læreren deltok i aktiviteten. Han brukte ikke ord når han satte sammen sekvensen, og forklarte heller ikke formålet med sekvensen.

Da læreren forlot rommet, sklei også Lars sin selvregulering ut igjen:

Lars: Ai ai, ikke den veien!

Lise: men da må du få han til å snu seg.

Det andre læringsparet i rommet krøp bort til Lars og Lise og fulgte med på det som skjedde. En elev spurte Lise og Lars: «Finner dere ikke pilen? Da kan dere gjøre sånn…» De fire elevene satt sammen rundt kartet. De fulgte med på om Lars klarte å snu retningen på objektet på gulvet. Etter hvert la Lars fra seg ipaden og begynte å kravle rundt på gulvet og bygget videre med LEGO. Etter det var det ingen som prøvde å hente Lars inn igjen, og han lekte med LEGO resten av kodetimen. Lise fikk hjelp av det andre læringsparet. Gutten som tidligere hadde prøvd å hente Lars inn, visste hvordan retningen kunne endres. Han byttet ut den delen av sekvensen som var feil. De andre i gruppa observerte løsningen, men Lars fikk ikke med seg dette.

At den voksne spiller en viktig del av hvordan elevene arbeider med testing og feilsøking når de programmerer, kom tydelig frem i datamaterialet. Dessuten ser vi at selvregulering og evne til å holde ut i et problem blir viktig for å utvikle problemløsningsferdigheter, og at enkelte barn trenger støtte for å trene opp utholdenhet.

I en kodeøkt med LEGO WeDo ble lærerens inngripen etter at elevgruppen hadde bygget ferdig og kodet en kort stund viktig. Hun konkretiserte og avgrenset oppgaven, noe som kan ha blitt avgjørende for hvordan resten av kodeøkten ble:

Lærer: Hva har dere laget nå da?

> Elevene: en bie (elev 1 viser en LEGO-figur som skal forestille en bie til læreren).. og en bil som skal frakte honningen…
>
> Lærer: Samler han også nektar?
>
> (elevene nikker.. sier ingenting)
>
> Lærer: kjører han?
>
> Elevene: jaa… ja. Nå skal vi ha lys og lyd…
>
> Lærer: Kan du få bilen til å stoppe etter 10 sek? Kan du få han til å lage lyd samtidig?

Resten av økten bruker barna på å fikle med inputs som kan få bilen til å gå i nøyaktig 10 sekunder. Det så ut til at elevene trivdes i denne delen av kodeøkten. Elevene talte høyt sammen når de testet om bilen stoppet etter 10 sekunder, og alle barna var engasjert. De delte på å utføre kodingen, selv om dette var noe de avtalte seg imellom. De testet ut ulike inputs. Da de måtte endre koden fordi den ikke passet med betingelsen deres, slettet de ikke hele sekvensen. De byttet bare ut den delen av koden som var feil. På den måten lokaliserte de problemet, delte problemet i deler og viste evner til testing og feilsøking. De gav ikke opp før de klarte å få bilen til å stoppe etter 10 sekunder.

Felles forståelse av oppgaven

Det så ut til at det var viktig at alle gruppedeltakerne var forente om hva målet med programmeringsdelen av oppgaven i LEGO WeDo skulle være. Derfor har jeg valgt denne overskriften for å belyse viktigheten av en felles forståelse i forbindelse med testing og feilsøkingsaktiviteter og LEGO WeDo. Bee-bot og Blue-bot kom med tilbehør som forenklet og tydeliggjorde strategiene med å bryte ned problemer for elevene. Ved hjelp av koordinater, kunne elevene telle og orientere seg i problemløsningen. Kartet gjorde programmeringen mer konkret og enklere å forholde seg til, enn da elevene programmerte LEGO WeDo. Samtidig ble øktene med Bee- og Blue-bot preget av at elevene hadde et tydelig mål med programmeringen sin, og både lærer og elev hadde en forståelse av hva som var en korrekt utført oppgave. Roboten skulle til skatten.

Kontaktlærer var opptatt av at elevene skulle få frie tøyler til å produsere meningsfulle produkter, og det kom godt frem i instruksjonen elevene fikk i forkant av kodetimen med LEGO:

> Kan dere lage noe som kan få pollen til å spre seg? Tenk at vi ikke hadde hatt så mye vind her hos oss… Tenk smart og kreativt!

Da elevene programmerte LEGO WeDo brukte de ikke kart eller andre visuelle uttrykk for å orientere seg i programmeringen. Oppgaven var å designe noe som kunne spre pollen og samle nektar. Deretter skulle barna kode produktet. Jeg er usikker på om elev og lærer hadde en felles forståelse av hva som ville være en korrekt utført oppgave før økten nærmet seg slutten. Denne felles forståelsen ble mer tydelig først etter at lærer hadde vært inne og veiledet i gruppa, *etter* at elevene hadde designet bilen og programmert den rundt i rommet med en lekende tilnærming, uten at det ble observert tegn på testing og feilsøkingsprosesser.

De gruppene som hadde bygget ferdig, begynte å programmere bilen. Elevene hadde ikke fått instruksjon på hvordan de skulle programmere etter at de hadde bygget ferdig. De satt på gulvet og fiklet med inputs. Aktivitetene virket intuitive, ikke planlagte. Elevene hadde ikke noe mål eller mening med kodingen, og jeg observerte ingen grupper som feilsøkte sammen når de programmerte produktet sitt med LEGO WeDo, *før* de fikk veiledning av lærer. De testet kodene sine, men da koden var utført (enten slik barna hadde til hensikt eller ikke) skjedde en av disse hendelsene:

1. barna slettet koden og laget en helt ny kode uten sammenheng med den forrige
2. ble distrahert av andre i gruppa som snakket til dem om samtaleemner som ikke var relevant for programmering eller oppgaven.
3. begynte å fikle med LEGO, bygge nye figurer

Først etter at læreren hadde vært inne og veiledet gruppen, ble testing og feilsøkingsprosesser en mer fokusert del av problemløsningen. I starten hadde de ikke et definert mål, men meningen med sekvensene i programmeringen ble til etter hvert som elevene lekte med produktet sitt. Det var *etter* at de hadde designet bilen slik at den fungerte funksjonelt, og lekt litt med å kjøre den fremover i forskjellige hastigheter, at elevene la inn farger på lys og lyd inn i sekvensene. Gjennom en fiklende tilnærming til programmeringen, ble kreative variasjoner lagt inn i produktet.

Etter at elevene fikk veiledning av lærer, der et nytt mål ble definert, fokuserte elevene på å få bilen til å stoppe etter 10 sekunder og at lysene skulle lyse i rekkefølgen rødt – blått – gult. Her begynte elevenes feilsøkingsferdigheter å vise seg, samtidig som den fiklende, lekende tilnærmingen ebbet ut. De talte sekunder sammen, gikk inn i

sekvensen og byttet ut inputs når koden var feil, og ble mer samstemt om hva som gjorde oppgaven vel utført.

5. Diskusjon

Det ser ut at datainnsamlingen med påfølgende analyser pekte ut noen nøkkelelementer som blir viktige når det skal graves dypere for å finne svar på følgende problemstilling og forskningsspørsmål:

Problemstilling: Hva kjennetegner programmeringsøkter i småskolen der barn får utvikle problemløsningsferdigheter?

F1: Hva mener lærere fremmer og hemmer småskoleelevers muligheter for å utvikle problemløsningsferdigheter sammen med andre når de jobber med programmering?

F2: Hva kjennetegner situasjoner der barn får mulighet til utvikling av problemløsning når de jobber med testing og feilsøking?

Fra intervjuene tar jeg med meg følgende stikkord som lærerne fortalte om: *planlegging og gjennomføring av programmeringsøkter*, *samarbeid*, *veiledning og støtte* og *problemløsning*. Fra observasjonene tar jeg med videre *testing og feilsøking med fysiske programmeringsverktøy* og *veiledning, samarbeid og støtte*. Ved hjelp av disse elementene fra analysen har jeg definert følgende tre tema som skal drøftes for å få en dypere forståelse av fenomenet og spørsmålene som er stilt i denne masteroppgaven:

- Støttende stillas
- Problemløsning og programmering
- Fysiske programmeringsverktøy

Sosiokulturelle læringsteorier viser hvordan barn *lærer sammen med andre og gjennom interaksjoner*. Tidligere forskning på programmering og problemløsning i småskolesammenheng viser betydningen *more capable peers* har for hvordan elever løser problemer de har vansker med. Dessuten peker flere studier (Kelleher & Pausch, 2005; Palmèr, 2017; Fessakis et al., 2013) på at fysiske, konkrete programmeringsverktøy forenkler problemløsningsprosesser for småskolebarn. Teori om den proksimale utviklingssonen, støttende stillas og medierende verktøy viser

hvordan barn lærer ved hjelp av interaksjoner mellom mennesker og/eller verktøy. Når problemløsning i programmering er vanskelig, støttes barnet ved hjelp av voksne, medelever eller selve programmeringsverktøyet. Derfor er det naturlig å dreie drøftingen rundt disse faktorene; støttende stillas, problemløsning i programmering og fysiske programmeringsverktøy.

