UiT Norges arktiske universitet

Fakultet for humaniora, samfunnsvitenskap og lærerutdanning

Eleveres arbeid med Scratch

En kvalitativ studie om elevers bruk av løsningsstrategier og typer argumentasjon i programmering.

Kristian Berg

Masteroppgave i LRU-3903 Matematikkdidaktikk Mai 2021

Forord

Denne masteroppgaven markerer slutten på mine fem år på lærerutdanningen på UiT. Arbeidet med å skrive masteroppgaven har vært krevende og læringsrikt. Det å få sette seg inn i et tema som programmering har vært spennende, og gitt meg mange ideer som jeg gleder meg til å prøve ut når jeg starter i jobb som lærer. Jeg vil rette en stor takk til klassen som sa ja til å være mine informanter.

Jeg vil også rette en stor takk til min veileder Kjersti Wæge som har gitt meg god støtte og veiledning gjennom arbeidet med denne oppgaven. Veiledningen har vært til stor hjelp og har inneholdt gode råd og konstruktiv kritikk som jeg har satt stor pris på.

Takk til kaffeklubben for gode faglige og ufaglige samtaler. Pausene vi har vært til hjelp med oppklaringer av teori og ikke minst til å slappe av i den stressende tiden en masteroppgave kan by på.

Tromsø, mai 2021

Kristian Berg

Sammendrag

I denne masteroppgaven har jeg undersøkt hvilke løsningsstrategier og type argumenter et utvalg elever bruker når de løser programmeringsoppgaver. Programmering kom inn på læreplanen til grunnskolen høsten 2020 (Utdanningsdirektoratet, 2020) og er nytt for mange lærere og lærerstudenter. Tanken bak programmering i skolen er å forberede elevene på å kunne delta i et raskt utviklende samfunn. Formålet med forskningsprosjektet er å belyse hvordan elever arbeider med programmeringsoppgaver med å se på løsningsstrategier og nivåer av argumentasjon. Forskningsspørsmålet for studien er dermed: *Hvilke løsningsstrategier og type argumentasjon bruker elever i programmeringsoppgaver?*

Dette er en casestudie der jeg benytter meg av observasjon og semistrukturert intervju for å samle inn data. Undervisningsopplegget som går over to undervisningsøkter, er planlagt og gjennomført av meg. For å kunne svare på forskningsspørsmålet har jeg benyttet teori om problemløsning, algoritmisk tenkning, algoritmisk metode og nivåer av argumentasjon. Dette skal sammen med dataanalysen bidra til å svare på mitt forskningsspørsmål.

Studien min viser at elevene benytter seg av to forskjellige løsningsstrategier og argumentasjonene er like. Den strategien som blir mest brukt er testing og feilsøking, der elevene tester hele programmet gjør noen endringer før programmet kjøres på nytt. Inkrementell og iterativ er den andre strategien som blir brukt, her jobber elevene mer trinnvis med å bygge opp programmet og tester underveis. Elevenes argumentasjon baseres på en generalisering basert på tidligere testing og kombineres med noe matematiske betraktninger.

Innholdsfortegnelse

Figurliste

1 Innledning

1.1 Bakgrunn for valg av tema

Tema for min masteroppgave er programmering i matematikk. I løpet av mine fem år som lærerstudent med matematikk som masterfag har vi blitt undervist om hvordan man skal undervise for å bygge opp elevenes forståelse rundt matematikk. Noe som er ganske annerledes fra det jeg husker av min egen skolegang på grunnskolen. Der var matematikk et fag der pugging av ulike algoritmer var nøkkelen til å få gode resultater på matteprøven. Matematikk var et av mine favorittfag på grunnskolen, men dette var ikke fordi det var så interessant det var mer fordi det var enklere å mestre for meg en andre fag. Først da jeg hadde matematikk på videregåendeskole ble det mer interessant, da vi der fikk oppgaver det ikke var åpenbart hvordan det skulle løses. Det å løse problemer er noe som jeg syntes er mer motiverende og mer givende en pugging av formler. Prosessen der man utforsker og lager hypoteser gir mer mening for meg en det å følge en formel der det man må passe på er riktig fortegn og følge oppskriften fra læreboka eller tavla.