5.1. Det støttende stillasets betydning

Gruppesammensetninger

Kontaktlærer mente det var avgjørende for læring at alle elevene var aktive i læreprosessen og at gode gruppesammensetninger ble viktig for elevaktiviteten. Avdelingsleder argumenterte for at homogene grupper fungerte best for samarbeidet, og på den andre siden mente lærerne til observasjonsklassen at varierende gruppesammensetninger var fornuftig. Homogene grupper argumenteres med at elevene arbeider på likt nivå, dermed unngår avdelingslederen passive elever. Gruppesammensetninger er ikke noe som er begrenset til kun å gjelde programmeringsopplæring, men en faktor som generelt påvirker elevsamarbeidet, uavhengig hvilke oppgaver elevene skal løse. Den proksimale utviklingssonen viser hvordan elevene lærer gjennom interaksjoner sammen med noen som kan mer enn dem selv (Vygotsky, 1978, s. 86). Dersom gruppene er homogene, kan det se ut til at det blir det vanskelig for elevene ved første øyekast å lære av noen som kan mer enn dem. De vil sitte sammen med noen som er like faglig sterke som dem selv, og dermed ikke ha noen å strekke seg mot.

Samtidig pekte lærerne til observasjonsklassen på at klassen er ei elevgruppe med godt læringsmiljø og et hjelpeklima. Dersom en elevgruppe er homogen, kan derfor læreren i klassen eller andre elevgrupper være *more capable peers*. Elevene kan lære mer enn de klarer alene, dersom det finnes en kultur og et trygt klassemiljø for å spørre andre om hjelp er tilstede i elevgruppa. Om man ser på dette i en større kontekst, faller det naturlig å tenke at et godt og trygt klassemiljø fører til et godt læringsmiljø. Læringsmiljøet er, i likhet med gruppesammensetninger, ikke begrenset til kun å gjelde samarbeidslæring i programmeringsøkter, men vil være en faktor som kan fremme eller hemme elevenes muligheter for læring i alle fag gjennom hele skolehverdagen. Uansett, blir det viktig at læreren som skal planlegge programmeringsøkter har tenkt gjennom hvordan han/hun ønsker gruppesammensetningen, og vurderer hva som vil egne seg i forhold til klassemiljø, verktøy og elever med ulike evner.

Lærerens støtte og veiledning

Fra observasjonsøkten med Bee- og Blue-boten fant jeg at lærerne ikke brukte mye tid på instruksjon i starten av timen, slik vi gjerne er vant med fra tradisjonelle skoletimer. Elevene var ikke flasker som læreren skulle fylle med kunnskap i disse timene. I stedet vektla læreren veiledning *der elevene trengte det* underveis i problemløsningen. Den proksimale utviklingssonen skulle tilsi at det er fornuftig å starte med en introduksjon, der den lærende skal lære av en mer kompetent person (Vygotsky, 1978). Først skal den lærende lære sammen med andre, og så på et indre plan. I økten med Bee- og Blue-bot fantes ingen «flaskepåfyllings»-instruksjon. I stedet ble økten satt i gang med at elevene fikk en veldig kort, nesten diffus, presentasjon av verktøyet («finn ut hvordan Blue-boten virker, jeg vet ikke!»), og så var tanken til læreren at hun skulle veilede ved behov.

Denne formen for problemløsning fungerte for de elevene som evnet å holde ut i et problem, som f.eks. Ina og Arne (s. 67), men problemløsningen ble utfordrende for barn som Lars (s. 70), som strevde med utholdenhet og motivasjon for problemløsning. Wood et al. (1976) beskrev hvordan det støttende stillaset skulle fremme *direction maintenance*, der den lærende skulle støttes og veiledes for å finne en mening med å fortsette problemløsningen. Lars ble vervet til oppgaven og fant mening når læreren satte seg ned med han, men det var synlig vanskelig for Lars å holde frem med problemløsning uten voksenstøtte. Lars er et eksempel på det Sipitakiat & Nusen (2012) konkluderte med; småskolebarn strever med konsentrasjon og utholdenhet i programmering, når de skal løse et problem uten støtte. Derfor fremmer lærerens nærhet og *direction maintenance* den lærende sine muligheter for å utvikle problemløsningsferdigheter, særlig om barnet strever med utholdenhet og motivasjon for å løse problemet.

I kodetimene med Bee- og Blue-boten valgte læreren å dele inn i grupper og levere ut verktøyet som skulle brukes. Lærerens veiledning og instruksjon kom inn ved behov i problemløsningen. Det ble elevenes oppgave å finne ut av hvordan verktøyet fungerer, som kan sammenlignes med en nedenfra og opp- tilnærming. Dette vil sannsynligvis fremme en *fiklende* tilnærming til programmering og problemløsning (Resnick & Rosenbaum, 2013), fordi fremgangsmåten for problemløsningen ikke er klart definert for elevene. Læreren gav ikke elevene en trinnvis, tydelig forventning om hva de skulle gjøre for å løse oppgaven. Læreren organiserte på denne måten timen slik at elevene

skulle være mest mulig aktive, og ikke sitte passive for å observere eller lytte til lærer. På denne måten fikk læreren aktivisert elevene i problemløsningsprosesser mye tidligere enn dersom hun hadde brukt første del av kodetimen med en forelesende tilnærming. Dette var en gunstig start på timen for barn som har ferdigheter for *selvstendig* problemløsning, som Ina og Arne, og selv benytter hensiktsmessige ferdigheter for problemløsning. De viste at de mestret denne tilnærmingen til problemløsning godt. Samtidig ble læreren avgjørende for at barn som Lars skulle bli værende i problemet, og være et stillas for han, slik at han også fikk stimulere sine ferdigheter for problemløsning og veilede han i problemløsningshjulet. Læreren benyttet *ledende spørsmål* (slik intervjuinformantene sa at de gjorde) for å hjelpe han med å *identifisere* problemet og for at Lars skulle få sjanse til å identifisere problemet, samle informasjon og generere ideer for å finne en tilfredsstillende løsning på problemet:

Skal han (Blue-bot) først til skogen? Hvordan skal vi få ham til skogen? (Kontaktlærer)

Under observasjon av Blue-bot måtte elevene orientere seg med et nytt verktøy de ikke har brukt før. Problemløsningshjulet til Wallace & Maker (Wallace et al., 2005) viser hvordan å *samle og organisere* informasjon er det første steget i problemløsningsprosessen. At læreren kastet verktøyet ut til elevene kunne forsvares med denne fasen av problemløsningsprosessen. Elevene måtte orientere seg og bli kjent med verktøyet (Blue-bot), og koble det til kjent kunnskap, som var kartet og koordinatene til Bee-boten. Dette tilbehøret kjente elevene fra før. På den måten ble også oppgaven avgrenset, et viktig element i teorien om støttende stillas (Wood, et al., 1976), fordi elevene opplevde at noen av elementene i oppgaven var kjente og gjennomførbare. Kartet og koordinatene kjente de fra før.

Lærerne til observasjonsklassen argumenterte for at det var fornuftig å møte elevene og deres problemer med en undrende tilnærming. I tillegg spilte de gjerne på elevenes konkurranseinstinkt. Dette kan knyttes sammen med prosesser i instruksjonsprosessen til Wood et al. (1976), særlig den delen som handler om å motivere og støtte. Når lærerne undrer seg sammen med elevene, spiller på konkurranseinstinktet deres eller later som om de ikke får det til, er måter å motivere elevene på. Alle intervjuinformantene mente at det er viktig at lærerne undrer sammen med elevene.

Voksne er rollemodeller for elevene, og barna lærer av mer *capable peers* (Vygotsky, 1978).

Alle intervjuinformantene var enige om at de ikke ønsket å gi direkte løsninger til elevene. Wood, et al. (1976) har argumentert for hvordan støttende stillas kan føre til læring hos barn. De var opptatt av avgrensing av oppgaven, at man skulle tilpasse oppgaven slik at elevene hadde forutsetninger for å utføre oppgaven. Lærerne som ga elevene et hint, ledende spørsmål eller som «trykket på noe» (læreren fysisk trykket på en blokk eller knapp på roboten) for å gi elevene et spark fremover, *avgrenset* problemet for elevene, slik at elevene igjen kunne verves til oppgaven når utholdenheten tok slutt (Wood, et al., 1976). Samtidig bærer disse strategiene for lærerveiledning et element av modellering, som Wood et. al (1976) også var opptatt av. Elevene observerer da at en del av problemet blir løst for dem, og det er mulig at eleven på egen hånd hadde vært inne på noe av det samme som læreren modellerte.

Lærerne fortalte at elevene ikke brukte de voksne så mye når de behøvde hjelp til programmering. De brukte hverandre, og de brukte hverandre ofte og mye. Om læreren trekker seg tilbake og lar elevene selv stå for all hjelp og veiledning, kan det være uheldig. For det første kan det, i ytterste konsekvens, føre til at elevene ikke øver seg på elementære faser i problemløsningshjulet, fordi de sitter passive og mottar løsninger servert på fat. For det andre risikerer læreren at han/hun i utgangspunktet ikke får en fullstendig oversikt over problemløsningsprosessene elevene arbeider med, om deres *computational thinking* utvikler seg eller ikke i løpet av timen. Lye & Koh (2014) trekker bl. annet frem Fessakis et al. (2013) som en studie der det blir antydet at elevene utvikler problemløsningsferdigheter av seg selv. Dersom læreren ikke følger med på elevenes dialoger når de er i samspill, kan nok Lye & Kohs (2014) påpekning om at problemløsningsferdigheter utvikles av seg selv gjelde her også. Samtidig har observasjonene i denne studien vist og bekreftet andre studier (Fessakis, et al., 2013; Lye & Koh, 2014; Sipitakiat & Nusen, 2012) som viser hvor viktig læreren blir for at elevene skal utvikle ferdigheter innen problemløsning. Elevene kan ikke erstatte lærerens profesjonelle kompetanse, som hun i denne studien viser i form av et støttende stillas for elevene.