Den nye læreplanen (LK20) viser at tiden der matematikkfaget er et puggefag er over. I LK20 inneholder nå matematikkfaget kjerneelementene; utforsking og problemløsning, modellering og anvendelser, resonnering og argumentasjon, representasjon og kommunikasjon, abstraksjon og generalisering og matematiske kunnskapsområder (Utdanningsdirektoratet, 2020). Den tidligere læreplanen LK06 hadde også noe av det samme målet som LK20, noe av det som derimot er nytt i LK20 er innføringen av programmering. Det at programmering har kommet inn i læreplanen er hovedgrunnen til valget av tema i denne masteroppgaven. I løpet av mine fem år på universitetet i Tromsø har det vært undervist lite i programmering, og siden dette er et tema jeg skal undervise i når jeg skal ut i jobb som lærer, ønsket jeg å bruke mitt siste semester på universitetet til å forberede meg til bruk av programmering i matematikkundervisning. En fordel jeg ser med å bruke programmering til å utvikle elevenes forståelse er at tiden på å regne ut løsningsforslagene minsker, noe som jeg ser på en mulighet til å rette mer fokus på forståelsen av matematikken.

Samfunnet vårt er i en hyppig utvikling der teknologiske ferdigheter blir viktigere og viktigere. Formålet med opplæringen er ifølge opplæringslovens formålsparagraf at skolen skal åpne dører mot verden og framtiden (Opplæringslova, 1998). Programmering er en ferdighet som allerede er viktig, og som vil bli stadig viktigere i det tjueførste århundre. Det er et økt politisk fokus på programmering og hvordan skoler og utdanningssektoren forvalter

kravene til bruk og utvikling av teknologi (Kaufmann & Stenseth, 2020). Flere land deler det politiske synet på at elevene må få en utdannelse som gjør det mulig for dem å lære og forstå hovedprinsippene til programmering (Kaufmann & Stenseth, 2020). Det er imidlertid ikke meningen at alle elever skal bli eksperter på programmering, men skolen skal legge til rette for at elevene har muligheten. Gjennom programmering kan elevene også øke sine evner til problemløsning, som er en viktig ferdighet i dagens raskt utviklende samfunn. Det kan være vanskelig for skolen å følge med på den raske utviklingen til samfunnet, dette gjør det viktig for elevene at de kan løse fremtidens problemer ved å være gode problemløsere samtidig som det også kan hjelpe dem med den videre skolegangen.

1.2 Formål og forskningsspørsmål

Formålet med denne oppgaven er å belyse hvordan et utvalg elever jobber med programmeringsoppgaver i programmeringsspråket Scratch. Dette for å bli kjent med ulike metoder og argumenter elevene bruker når de arbeider med programmeringsoppgaver i matematikk. Ved å avdekke hvordan elevene løser oppgaver, kan jeg som lærer hjelpe elvene til å videre utvikle seg, og med det bygge mer forståelse om matematikkfaget. Elevene kan bli gode problemløsere i matematikk, dette er ferdigheter som også kan hjelpe elevene i problemløsningsoppgaver der det ikke er noe programmering.

For å kunne si noe om hvordan elever jobber med programmeringsoppgaver ser jeg på løsningsstrategier elevene bruker, og hvordan de argumenterer for ulike løsninger. Ved å se på løsningsstrategier ser jeg hvilke metoder de benytter seg av for å komme fram til en løsning. Hvordan elevene argumenterer vil hjelpe meg å fange opp hvordan elevene tenker. Jeg har formulert følgende forskningsspørsmål:

Hvilke løsningsstrategier og type argumentasjon bruker elever i programmeringsoppgaver?

For å undersøke dette forskningsspørsmålet, har jeg benyttet meg av semistrukturert intervju og observasjon som metode for å samle inn empiri. Utvalget i studien er ni elever som jobbet sammen fordelt på 4 par/grupper. Elevene gikk i samme klasse på 6.trinn. Klassen fulgte et undervisningsopplegg gjennom to økter som jeg planla og gjennomførte. Jeg plasserer min

studie innenfor det konstruktivistiske paradigmet (Mertens, 2015). Virkeligheten er i stadig utvikling er mitt syn, og at den er sosialt konstruert. For at sosiale fenomener skal virkelig kunne forstås bør man få tak i hvordan mennesker tolker den sosiale virkeligheten (Postholm & Jacobsen, 2018). Gjennom åpne tilnærminger som observasjon og åpent intervju får man frem hvordan mennesker selv konstruerer virkeligheten (Postholm & Jacobsen, 2018).

Jeg har i denne oppgaven brukt teori som omhandler problemløsning, algoritmisk tenkning, algoritmiske metoder og nivåer av argumentasjon for å svare på forskningsspørsmålet. Polya (1957, 1981) og Schoenfeld (1989) sin teori om problemløsning, og algoritmiske metoder fra Brennan & Resnick (2012) blir brukt for å kategorisere hvilke løsningsstrategier elevene bruker. Bocconi med flere (2016) sine begreper om algoritmisk tenkning, og Lavy (2006) og Kaufmann & Stenseth (2020) blir brukt for å beskrive hvordan elevene tenker og argumenterer. Kaufmann & Stenseth (2020) ser på hvordan elever argumentere når de skal endre et program for at det skal utføre den riktige handlingen. I denne studien vises det en utvikling av elevenes kognitive evner når elevene sammen diskutere hva som skal gjøres. Elevene går fra å tilfeldig teste ulike versjoner av programmet til å analysere og bruker matematikken til å endre programmet slik at det utfører riktig handling.

Det finnes noen tidligere studier på programmering i skolen. Scratchmaths var en toårig intervensjon i England designet som svar på endringene i den primære læreplanen for å implementere digitale ferdigheter. Målet til Scratchmaths var å forsterke læring i matematikk som følge av denne endringen (Clark-Wilson, Noss, Hoyles, Saunders, & Benton, 2019). Clark-Wilson med flere (2019) kommer ikke med noen konkrete svar på studien siden det var stor forskjell på hvor godt skolene fulgte planene til ScratchMaths, noen skoler prioriterte nasjonale tester og mindre tid ble satt til ScratchMaths. I de skolene som fulgte ScratchMaths programmet godt, ble det vist at elevene klarte å forstå Scratch gjennom å diskutere og utforske sammen. Det at en skole som hadde ekstra utfordringer klarte å oppnå gjennomsnittlig poengscore på nasjonale tester ved å følge ScratchMaths trekker Clark-Wilson med flere (2019) som et positivt funn med intervensjonen. Lavy (2006) studerer hvordan et elevpar jobber med oppgaver i programmeringsspråket Logo. Ut ifra hvordan elevene argumenterer lager Lavy (2006) fire nivåer for argumentasjon: *grunnleggende argument, sammensatt argument, utdypet argument* og *generalisert argument presenter som spesifikt*. Kaufmann & Stenseth (2020) er en nyere studie som også ser på hvordan elever

argumenterer når de løser programmeringsoppgaver. I denne studien brukes programmeringsspråket Processing. Kaufmann & Stenseth (2020) kommer med tre nivåer for argumentasjon: *Intuitiv og kodefokusert argument, argumenter basert på matematiske betraktninger* og *argumenter basert på generalisering*. En annen studie der det brukes Scratch er gjort av Niels Bonderup Dohn. Dohn (2020) ser i sin studie på hvordan scratch påvirker interessen til to klasser på 6.trinn i koding og matematikk. Funnene i denne studien viser til at programmeringsoppgavene hadde en signifikant negativ påvirkning på elevenes interesse. Dohn (2020) peker på en overstrukturert undervisningsplan og liten grad av uavhengighet som hovedårsaken til den negative påvirkningen, og en undervisningsplan med mindre struktur, høyere grad av uavhengighet og som tillater fantasi bør brukes for å ikke får en negativ påvirkning av interessen. Dohn sine funn ble brukt til å utforme mitt eget undervisningsopplegg der jeg ønsket at elevene skulle bli engasjert av å arbeide med programmering. Annen forskning som jeg vil belyse i teorikapittelet er problemløsning av Polya (1957, 1981) og Schoenfeld (1989). Algoritmisk tenkning knyttes opp mot programmering og nevnes også i kjerneelementene (Utdanningsdirektoratet, 2020). Denne problemløsningsmetoden blir videre utdypet i teorikapittelet sammen med algoritmiske metoder og argumentasjonsnivåer.