Ledende spørsmål, velvalgte hint (uten å avsløre en løsning) og emosjonell støtte for å motivere barna til videre problemløsning, var strategier intervjuinformantene brukte når

de skulle hjelpe elever som sto fast i problemer under programmering. Strategiene som lærerne valgte, viste seg flere ganger under observasjonene når elever behøvde hjelp. Spesielt viktig ble læreren for eleven som slet med utholdenheten da han skulle programmere Blue-Bot. Han ble distrahert av LEGO-klosser og andre apper på ipaden (Spør Siri). Læreren satte seg ned ved siden av gutten, og dette er en form for emosjonell støtte, som fører til at læreren blir et støttende stillas (Meyer & Turner, 2002). Gutten krøp så tett inntil læreren sin som han kunne. Læreren *vervet* eleven til oppgaven igjen med å spørre om roboten skulle til skogen. Måten læreren opptrer på og strategiene hun velger for å hjelpe eleven, kan plasseres i et sosiokulturelt læringsperspektiv, og spesielt som en del av instruksjonsprosessen til Wood, et al. (1976). Læreren hjalp eleven til å hente seg inn igjen, og å fokusere på oppgaven. Ifølge instruksjonsprosessen til Wood, et al. (1976) kalles denne delen av instruksjonsprosessen for *direction maintenance.* Her skal læreren synliggjøre veien videre i problemet, slik at eleven opplever en mening med å fortsette oppgaven. For denne eleven ble det vanskelig å fortsette oppgaven når læreren forlot han. Dermed ser vi hvor avgjørende læreren var for at eleven skulle holde ut lenge nok til at han klarte å løse problemet. Når læreren forsvant, mistet gutten på ny fokuset på problemløsningen. Derfor er lærerens evne til å verve elevene som strever med utholdenhet i programmering, og hennes evne til *direction maintenance*, en faktor som kan fremme, men også hemme elevenes problemløsningsferdigheter i programmeringsøkter.

Samarbeid

Observasjonsklassen var en klasse som ofte samarbeidet og ba hverandre om hjelp. Ut ifra en sosiokulturell tilnærming passer disse uttalelsene godt med hvordan barn lærer. Det er samtidig viktig å reflektere rundt hjelpekulturen i klassen, begrepene mestring og appropriering og evnen til å holde ut i et problem. I problemløsningshjulet til Wallace & Maker (Wallace et al., 2005) er å generere ideer og beslutningstaking to av fasene i hjulet. Her er det viktig at elevene øver seg på å fundere og vurdere flere løsninger. Dersom de velger å spørre andre elever om hjelp, har ikke disse elevene faglig tyngde, kunnskap om eller intensjoner om å fungere som støttende stillas på en hensiktsmessig måte. Med det mener jeg at elever som hjelper medelever, gjerne vil komme med «lettkjøpte løsninger» som ikke utfordrer den lærende, og bare viser en løsning. Dermed er det grunn til å tro at eleven som trenger hjelp, kopierer en mer kompetent elev, uten at appropriering skjer (Wertsch, 1998). Dette vil heller ikke utfordre barnets utholdenhet

og evnen til å vurdere forskjellige løsninger. Sagt med andre ord, blir ikke eleven *metakognisjon* stimulert på en hensiktsmessig måte. Metakognisjon er en avgjørende ferdighet for problemløsning, fordi metakognisjon skal kontrollere og styre de andre egenskapene og ferdighetene som kreves for være en god problemløser (Wallace, et al., 2005). Når det er sagt, vil den eleven som skal hjelpe, lære mer av å presentere sine løsninger for andre. Det å fortelle om sine løsninger og presentere for andre en sentral del av problemløsningshjulet til Wallace & Maker (Wallace et al., 2005). Når elevene forteller om sine egne valg og løsninger, bruker de språket som et medierende artefakt. Språket hjelper dem til å oppnå en bedre forståelse av løsningen de selv har valgt og gjennomført.

5.2. Programmering og problemløsning

Fikling

Under observasjonene fra LEGO WeDo ble det observert aktivitet som passer inn under fikling. Denne tilnærmingen til programmering skjedde da elevene hadde designet produktet sitt ferdig. Målet var å designe et produkt, å følge en forhåndsbestemt fremgangsmåte. Kodingen i etterkant skjedde uten et klart mål. Programmeringen kunne ikke umiddelbart knyttes til testing og feilsøkingsprosesser. Slik jeg så det, var lærerens *ubevisste* manglende avgrensing av oppgaven, instruksjoner om videre arbeid etter designfasen og nære støtte grunnen til at dette skjedde. Læreren ønsket kreative produkter som kunne samle nektar. Elevene hadde fått en oppgave, som skulle utfordre elevene kreativt, noe som gjorde at programmeringsarbeidet etter designfasen ble tilfeldig. Da kodearbeidet ble tilfeldig, var den enkleste og behageligste løsningen å la være å gå inn feilsøkingsprosesser. Barna hadde ikke fått en videre instruks.

Analysene fra observasjonene viste at elevene benyttet en fiklende tilnærming til programmering da de ikke hadde et klart, definert mål med programmeringen sin (LEGO WeDo). Ved å velge en lite konkret målstyring i undervisningsøktene, blir trolig aktivitetene preget av en mer fiklende tilnærming til programmering. Resnick & Rosenbaum (2013) beskriver dette som en nedenfra-og-opp tilnærming. Målet blir ikke gitt på forhånd, og avgrenser ikke problemløsningen, slik en målstyrt ovenfra-og-ned-tilnærming ville ført til.

Fiklingen var synlig i elevsamarbeidet *før* lærer kom og definerte videre gang i oppgaven. Fikling var en tilnærming elevene benyttet seg av etter at produktet var ferdig designet og elevene *ikke lenger hadde et tydelig mål å arbeide mot*. Etter hvert

kom læreren og avgrenset og tydeliggjorde oppgaven for dem, og dermed fungerte hun som et støttende stillas og laget en ramme rundt oppgaven (Wood, et al., 1976). Etter at oppgaven ble avgrenset og elevene ble styrt mot et tydelig mål, som var å stoppe bilen etter 10 sekund, forsvant den fiklende tilnærmingen til problemløsningen. Ut fra dette eksempelet, ser det ut til at testing og feilsøking skjer tydeligere eller er lettere å observere i programmeringsaktiviteter der elevene har en felles forståelse av hva de ønsker med programmeringen. Det betyr at dersom elevene arbeider mot et tydelig mål, vil testing og feilsøking lettere kunne skje, fordi elevene blir opptatt av å finne ut om de har nådd målet. Da blir testing og feilsøking naturlige problemløsningsstrategier. På den måten kan tydelige mål med programmeringen fremme testing- og feilsøkingsaktiviteter, som er en *computational practice* (Brennan & Resnick, 2012).

Prinsippene som ligger bak fikling og problemløsningshjulet (TASC), viser en vesentlig forskjell i måten disse tilnærmingene til problemløsning forholder seg til *planlegging* av mulige løsninger på. En elev som *fikler* med sekvensene/programmene sine, vil være opptatt av en lekende tilnærming til problemløsning. Som jeg har drøftet i avsnittet over, kan testing og feilsøking være strategier som er enklere å ta i bruk for elevene, når de vet hva målet med testingen er. Ut ifra det Resnick & Rosenbaum (2013) skriver, vil den lærende heller ikke være opptatt av å planlegge og tenke gjennom løsningen sin han han benytter en fiklende tilnærming. Derfor er det grunn til å tro at *intuisjonen* vil styre en del av valgene som blir tatt. Intuisjon er derfor en faktor som kjennetegner programmeringsaktiviteter der barn benytter seg av *computational practices* som testing og feilsøking.

Wallace & Maker (Wallace et al., 2005) skriver om hvor viktig intuisjon er i problemløsningsprosessen. Samtidig hevder de at når elevene skal ta beslutninger, må «the course of action (…) be planned and the possible solution trialled» (Wallace, et al. 2005, s. 42). Her ser vi at teorien som er valgt ut i dette masterprosjektet ikke er entydig i synet på hvor viktig intuisjon og planlegging er i problemløsningsfasen.

Avsnittene over viser hvordan målstyrte oppgaver kan fremme testing og feilsøkingsaktiviteter hos elevene. Samtidig står dette i en kontrast til en fiklende tilnærming til programmering. Vi ser og hvilken rolle intuisjon og planlegging har for problemløsningen, og at planlegging og fikling kan stå i kontrast til hverandre. Dette kan igjen tyde på at det finnes mange strategier som brukes for å løse problemer på, og

eleven må derfor få mulighet til å utvikle sine *metakognitive* evner, som er en sentral problemløsningsferdighet (Wallace et al., 2005), slik at de selv kan vurdere hvilke strategier, tilnærminger og ferdigheter som trengs i en del av en oppgave.

Prøve og feile

Fra observasjonene fra LEGO WeDo gjorde jeg noen interessante analyser av Gunnar:

> *«Han virker konsentrert og blar gjennom ulike blokker for å øke hastigheten på roboten. Han legger til blokker i sekvensen og bytter de ut igjen når de ikke gir ønsket effekt. Gunnar (…) viser evne til å holde ut i et problem over tid. Han leker seg med de forskjellige blokkene. Til slutt gir han opp å få roboten til å gå raskere. I stedet begynner han å fikle med å legge inn ulike lyder i sekvensen sin. Gunnar leker seg med forskjellige lyder som han legger inn i sekvensene sine. Jentene reagerer ikke på at han gir opp målet med å få roboten til å gå raskere.»* (s. 66)

Her ser vi hvordan Gunnar gjentatte ganger *prøver* å få roboten til å gå raskere, men *mislykkes* så mange ganger at han til slutt gir opp. Han viser ingen synlig frustrasjon, men velger å gi opp målet sitt og pensler over på et nytt mål når han ikke opplever mening med målet lenger (jf. *direction maintenance*). Han bruker tiden på å leke seg med de ulike blokkene, og det ser ut til at han er mest interessert i handling. Han *grubler* tilsynelatende lite, og spør heller ikke gruppemedlemmene sine om hjelp. Det er ingen gruppemedlemmer eller voksne som oppmuntrer eller hjelper han med å gjennomføre målet han først hadde. Et mer kompetent gruppemedlem eller en lærer kunne vervet Gunnar inn igjen i oppgaven med å få roboten til å gå raskere, og fungert som Gunnars støttende stillas (Wood, et al., 1976). Når Gunnar ikke får hjelp, oppmuntring eller støtte, velger han den mest behagelige løsningen (som blir å jobbe mot et nytt mål).