1.3 Oppgavens oppbygning

Kapittel 2 omhandler teori om problemløsning, programmering i skolen, Scratch, algoritmisk tenkning, algoritmiske metoder og nivåer av argumentasjon. Problemløsning er en stor del hvordan man ønsker at elevene skal jobbe med programmering og problemløsningsprosessen sammen med et rammeverk av algoritmiske metoder viser hvilke løsningsstrategier elevene bruker i programmeringsoppgaver. Begrepene under algoritmisk tenkning og de ulike nivåene av argumentasjon sammen med rammeverkene beskriver den mentale prosessen til elevene når de løser oppgavene. I kapittel 3 presenterer jeg valg av metode og utvalg begrunnet mot forskningsspørsmålet, undervisningsopplegget og hensikten med den. Jeg diskuterer validitet, relabilitet og overførbarhet, og hvordan jeg fikk samtykke fra elever og foresatte sammen med forskningsetiske prinsipper. Kapittel 4 består av funn fra studien og belyses med teori fra teorikapittelet. I kapittel 5 sammenligner jeg parene fra studien, ser funnene opp mot tidligere studier, diskuterer undervisningsopplegget mitt og drøfter begrensninger for studien og veien videre.

2 Teori

Dette kapittelet starter med å vise til forskning om problemløsning og viser videre til forskning om programmering i skolen. Teoriene jeg belyser her vil jeg bruke til å analysere mine data for å svare på forskningsspørsmålet.

2.1 Problemløsning

Utforsking og problemløsning er et av kjerneelementene for matematikken i kunnskapsløftet 2020 (Utdanningsdirektoratet, 2020). Her beskrives problemløsning som en handling der elevene utvikler en metode for å løse et problem der de enda ikke har kjennskap til. Ifølge Lesh & Zawojewski (2007) blir en målrettet aktivitet eller en oppgave et problem når problemløseren får behov for å utvikle en mer produktiv måte å tenke på. Schoenfeld (1989) definerer et matematisk problem i to deler der del en er at det blir et problem for en elev når eleven er interessert og engasjert, og ønsker å få en oppklaring, og del to når eleven ikke har en klar oppfatning om hvordan eleven kan oppnå denne oppklaringen. Begge definisjonene fra Schoenfeld (1989) og Lesh & Zawojewski (2007) går ut på det samme med at eleven må utvikle seg for å løse problemet. Dette vil si at noe som er et problem for en elev trenger ikke å være et problem for en annen elev, eller at noe som var et problem for en elev ikke trenger å være et problem for eleven på et senere tidspunkt Schoenfeld (1989) presiserer også at elevene må bry seg og bli engasjert over oppgaven for at det skal bli et problem. I undervisning der elevene skal løse problemløsningsoppgaver bør elevene jobbe sammen og det er viktig at elevene får tid til «leke» med problemet (Torkildsen, 2017). Elevene må prøve ut ideer som kan ende i blindvei, justere retning utfra erfaring, være villig til å ta risiko og diskutere erfaring med andre.

Polya (1957) ser på problemløsning som en praktisk ferdighet, i samme grad som svømming er en ferdighet, han hevder at slike ferdigheter skaffer man seg gjennom imitasjon og praksis. Om man ønsker seg å lære seg å løse problemer må personen observere og imitere det andre problemløsere gjør, da vil man etter hvert kunne klare å løse problemet på egenhånd. Polya (1981) mener at noe blir et problem når man må søke bevist etter en handling som passer for å oppnå et bestemt og tydelig mål, men som ikke er umiddelbart oppnåelig. Å finne denne handlingen er problemløsning. Et ønske om å oppnå et mål kan i noen tilfeller føre til et problem, men det trenger heller ikke å gjøre det. Det blir ikke et problem for problemløseren dersom han med engang ser for seg en handling som kan føre til målet (Polya, 1981).

Å hjelpe elvene sine er en av de viktigste arbeidsoppgavene til en lærer, denne oppgaven kan være vanskelig og krever tid, øving, hengivenhet og prinsipper (Pólya, 1957). Eleven kan derimot stå fast og ikke få noen videre progresjon om eleven blir alene med et problem uten hjelp eller ikke god nok hjelp (Pólya, 1957). Læreren kan heller ikke hjelpe til for mye, men gi eleven nok hjelp og arbeid slik at eleven får en mer fornuftig del av arbeidet (Pólya, 1957). Om eleven står fast og ikke får gjort noe, bør læreren gi eleven litt tro på at eleven får til noe selv ved at læreren hjelper eleven veldig diskre (Pólya, 1957). Pólya (1957) mener at den beste måten å hjelpe en elev på er å hjelpe naturlig ved at læreren ser gjennom elevens øyne. Med dette mener Pólya (1957) at læreren skal prøve å forstå hvordan eleven tenker og spørre eleven om noe som eleven selv kunne kommet på for videre progresjon.

For å finne en løsning på et problem kan man bli nødt til å se på problemet fra flere vinkler gjentagende ganger (Pólya, 1957). Pólya (1957) skiller mellom fire faser når man arbeider med problemløsning. *Forstå problemet* er den første fasen. Eleven må forstå hva som er problemet som skal løses og ønske å komme fram til en løsning (Pólya, 1957). Den andre fasen er å *lage en plan*, når man vet hvilke kalkulasjoner, beregninger eller konstruksjoner som det er behov for å utføre for å løse problemet har man en plan (Pólya, 1957). Det å lage en god plan er ifølge Pólya (1957) en av hovedpunktene i en problemløsningsstrategi og kan komme gradvis og/eller etter flere mislykkete forsøk, det er denne fasen som er mest krevende. Fase tre å *utføre planen* er betydeligere enklere enn å lage en plan, og hovedfokuset her er tålmodighet til å utføre arbeidet som trengs (Pólya, 1957). Den fjerde og siste fasen er å *se tilbake* (Pólya, 1957), her må eleven se på hva som er gjort og hvorfor dette fungerte. Ved å kontrollere og dobbeltsjekke alle stegene i planen kan eleven utvikle en større kompetanse til å løse problemer.

Schoenfeld (1989) sammenligner hvordan en matematiker jobber med å løse et geometrisk problem med hvordan mange elever arbeider med problemer. Problemløsningsfasen deler han inn i seks faser: 1) lese, 2) analysere, 3) utforske, 4) planlegge, 5) implementere og 6) sjekke. Matematikeren bruker her over halvparten av tiden på å forstå problemet. Han brukte store deler av tiden til å analysere og starter ikke å implementere før han var sikker på at han var på riktig vei. Man ser den rake motsetningen hos mange elever. De leser problemet raskt før deretter tar en avgjørelse på hvordan oppgaven skal løses og holdt fast på den retningen. Selv om de forstod at retningen var feil endret de ikke retning, men brukte hele tiden på den bestemte retningen (Schoenfeld, 1989). Når elever jobber med ukjente problem som er fri for kontekst er arbeidsmåten å lese og fort bestemme seg for en retning vanlig (Schoenfeld,

1989). Selv om elevene har kunnskapen de trenger for å løse oppgaven klarer de allikevel ikke å løse den. Årsaken her er ifølge Schoenfeld (1989) at elevene ikke kan bruke eller ikke er klar over bruken av de metakognitive ferdighetene som man ser hos matematikeren. Dette er ferdigheter som kan læres ved å sette fokus på disse ferdighetene i undervisning (Schoenfeld, 1989). Schoenfeld (1989) viser til egen undervisning der han ofte deler klassen i grupper på tre til fire elever og gir dem problematiske oppgaver, der han går rundt og stiller dem veiledende spørsmål.

Jeg vurderer at Polya og Schoenfeld beskriver problemløsningsprosessen ganske likt. Polya deler inn i fire faser imens Schoenfeld deler den inn i seks faser, men Schoenfeld sine tre første faser (lese, analysere og utforske) kan man si omfatter det som er Polya sin første fase, forstå problemet. De tre siste fasene til Schoenfeld vurderer jeg også lik de tre siste fasene til Polya.