Fessakis et al. (2013) fant at elevene programmerte ut fra en prøve og feile-strategi i sine analyser. Dette samsvarer også med det jeg fant i mine analyser, aller mest tydelig i observasjonene med Bee- og Blue-bot. Her ble prøving og feiling utført med medierende artefakter som støttet og synliggjorde problemløsningen på gulvet med elevene, i form av kart og koordinater. Det ser ut til at koordinatene som følger med Bee- og Blue-bot hjelper elevene i problemløsningen, konkretiserer og bryter ned problemet for dem.

Problemløsningshjulet og sentrale ferdigheter for problemløsning

Flere av de sentrale elementene i problemløsningshjulet og problemløsningsferdighetene kan finnes igjen i programmeringsøktene med Bee- og Blue-bot, og også gjennom LEGO WeDo. Likevel er det enkelte forskjeller på verktøyene, som gjør at ferdighetene som knyttes til problemløsning viser seg på forskjellige måter.

Avdelingslederen mente at enkelte programmeringsoppgaver i seg selv kan bli en faktor som kan hemme utviklingen av problemløsningsferdigheter gjennom programmering. Dersom ikke elevene får utvikle sine kreative evner, og i stedet forblir værende i en oppskriftspreget oppgaveløsning, får ikke elevene trene seg på å bli innovative problemløsere. Elevene blir værende i aktiviteter der de ikke får muligheten til *appropriering*. Wallace & Maker (Wallace et al., 2005) argumenterer for hvor viktig kreativitet er for problemløsning, noe de trekker frem som en *vital learning capacity* og under å *generere flere ideer* i problemløsningshjulet. Når barnets oppgave er å følge en oppskrift, blir det vanskelig å bruke sin kreativitet, å generere og vurdere ulike løsninger. Dette kan sammenlignes med det intervjuinformantene kunne fortelle om elever som arbeidet med oppgaver preget av oppskrifter og faste, gitte fremgangsmåter for å løse problemer. På den måten kan oppskriftspregede aktiviteter være med på å hemme utviklingen av problemløsningsferdigheter.

Likevel hevder Resnick & Rosenbaum (2013) at å bygge videre på andres produkt er en *computational practice*. Å bruke om igjen noe som andre har utviklet, og deretter bygge videre på det, er en problemløsningsfremmende aktivitet. Resnick & Rosenbaum brukte begrepene *reusing & remixing* i en programmeringssammenheng. Under observasjonene av LEGO WeDo brukte elevene andres program (jf. oppskriftene). Samtidig bygget elevene videre på andres arbeid, og tilførte produktene egne kreative funksjoner og design (f.eks. lys og lyd). Om man legger teori om stillasbygging til grunn, er det fornuftig å starte med oppskriftspregede problemløsningsoppgaver, fordi oppskriftene i appen fungerer som en *avgrensing* av oppgaven (Wood et al., 1076).

Elevene utførte designfasen i LEGO WeDo stegvis ved hjelp av appen, og derfor var det begrenset hvor tydelig fasene i problemløsningshjulet til Wallace & Maker (Wallace et al., 2005) kom frem. Elevene behøvde f.eks. ikke *planlegge og generere ideer* og valg av løsninger da de designet produktet, for fremgangsmåten var allerede gitt i oppskriften. I praksis valgte bare elevene den arbeidstegningen som de vurderte som

best egnet til å møte lærerens krav om å bygge et produkt for å samle nektar. Dette må defineres som en form for *beslutningstaking*. Småskoleelevene i studien er fremdeles nokså små, og de behøver et stillas for å trene på problemløsning, men dette er en faktor som etter hvert kan hemme utviklingen av problemløsningsferdigheter. Noe av det samme fant også Yu & Roque (2019):

> Step-by-step instructions or specific challenges can help to scaffold children's learning experiences, more open-ended systems can enable children to express ideas, develop their own goals, and pursue more personally meaningful projects. (Yy & Roque, 2019, s. 11)

Det betyr at verktøy som Bee- og Blue-bot og koordinatene som hører til vil fungere som støttende stillas for elevene. I fortellingen som er skissert over om Blue-Boten, starter eleven en hurtig og *intuitiv* sekvens for å komme i gang med programmeringen. Det kan se ut til at intuitive beslutninger og valg er nødvendig for en prøve- og feile-tilnærming til problemløsning. Dette er et velegnet utgangspunkt for en videre feilsøkingsprosess. Når eleven ikke er redd for å gjøre feil, men har en lekende, intuitiv tilnærming til problemet, legger eleven utgangspunkt for problemløsningsprosessen videre og kommer i gang med en løsning.

Samtidig forklarte kontaktlærer hvordan hun gjerne «trykket på noe» (læreren fysisk trykket på en blokk eller knapp på roboten), for å *modellere* denne delen av problemløsningsprosessen for elevene. Dette er fornuftig mtp. teori om problemløsning, fordi intuisjon er en av Wallace & Makers sentrale *learning capacities* (Wallace et al., 2005). Modellering er et av stegene i instruksjonsprosessen til Wood, et al. (1976). Da vil elevene lære at det er greit å trykke på noe, uten alltid å ha mål og mening med valgene de gjør når de programmerer. Derfor er situasjoner med intuitive valg og handlinger et godt utgangspunkt for testing og feilsøking når barn programmerer.

I observasjonen med Nina, Nils og Bee-boten fant jeg et tegn på at elevene overvåket og evaluerte løsningene sine. Wallace & Maker (Wallace et al., 2005) kaller denne fasen i problemløsningshjulet for *implementering*:

> *Bee-boten begynner å kjøre mot blekkspruten (feil retning). Nils løfter opp roboten før sekvensen er utført. Nina sitter passivt og ser på.*
>
> *Nils: ups…*
>
> *Nina: du må jo snu han først!*

Nils setter ned Bee-boten når den er ferdig med sekvensen.

Nils… nope. Jeg kan bare snu han! (s.68)

I stedet for å generere ideer som inkluderer programmering, velger eleven å fysisk snu roboten. Det er mulig at måten barna arbeider med disse robotene på, ikke legger opp til så mye vurdering og overvåkning av problemløsning. Heller ikke fikk jeg observert at barna *evaluerte* prosessen mot målet etter at de hadde fullført programmeringen. Evaluering er også en del av problemløsningshjulet (Wallace et al., 2005). For Bee-boten sin del, finnes det ingen «spor» av løsningen barna har valgt. De får ingen visuell oversikt over inputene. Det blir utvilsomt fordelen med Blue-boten, her kan elevene *se* sekvensene sine når roboten har nådd målet. Likevel observerte jeg ingen barn som brukte tid på evaluere løsningene sine i etterkant av programmeringen. Barna var stort sett mest interessert i å gå videre i oppgaven sin, å planlegge hvordan de skulle få robotene til neste koordinat.

Sammenheng mellom språk, argumentasjon og feilsøking

Det finnes et godt eksempel på sammenhengen mellom argumentasjon, logikk og språk i datamaterialet. Da Ina testet sekvensen sin, kom en annen elev på gruppa inn og feilsøkte løsningen hennes. Denne eleven kom også med et forslag til løsning, selv om det i starten var vanskelig for barnet å forklare løsningen for medeleven sin. Dette måtte også kreve tålmodighet og forståelse fra Ina, hun godtok at det var vanskelig for medeleven sin å forklare med ord hva hun hadde gjort feil:

Arne: Nei!!! Du må se… han har gått… bare to fremover. Han må gå en til…

Arne lener seg over mot Ina og peker på et sted inni sekvensen, som han studerer på appen. Så griper han ipaden og river den ut av hendene på Ina. Arne snur seg og ipaden litt bort fra Ina. Ina ser på han uten å si noe, men smiler svakt.

Arne: Sånn… du må legge et skritt til der.

Arne gir ipaden tilbake til Nina. Nina drar en ny input inn i sekvensen.

Det kan se ut til at eleven i starten brukte sin intuisjon (og ropte ut at noe var feil). I tillegg rev han ipaden utav hendene på Ina, noe som igjen kan tolkes som en intuitiv handling. Det var først da han fikk *tid* på seg til å forstå *logikken* og *argumentasjonen* i løsningen (Wallace et al., 2005), at klarte han å forklare medeleven hva han mente var en mer formålstjenlig løsning. Vyogtsky var opptatt av et indre og et ytre språk

(Vygotsky, 1978). Først skal barna lære på et ytre plan, i en sosial prosess, og deretter skal barnet mestre det på et indre plan. Når Bee-boten blir et medierende artefakt, som forenkler argumentasjonen for elevene, kan de strekke seg mot å forklare hvordan de tenker fordi det fysiske verktøyet støtter dem. Kelleher & Pausch (2005) konkluderte også med at fysiske roboter som Bee- og Blue-bot gjorde resonnering og argumentasjon mer tilgjengelig for småskolebarn.

5.3.Fysiske, visuelle programmeringsverktøy

Det ser ut til at kontaktlærer og andrepedagog foretrekker fysiske programmeringsverktøy i begynneropplæringen. Dette er valg som skolen har gjort etter at de har *erfart* hvilke verktøy som fungerer for deres elever. Dessuten har skolen tilegnet seg kompetanse og oppsøkt nasjonale arrangement der de fikk lære mer om programmering i skolen. Slik det er for barn som lærer av mer kompetente voksne eller medelever (Vygotsky, 1978; Wood et al., 1976), slik kan også fagpersoner lære av noen som kan mer enn dem. Intervjuinformantene fremstår som undrende sammen med elevene, og ydmyke i forhold til sin egen programmeringskompetanse. Den ydmyke, undrende holdningen og viljen til kompetanseheving er utvilsomt noe som fremmer lærernes kompetanse i forhold til programmering i skolen, og hvordan man på best mulig måte kan drive programmeringsopplæring for barn. Dermed blir også dette en faktor som fremmer elevenes muligheter for å utvikle problemløsningsferdigheter.

I alle observasjonsøktene ble fysiske programmeringsverktøy benyttet. Et verktøy ble brukt sammen med app på ipad. Fysiske programmeringsverktøy settes i bevegelse uansett, selv om programmeringen inneholder feil når elevene tester programmet. Dermed blir det lettere for barna å oppdage og korrigere feil. Det blir mer konkret og visuelt for barna. Som nevnt over, finnes det tidligere forskning som argumenterer for at håndgripelige (*tangible)* programmeringsverktøy fremmer elevenes *computational thinking* og generelle læreforutsetninger (Beers, et al., 2014; Palmèr, 2017; Kelleher & Pausch, 2005). Analysene fra intervjuene peker på noe av det samme. Blant annet forteller lærerne om at verktøyene later til å være mer motiverende, fordi de *beveger seg uansett*. Dersom sekvensene er feil, er sjansen stor for at barna uansett får visuelle inntrykk og en opplevelse av årsak-virkning.

Samtidig peker analysene fra intervjuene i dette prosjektet at det kan være en fordel om verktøyet knyttes opp mot en digital flate. At barn profitterer på visualiserte,

konkretiserte sekvenser ble tydelig med ipaden og Blue-boten. Elevene kunne følge sekvensen sin samtidig som roboten beveget seg. Dette var et moment som gjorde at elevene lettere kan bryte ned problemene sine, konkretisere dem og utføre korrigeringer i feilsøkingsprosessen. Derfor blir selve verktøyet et medierende verktøy (Säljö & Moen, 2001), og fungerer som et støttende stillas for elevene. Wood et al. (1976) forklarte hvor viktig det var å avgrense oppgaven for barna. Konkretiserende sekvenser på digital flate vil føre til slik avgrensing. På den måten kan barna får en konkretisering og visualisering over hvordan programmet deres er bygget opp og hvor i utførelsen av sekvensen roboten befinner seg til enhver tid. Her går mine analyser videre fra Beers, et al. (2014). Appen fungerer som konkretiserende, visuell støtte for barna, der de kan få oversikt over sekvensene sine, og vil på den måten vil appen fungere som et medierende verktøy (Säljö & Moen, 2001). Derfor er det fornuftig å la elevene arbeide med roboter som styres ved hjelp av en digital flate.

Progresjon i programmeringsopplæringen

Lærerne tenker progresjon i programmeringsopplæringen. Det å starte enkelt, å *mestre* en tankegang er fornuftig. Da får barna en tilnærming til programmering som er preget av muntlig språk og veiledning sammen med andre, konkrete og lekpregede aktiviteter, før de starter med mer avanserte verktøy som kan strekke dem lenger i læreprosessen og selv utvikle meningsfulle program. I analysene trekkes det frem at barna blir introdusert til konseptet programmering med aktiviteter som ligner språkleker. Denne tilnærmingen kan knyttes opp til begrepene *mastery* og appropriering, ved at man lar elevene blir kjent med en tankegang eller arbeidsmåte på en lekende måte, før de får stadig mer krevende og abstrakte oppgaver. Å gå fra muntlige instrukser til programmering i blokkbaserte programmeringsspråk er en overgang som bør forberedes slik lærerne i utvalget har gjort det. Da legges det til rette for at appropriering skje, fordi elevene etter hvert har tilegnet seg kunnskaper og ferdigheter som de kan bruke for å gjøre programmene til sine egne (Wertsch, 1998). Elevene begynner mykt med språklekpregede aktiviteter der de programmerer hverandre med språk som medierende artefakt, før de etter hvert tar i bruk andre, mer avanserte redskaper.

Alle intervjuinformantene fortalte om hvordan de ønsker at barna bruker programmeringsbegreper, og at skolen har lagt *computational concepts* inn i årsplanene. Samtidig ser det ut til at internalisering av programmeringsbegreper er lettere sagt enn gjort. Vygotsky (1978) argumenterte for at først lærer barn begreper sammen med

andre, deretter skal barnet mestre det for seg selv, på et indre plan. På den måten ble språket et artefakt for læring. I analysene kom det frem at lærerne mener at programmeringsbegreper skal være en del av innholdet i programmeringsopplæringen i småskolen, allerede fra 1. klasse. I intervjuet og i observasjonene fant jeg at internalisering av programmeringsbegreper er vanskelig for elevene å bruke i egen programmering. Appropriering av programmeringsbegreper har ikke skjedd i de tilfellene hvor elevene ikke bruker programmeringsbegreper i programmeringsaktivitetene der de designer egne program (Wertsch, 1998). Det er nærliggende å tenke at dersom barn har internalisert og appropriert begreper som sekvens, løkker og variabler, vil barna bli mer bevisst på problemløsningsprosessen. De har forstått begrepene og innholdet i dem, og dermed vil hele problemløsningsprosessen bli preget av barnas vurdering av hvilke muligheter (inputs) som er mest hensiktsmessig.

6. Avslutning

6.1. Konklusjoner

Problemstillingen i dette masterprosjektet var:

> *Hva kjennetegner programmeringsøkter i småskolen der barn får utvikle problemløsningsferdigheter?*

Ved hjelp av to forskningsspørsmål skulle det være mulig å finne et svar på problemstillingen:

> ***F1**: Hva mener lærere fremmer og hemmer småskoleelevers muligheter for å utvikle problemløsningsferdigheter sammen med andre når de jobber med programmering?*
>
> ***F2**: Hva kjennetegner situasjoner der barn får mulighet til utvikling av problemløsning når de jobber med testing og feilsøking?*

Oppgaven har gjennomgått tidligere forskning på som er aktuell for problemstillingen og relevant teori. Ut fra empirien er det presentert analyser som har vært drøftet, og derfor kan det trekkes noen oppsummeringer som svarer på problemstillingen.

***F1**: Hva mener lærere fremmer og hemmer småskoleelevers muligheter for å utvikle problemløsningsferdigheter sammen med andre når de jobber med programmering?*

Gruppesammensetninger kan både fremme og hemme mulighetene for å utvikle problemløsningsferdigheter sammen med andre i programmeringen. Avdelingsleder argumenterte for hvorfor hun foretrekker homogene elevgrupper. Lærerne i observasjonsklassen er ikke så tydelige på at homogene grupper er å foretrekke, og det kan nok forklares ut fra et klassemiljø som oppleves som trygt og inkluderende. Uansett er gruppeinndelingen en faktor som lærere må vurdere og tenke gjennom for at elevene skal utvikle problemløsningsferdigheter gjennom programmering. Sosiokulturelle læringsteorier forklarer hvordan elever lærere sammen med andre, enten det er egne gruppemedlemmer, andre elevgrupper eller lærer (Vygotsky,1978; Wood, et al., 1976).

Det pekes også på behov for lærere med programmeringskompetanse, slik at lærerne kan være *more capable peers* for elevene, og gjøre didaktiske valg og vurderinger knyttet til programmeringsaktiviteter i klasserommet. Uten kompetanse vil lærerne bruke programmeringsverktøy som baserer seg på å følge oppskrifter. Da støtter lærerne seg på stillaser. Dette vil hemme utviklingen av problemløsningsferdigheter fordi elevene får begrensede muligheter for å generere kreative ideer.

Pedagogene i utvalget oppgir at de er opptatt av å støtte elevene med ledende spørsmål, en ledetråd eller «trykker på en knapp» for å komme videre i et problem i programmeringen. De er også opptatt av at den voksne selv må være en undrende rollemodell, og modellere hvordan man kan stå i en situasjon hvor man ikke har en umiddelbar løsning. Læreren fungerer da som støttende stillas for elevene, og elementer fra instruksjonsprosessen (Wood, et al., 1976) finnes igjen i empirien i dette prosjektet og i annen forskning (Sipitakiat & Nusen, 2012; Lye & Koh, 2014). Dessuten gir læreren verdifull støtte gjennom non-verbalt språk. Gjennom fysisk plassering, at læreren f.eks. setter seg på gulvet sammen med eleven, strekker hun ut en hånd for å hjelpe eleven med å holde ut i et problem. Utholdenhet en ferdighet som er nødvendig for å sortere informasjon og generere ideer når man skal løse et problem (Wallace, et al., 2005).

F2: Hva kjennetegner situasjoner der barn får mulighet til utvikling av problemløsning når de jobber med testing og feilsøking?

Fysiske programmeringsverktøy som brukes sammen med digitale flater der barna kan utføre programmeringen sin, gir mulighet for at elevene får større kontroll over testing og feilsøkingsprosessene sine. Da blir feilsøkingen mer konkret og visuell for barna. Tidligere forskning argumenterer for hvordan fysiske programmeringsverktøy er mer håndgripelig for småskolebarn (Kelleher & Pausch, 2005; Fessakis, et al., 2013). Mine analyser og drøftinger går videre, og konkluderer med at fysiske programmeringsverktøy i kombinasjon med programmering på app gjør feilsøkingsaktiviteter mer konkret. Det blir lettere å lokalisere feilen. Programvaren vil da fungere som et støttende stillas for eleven (Wood, et al., 1976), og støtte eleven kognitivt i læringsprosessen (Meyer & Turner, 2002).

Dette prosjektet bekrefter tidligere forskning som konkluderer med lærerens avgjørende for utvikling av problemløsningsferdigheter gjennom programmering (Fessakis, et al., 2013; Lye & Koh, 2014). Lærerens støtte og veiledning til barn som strever med utholdenhet i problemløsning, og som ikke evner å dele et problem opp i mindre biter, har vist seg veldig viktig for at disse elevene skal delta i programmeringsøkter og aktivt utføre testing og feilsøking. Lærerens oppgave blir å holde eleven på rett kurs, avgrense oppgaven slik at den virker overkommelig for eleven (Wood, et al., 1976) og støtte eleven verbalt eller non-verbalt.

Alle elevene som samarbeider om et prosjekt må ha tydelige forventninger om hva som er en vel utført oppgave for å utvikle problemløsningsferdigheter gjennom testing og feilsøking. Dersom de ikke har det, er sjansen større for at elevene distraherer seg selv eller andre, eller unngår å feilsøke problemer i programmeringen. Samtidig kan dette stå i kontrast til en fiklende tilnærming til problemløsning (Resnick & Rosenbaum, 2013). Det ideelle vil være en kombinasjon, der elevene starter med en lekende tilnærming, benytter *intuisjon* i handlingene sine, og angriper problemet nedenfra-opp. Deretter spisses løsningen mot et endelig produkt, der alle gruppemedlemmene har en forventning om hvordan produktet vil bli når det er ferdig utviklet.

Forskningsspørsmålene er konstruert slik at F1 fokuserer på hva lærere vektlegger og mener fremmer og hemmer utvikling av problemløsningsferdigheter i programmeringsøkter. F2 åpner for å undersøke om det lærerne forteller, stemmer med

praksis. Sammen svarer forskninsspørsmålene på problemstillingen, som dreier seg rundt hva som kjennetegner programmeringsøkter i småskolen, der barn får utvikle problemløsningsferdigheter. Det finnes noen sammenfall i svarene på forskningsspørsmålene som kan brukes for å svare på selve problemstillingen. Det er tydelig at *fysiske programmeringsverktøy* som kan brukes med en konkretiserende flate er et godt utgangspunkt for at barn kan utvikle problemløsningsferdigheter gjennom programmering. Som i all annen undervisning blir *læreren* viktig for elevenes læring, i form av støtte og veiledning. Læreren skal støtte elevene, avgrense og tydeliggjøre problemet, uten å avsløre ferdige, lettkjøpte løsninger for elevene.

6.2. Avgrensninger

Forskningsprosjektet er utformet som et casestudie, som presenterer funn og håndterer analyser fra et enkelt kasus. Utvalget er valgt strategisk, fordi skolen er kommet godt i gang med digitalisering og teknologirike klasserom. Skolen har investert i digitale verktøy, og har erfaringer med programmeringsverktøy og ressurser som kanskje ikke andre skoler har tilegnet seg ennå. Likevel kan naturalistisk generalisering føre til at analyser og funn fra dette prosjektet kan overføres til andre situasjoner. Studien er presentert så transparent som mulig, slik at leseren selv kan vurdere hvor vidt mine funn er gyldige for leseren og hennes situasjon.

6.3. Fremtidig forskning

I den nye læreplanen vil algoritmisk tenking og programmering spille en rolle, selv om det endelige utkastet av læreplanen ikke er publisert ennå. Derfor trenger vi mer forskning som beskriver hvordan elevene kan tilegne seg *computational thinking* i en kontekst med rammefaktorer som er hentet fra norske klasserom. Særlig bør barnas selvregulering i programmeringsaktiviteter undersøkes.

Dessuten bør progresjonen i programmeringsundervisningen problematiseres og vurderes, slik at vi kan få forskning som forteller noe om hvordan progresjonen fra fysiske programmeringsverktøy til blokkbaserte programmeringsspråk og videre til tekstbasert programmering bør legges opp og gjennomføres i grunnskolen. Det er mange skoler som ikke har implementert programmering i årsplanene sine ennå, og for å hjelpe disse skolene på vei, er det nødvendig med forskning som hjelper dem å legge opp en progresjon i programmeringsaktivitetene.

7. Kilder/referanser

Ananiadou, K., & Claro M. (2009). 21st Century Skills and Competences for New Millennium Learners in OECD Countries. doi: 10.1787/218525261154

Angrosino, M. V., & Mays de Perez, K. A. (2000). Rethinking observation from method to context. I Lincoln, N.K.A., & Denzin, Y.S., *Handbook of Qualitative Research* (s. 107-154). Thousand Oaks, CA: Sage.

Belland, B. (2017). *Instructional Scaffolding in STEM Education*: Springer, Cham.

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. doi: 10.1016/j.compedu.2013.10.020

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The Nordic approach to introduciong Computiational thinking and Programming in compulsory education.* doi: 10.17471/54007

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Vancouver, BC, Canada.

Clavio, J. C. & Fajardo, A. C. (2008). Toys as Instructional Tools in Developing Problem-Solving Skills in Children. *Education Quarterly, 66*(1), 87-100.

Computer Hope. (2018, 30.03.2019). Input. Hentet fra https://www.computerhope.com/jargon/i/input.htm

Cresswell, J. W. (2013). *Qualitative inquiry & research design: choosing among five approaches* (Vol. 3). Los Angeles: Sage.

Dede, C. (2010). *Comparing frameworks for 21st century skills*.

Denzin, Y.S. & Lincoln, N.K.A. (1994/2000). Introduction. I Lincoln, N.K.A. & Denzin, Y.S. (Red.), *Handbook of Qualitative Research*. Thousand Oaks: Sage Publications.

Dewey, J. (1938). *Experience and education*. New York: Macmillan.

Di Lieto, M. C., Inguaggiato, E., Castro, E., Cecchi, F., Cioni, G., Dell'Omo, M., Dario, P. (2017). Educational Robotics intervention on Executive Functions in preschool children: A pilot study. *Computers in Human Behavior, 71*, 16-23. doi: 10.1016/j.chb.2017.01.018

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97. doi: 10.1016/j.compedu.2012.11.016

Flyvbjerg, B. (2010). Fem misforståelser om casestudiet. I Brinkmann, S., & Tanggaard, L. (Red.), *Kvalitative metoder* (s. 463–487). København: Hans Rietzels Forlag.

Given, L. M. (2008). Rich data. I L. M. Given (Red.), *The SAGE Encyclopedia of Qualitative Research Methods*: SAGE.

Gold, R. L. (1958). Roles in sociological field observation. *Social forces, 36*, 217-223.

IKTipraksis. (2018, 25.04.2019). Ubrukelige roboter. Hentet fra: https://iktipraksis.iktsenteret.no/content/ubrukelige-roboter-med-microbit

Jonassen, D. H. (2010). *Learning to solve problems: A handbook for designing problem-solving learning environments*.

Kafai, Y. B., & Burke, Q. (2013). Computer Programming Goes Back to School. *Phi Delta Kappan Magazine, 95*(1), 61-65. doi:10.1177/003172171309500111

Kafai, Y. B., Burke, Q., & Resnick, M. (2014). *Connected Code : Why Children Need to Learn Programming*. Cambridge, UNITED STATES: MIT Press.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR), 37*(2), 83-137. doi: 10.1145/1089733.1089734

Kim, M. C., & Hannafin, M. J. (2011). Scaffolding problem solving in technology-enhanced learning environments (TELEs): Bridging research and theory with practice. *Computers & Education, 56*(2), 403-417. doi: 10.1016/j.compedu.2010.08.024

Kvale, S., & Brinkmann, S. (2015). *Det kvalitative forskningsintervju* (Vol. 3). Oslo: Gyldendal Akademisk.

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

Mayer, R., & Wittrock, M. (2006). Problem-solving transfer. I Berliner, D. &. Calfee, R. (Red.), *Handbook of Educational Psychology*. Mahwah: NJ: Erlbaum.

Meyer, D. K., & Turner, J. C. (2002). Discovering Emotion in Classroom Motivation Research. *Educational Psychologist, 37*(2), 107-114. doi: 10.1207/S15326985EP3702_5

NESH. (2006, 11.05.2019). Forskningsetiske retningslinjer for samfunnsvitenskap, humaniora, juss og teologi. Hentet fra:

https://www.etikkom.no/forskningsetiske-retningslinjer/Samfunnsvitenskap-jus-og-humaniora/

NOU. (2015). *Fremtidens skole — Fornyelse av fag og kompetanser*. https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/sec1.

Palmér, H. (2017). Programming in preschool: with a focus on learning mathematics. *International Research in Early Childhood Education, 8*(1), 75-87.

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism, 36*(2), 1-11.

Postholm, M. B. (2010). *Kvalitativ metode. En innføring med fokus på fenomenologi, etnografi og kasusstudier* (Vol. 2). Oslo: Universitetsforlaget.

Postholm, M. B., & Jacobsen, D. I. (2018). *Forskningsmetode for masterstudenter i lærerutdanning* (Vol. 1). Oslo: Cappelen Damm akademisk.

Resnick, M., & Rosenbaum, E. (2013). Designing for Tinkerability. I Honey, M. & Hunter, D.E. (Red.), *Design, make, play* (s. 163-181). Routledge, London.

Sipitakiat, A., & Nusen, N. (2012). *Robo-Blocks: designing debugging abilities in a tangible programming system for early primary school children*. Hentet fra the Proceedings of the 11th International Conference on Interaction Design and Children. Bremen, Germany.

Stake, R. E. (1995). *The Art of Case Studies*. Thousand Oaks, CA.: Sage Publications, Inc.

Säljö, R., & Moen, S. (2001). *Læring i praksis : et sosiokulturelt perspektiv*. Oslo: Cappelen akademisk.

Thagaard, T. (2009). *Systematikk og innlevelse: en innføring i kvalitativ metode* (3. utg.). Bergen: Fagbokforl.

Tjora, A. (2010). *Kvalitative forskningsmetoder i praksis.* Oslo: Gyldendal Akademisk.

Utdanningsdirektoratet. (2019). Algoritmisk tenking. Hentet fra https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2018). *Kjerneelementer i matematikk*. Hentet fra https://hoering.udir.no/Hoering/v2/197?notatId=358.

Utdanningsdirektoratet. (2019). *Høring- læreplaner i matematikk.* Utdanningsdirektoratet på oppdrag fra Kunnskapsdepartementet. Hentet fra https://hoering.udir.no/Hoering/v2/343.

Vygotsky, L. S. (1978). *Mind in society. The development of higher Psychological Processes*. Cambridge, Massachusetts: Harvard University Press.

Wallace, B., & Adams, H. B. (1993). *Thinking Actively in a Social Context; TASC*. Oxford, UK: AB Academic Publishers.

Wallace, B., Maker, J., Cave, D., & Chandler, S. (2005). *Thinking Skills and Problem-Solving - an Inclusive Approach : A Practical Guide for Teachers in Primary Schools*. Abingdon, Oxon, UNITED KINGDOM: David Fulton Publishers.

Wertsch, J. V. (1998). *Mind as Action*. New York, NY: Oxford University Press.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 2.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences, 366*(1881), 3717-3725. doi: 10.1098/rsta.2008.0118

Wolcott, H. F. (2008). *Ethnography- a way of seeing* (Vol. 2). Lanham: AltaMira Press.

Wood, D., Bruner, J. S., & Ross, G. (1976). THE ROLE OF TUTORING IN PROBLEM SOLVING*. *Journal of Child Psychology and Psychiatry, 17*(2), 89-100. doi: 10.1111/j.1469-7610.1976.tb00381.x

Yin, R. K. (2009). *Case Study Research: Design and Methods*: SAGE Publications.

Yu, J., & Roque, R. (2019). A review of computational toys and kits for young children. *International Journal of Child-Computer Interaction*. doi: 10.1016/j.ijcci.2019.04.001

8. Vedlegg

Vedleggsliste:

1. Intervjuguide for lærere og avdelingsleder
2. Oversikt over koder og kategorier i dataanalyse
3. Kvittering fra godkjent NSD-søknad
4. Informasjonsskriv og samtykkeskjema til lærere
5. Informasjonsskriv og samtykkeskjema til foresatte
6. Mal for observasjonsnotat

Vedlegg 1: Intervjuguide for lærere og avdelingsleder

Læringsmiljø/motivasjon

- Hvordan vil du beskrive det generelle klassemiljøet/læringsmiljøet?
    - Trygge elever, samarbeid, grupperinger?
- Hvordan kan du beskrive klassen sin vilje og evne til hjelpe hverandre med programmering og vise/fortelle om sine produkter og løsninger?
- Hva gjør du for å sikre at alle elevene er aktive i læringsprosessen?

Språk

- Hvordan snakker barna sammen gjennom øktene?
    - Relasjon elev-elev, elev-lærer
    - Hvordan opplever du at barna formulerer seg når de ber om hjelp?
        - Språk, ordforråd, tegn til systematikk, logisk resonering, sette ord på problemet?
- Hvordan kan du beskrive elevenes vilje og evne til å begrunne og forklare valg og løsninger som de finner for andre?

Proksimale utv.sone

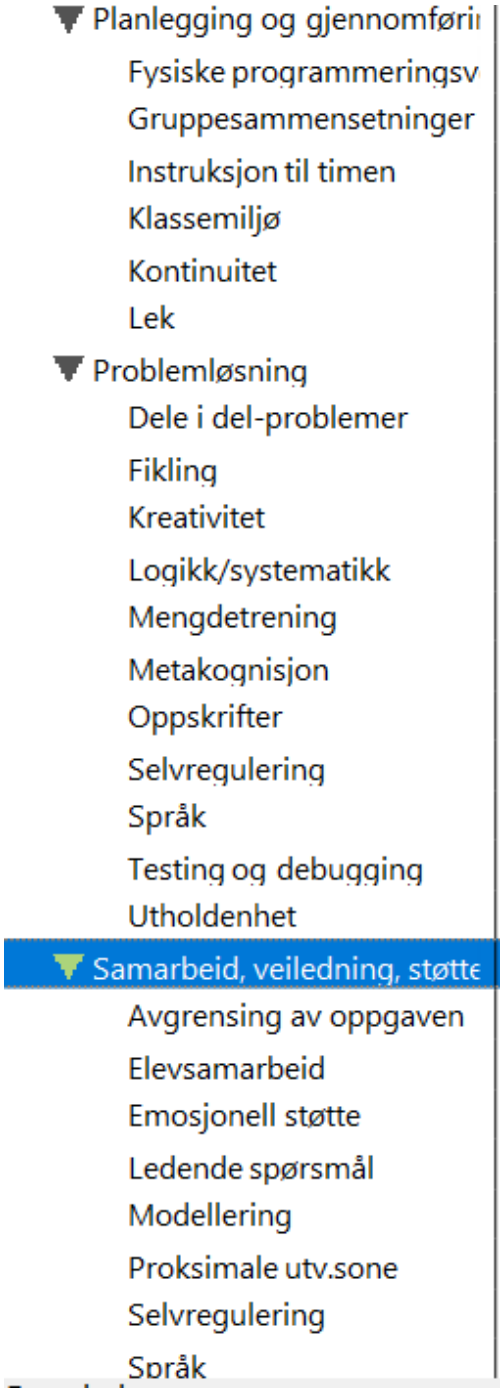
- Hvordan hjelper de voksne barna for å komme videre når de programmerer?
    - Spm.formuleringer
- Hvordan reagerer barna på introduksjonen/veiledningen de får i starten av timene?
    - F.eks. at elevene får introduksjon/starthjelp fra lærer, og så løser problemer selv i egen koding..
    - Motivert for innsats?
- Hvordan reagerer barna på introduksjon/veiledning som de får underveis i aktiviteten?
    - Når de har fått et problem? Evne til å ta i mot hjelp om de er frustrert eller ikke er motivert for å ta i mot hjelp
    -

- Forholdet mellom sterke og svake elever.
    - Hvem samarbeider?
    - Hvilket samarbeid fungerer?
- Hvordan hjelper elever som mestrer programmering andre elever?
- Hvilken læringseffekt ser du av at elevene hjelper hverandre i disse aktivitetene?

Elevaktivitet/problemløsning

- Hvordan vil du beskrive dine elevers evner til å handle intuitivt/prøve og feile/fikle med programmene sine?
    - Hvordan forholder barna seg til hverandre når de programmerer? Observerer de hverandre, ber om hjelp, sitter på egen tue?
- Hvilke tegn kan du/har du observert hos elevene for å se at de tenker logisk og systematisk?
- Hvordan vil du beskrive at elevene evner å bryte ned problemer til del-problemer?

Vedlegg 2: Oversikt over koder og kategorier i analysearbeidet



Vedlegg 3: Kvittering for godkjent NSD-søknad

NSD NORSK SENTER FOR FORSKNINGSDATA

NSD sin vurdering

Prosjekttittel

Programmering og problemløsning i småskolen

Referansenummer

501949

Registrert

19.02.2019 av Christina Hemnes - 095652@stud.hvl.no

Behandlingsansvarlig institusjon

Høgskulen på Vestlandet / Fakultet for lærerutdanning, kultur og idrett / Institutt for pedagogikk, religion og samfunnsfag

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Anders Grov Nilsen, anders.nilsen@hvl.no, tlf: 53491511

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Christina Hemnes, 095652@stud.hvl.no, tlf: 95974122

Prosjektperiode

01.02.2019 - 31.12.2019

Status

21.03.2019 - Vurdert

Vurdering (1)

21.03.2019 - Vurdert

Det er vår vurdering at behandlingen av personopplysninger i prosjektet vil være i samsvar med personvernlovgivningen så fremt den gjennomføres i tråd med det som er dokumentert i meldeskjemaet med vedlegg 21.03.19, samt i meldingsdialogen mellom innmelder og NSD. Behandlingen kan starte.

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til NSD ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilke type endringer det er nødvendig å melde:

https://nsd.no/personvernombud/meld_prosjekt/meld_endringer.html

Du må vente på svar fra NSD før endringen gjennomføres.

TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til 31.12.19.

LOVLIG GRUNNLAG
Prosjektet vil innhente samtykke fra de registrerte til behandlingen av personopplysninger. Vår vurdering er at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse som kan dokumenteres, og som den registrerte kan trekke tilbake. Lovlig grunnlag for behandlingen vil dermed være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 bokstav a.

PERSONVERNPRINSIPPER
NSD vurderer at den planlagte behandlingen av personopplysninger vil følge prinsippene i personvernforordningen om:

- lovlighet, rettferdighet og åpenhet (art. 5.1 a), ved at de registrerte får tilfredsstillende informasjon om og samtykker til behandlingen
- formålsbegrensning (art. 5.1 b), ved at personopplysninger samles inn for spesifikke, uttrykkelig angitte og berettigede formål, og ikke behandles til nye, uforenlige formål
- dataminimering (art. 5.1 c), ved at det kun behandles opplysninger som er adekvate, relevante og nødvendige for formålet med prosjektet
- lagringsbegrensning (art. 5.1 e), ved at personopplysningene ikke lagres lengre enn nødvendig for å oppfylle formålet

DE REGISTRERTES RETTIGHETER
Så lenge de registrerte kan identifiseres i datamaterialet vil de ha følgende rettigheter: åpenhet (art. 12), informasjon (art. 13), innsyn (art. 15), retting (art. 16), sletting (art. 17), begrensning (art. 18), underretning (art. 19), dataportabilitet (art. 20).

NSD vurderer at informasjonen om behandlingen som de registrerte vil motta oppfyller lovens krav til form og innhold, jf. art. 12.1 og art. 13.

Vi minner om at hvis en registrert tar kontakt om sine rettigheter, har behandlingsansvarlig institusjon plikt til å svare innen en måned.

FØLG DIN INSTITUSJONS RETNINGSLINJER
NSD legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

For å forsikre dere om at kravene oppfylles, må dere følge interne retningslinjer og/eller rådføre dere med behandlingsansvarlig institusjon.

OPPFØLGING AV PROSJEKTET
NSD vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene avsluttes i tråd med den behandlingen som er innsendt og dokumentert.

Lykke til med prosjektet!

Kontaktperson hos NSD: Kjersti Haugstvedt
Tlf. Personverntjenester: 55 58 21 17 (tast 1)

Vedlegg 4: Informasjonsskriv og samtykkeskjema til lærere

Forespørsel om å delta i intervju i forbindelse med masteroppgave

Jeg heter Christina Hemnes og jobber med en masteroppgave i IKT i læring ved Høgskulen på Vestlandet (HVL). Prosjektet handler om programmering i småskolen, og hvordan barn kan utvikle problemløsningsferdigheter gjennom programmering. Formålet med intervjuene av lærere er å hente inn informasjon om hvordan lærere mener barna kan lære problemløsningsferdigheter gjennom programmering sammen med andre, og hvordan denne læringen skjer best. For å finne svar på dette, ønsker jeg intervju med lærere som har erfaring med programmeringsundervisning i småskolen. Behandlingsansvarlig institusjon er Høgskulen på Vestlandet (HVL), avdeling Stord.

Intervjuet vil være et halvstrukturert intervju. Det betyr at hovedspørsmålene er planlagt og skrevet inn i en intervjuguide, men at vi har mulighet til å snakke mer utdypende om interessante emner som informanten forteller om. I intervjuguiden ligger spørsmål om hvordan læreren legger til rette for problemløsning gjennom programmeringsaktiviteter, ut ifra et sosiokulturelt perspektiv på læring. Fokuset vil derfor være på hvordan barna arbeider sammen, språk, hjelp/veiledning fra voksne og andre barn i elevgruppa og motivasjon/læringsmiljø.

Det er masterstudent og veiledere ved HLV som vil ha tilgang til personidentifiserbart data (navn i forbindelse med samtykkeerklæring og lydopptak). Konfidensialiteten bevares ved at alle navn og personidentifiserbart materiale vil bli anonymisert i resultatet og i selve publikasjonen. Lærerne i intervjuene vil få pseudonymer. Jeg har taushetsplikt.

Jeg vil bruke en digital lydopptaker i intervjuet, og ta notater når vi snakker sammen. Lydopptaket tas med min mobiltelefon, og lastes opp på HVL sin trygge forskningsserver, der jeg som student har eget lagringsområde. Dette vil bli gjort innen et døgn etter at vårt intervju er gjennomført. Opptaket på telefonen vil bli slettet når opptaket er overført til forskningsserveren. Lydopptak slettes etter transkribering (senest 30.12.2019). Intervjuets varighet er beregnet til ca. en til to timer, og vi blir enige om tid og sted for intervjuet.

Det er frivillig å delta i intervjuet. Dersom du ønsker å trekke deg fra intervjuet, står du fritt til det. Du trenger ikke oppgi grunn til hvorfor du trekker deg. Du kan be om innsyn, retting, sletting og begrensning av data som er samlet inn fra intervjuet ditt. Dersom du mener at datainnsamlingen ikke foregår på en forsvarlig måte/slik den er skissert i dette skrivet, kan du klage til Datatilsynet.

Dersom du har spørsmål til intervjuene eller masteroppgaven eller ønsker å trekke deg, kan du kontakte meg på mobil: 95974122 eller mail: 095652@stud.hvl.no . Mine veiledere ved HVL kan kontaktes, ved Anders Grov Nilsen, på anders.nilsen@hvl.no. HVL sitt personvernombud er Halfdan Mellbye. Kontaktinfo: personvernombud@hvl.no. Tlf. 55 30 10 31.

Studien er meldt til NSD (Norsk senter for forskningsdata).

Vennlig hilsen Christina Hemnes

Samtykkeerklæring:

Jeg har mottatt informasjon om masterstudien om programmering i småskolen, og ønsker å delta i intervjuet.

Underskrift: Dato/sted:

Vedlegg 5: Informasjonsskriv og samtykke til observasjon

Forespørsel om samtykke til å observere barnet ditt/deres i forbindelse med masteroppgave

Jeg heter Christina Hemnes og jobber med en masteroppgave i IKT i læring ved Høgskulen på Vestlandet (HVL). Prosjektet handler om programmering i småskolen, og hvordan barn kan utvikle problemløsningsferdigheter gjennom programmering. Jeg ønsker å observere barn som jobber med programmering/koding i skolen. Formålet med observasjonene er å hente inn informasjon om hvordan barna kan lære problemløsningsferdigheter gjennom programmering sammen med andre, og hvordan denne læringen skjer best. For å finne svar på dette, ønsker jeg observasjon av elever som jobber med koding i kodetimer på skolen. Behandlingsansvarlig institusjon er Høgskulen på Vestlandet (HVL), avdeling Stord.

Jeg har skissert et observasjonsnotat i forkant av observasjonene. I dette notatet vil det bli listet opp forskjellige emner for observasjonene: hvordan barna arbeider sammen med problemløsning, hvordan de snakker sammen, hjelp/veiledning fra voksne og andre barn i elevgruppa og motivasjon/læringsmiljø. Jeg noterer det jeg ser og hører i dette notatet. Jeg vil ikke bruke elevnavn i notatet. I etterkant av observasjonene vil forskningsdata bli oppbevart på HVL sin egen forskningsserver, der jeg som student har et eget lagringsområde.

Det er masterstudent og veiledere på HVL som vil ha tilgang til personidentifiserbart data (navn på foreldre/foresatte i forbindelse med samtykkeerklæring). Konfidensialiteten bevares ved at alle navn og personidentifiserbart materiale vil bli anonymisert i observasjonsnotatet, resultatet og i selve publikasjonen. Elevene får pseudonymer (ved fiktive navn eller «elev 1», «elev 2», osv.). Jeg har taushetsplikt.

Det er frivillig å delta i observasjonene. Dersom du ønsker å trekke barnet fra observasjonene, står du fritt til det. Du trenger ikke oppgi grunn til hvorfor du trekker henne/han. Du kan be om innsyn, retting, sletting og begrensning av data som er samlet inn fra observasjonene. Dersom du mener at datainnsamlingen ikke foregår på en forsvarlig måte/slik den er skissert i dette skrivet, kan du klage til Datatilsynet. Prosjektet avsluttes innen 30.12.219. All data slettes ved prosjektslutt.

Dersom du har spørsmål til intervjuene eller masteroppgaven eller ønsker å trekke deg, kan du kontakte meg på mobil: 95974122 eller mail: 095652@stud.hvl.no . Mine veiledere ved HVL, ved Anders Grov Nilsen, kan kontaktes på anders.nilsen@hvl.no. HVL sitt personvernombud er Halfdan Mellbye. Kontaktinfo: personvernombud@hvl.no. Tlf. 55 30 10 31.

Studien er meldt og godkjent av NSD (Norsk senter for forskningsdata).

Vennlig hilsen Christina Hemnes

Samtykkeerklæring:

Jeg har mottatt informasjon om masterstudien om programmering i småskolen, og ønsker at mitt/vårt barn kan være informant i observasjonene.

Foresattes underskrift: Dato/sted:

Foresattes underskrift: Dato/sted:

Vedlegg 6: Feltnotater, mal for observasjonsnotat

Hva observeres	Hva skjer? Kommentarer
Læringsmiljø/motivasjon - antall delaktige elever, antall passive elever i ei gruppe -aksept for å foreslå løsninger -hvem deler valg og løsninger -får alle elevene forslagene sine hørt Av medelever -vilje og evne til å stå i aktiviteten over lengre tid Språk - Hvordan snakker barna sammen gjennom øktene? o Relasjon elev-elev, elev-lærer o Hvordan formulerer seg når de ber om hjelp? ▪ Språk, ordforråd, tegn til systematikk, logisk resonering, sette ord på problemet? Proksimale utv.sone - Voksnes veiledning - introduksjonen/veiledningen de får i starten av timene - introduksjon/veiledning underveis i aktiviteten - Samarbeid mellom elever: hvem samarbeider? Hvordan er dynamikken? Sterke/svake elever, hvem snakker, hva blir sagt. Elevaktivitet/problemløsning - Prøving og feiling, fikling - Logisk resonering/systematisk tenking (sammen med andre og så alene) - Bryte ned til del-problemer	