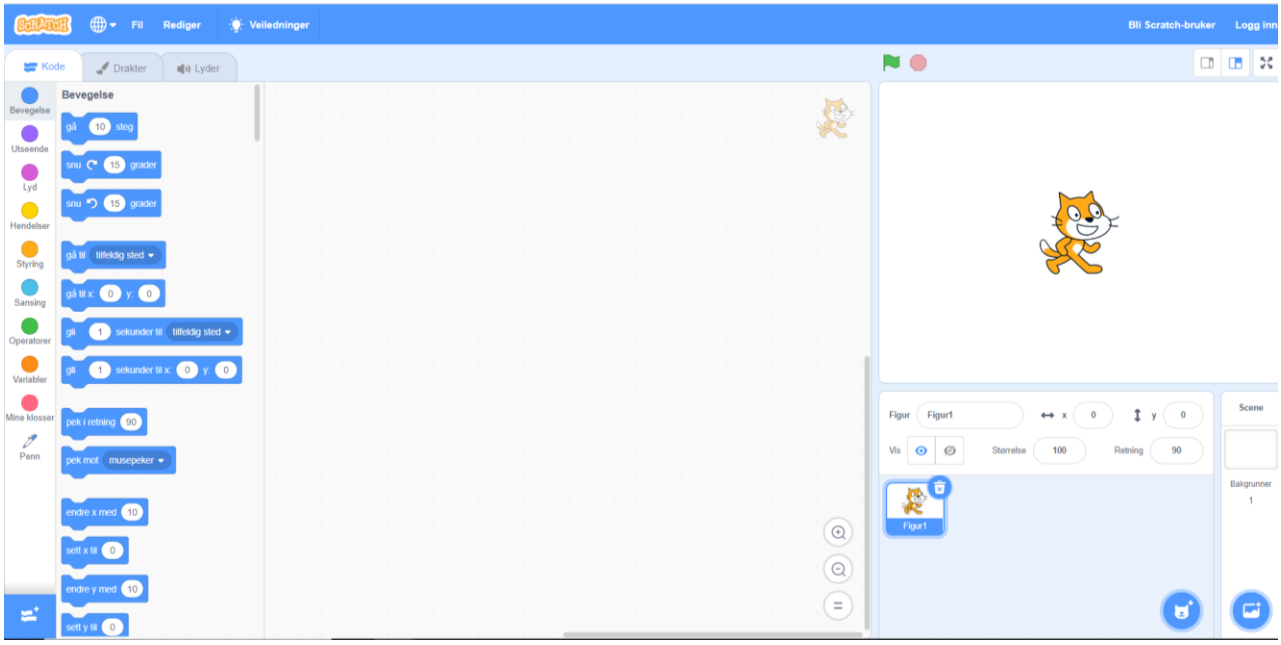
2.2 Programmering i skolen

Programmering kan ses på som en grunnleggende ferdighet for delta aktivt i den digitale verden (Stenseth, Kaufmann, & Forsstøm, 2019). I LK20 er programmering del av digitale ferdigheter i matematikk, og i kompetansemålene i matematikk blir programmering fra og med 5.trinn til og med 10.trinn og algoritmisk tenkning fra 1.trinn (Utdanningsdirektoratet, 2020). Stenseth et al. (2019) definerer programmering som prosessen bundet til utviklingen og implementeringen av instruksjoner for dataprogrammer slik at datamaskinen kan kjøre spesifikke oppgaver, løse problemer og støtte menneskelige interaksjoner. Det kreves derfor at de som programmerer har kunnskap om programmeringsspråket, fagkunnskap relatert til utvikling av spesialiserte algoritmer og evne til å analysere, å forstå og løse problemer ved å verifisere algoritmiske krav og vurdere korrekthet og kodingen av algoritmen i et bestemt programmeringsspråk (Stenseth, Kaufmann, & Forsstøm, 2019). Disse prosessene knyttes ofte til matematisk tenkning og algoritmisk tenkning og har blitt en viktig ferdighet for framtidens kompetanse innen problemløsning, logisk tenkning og kreativitet og det digitale samfunn. Algoritmisk tenkning og feilretting er to elementer innenfor programmering som Stenseth, Kaufmann og Forsstøm (2019) peker på kan bidra med tenkning og problemløsningsmetodikk som styrker matematikken.

2.2.1 Scratch

Scratch er det aller mest utbredte blokkprogrammeringsverktøyet og er utviklet av Massachusetts Institute of Technology, MIT Det ble laget for å være et verktøy for å

introdusere programmering til nybegynnere (Haraldsrud, Sveinsson, & Løvold, 2020). Tanken bak Scratch er at språket skal være en start inn i programmeringens verden for å lære tankegangen bak programmering, uten å lære seg syntaksen til tekstbasert språk som er tyngre. Scratch kan kjøres på nettleseren, det er dermed ikke behov for å laste ned programvare på elevenes datamaskiner. Dette gjør at det er enkelt å komme i gang med Scratch. Elevene kan lage en egen brukerprofil slik at alt av arbeid de ønsker å beholde blir lagret. Når de starter Scratch, ser man i menyen til venstre, tilgjengelige blokker under de forskjellige kategoriene; bevegelse, utseende, lyd, hendelser, styring, sansing, operatorer, variabler og mine klosser. Penn er en annen nyttig kategori man kan hente fra Scratch sitt bibliotek av ekstrafunksjoner. Blokkene man ønsker å bruke drar man over i vinduet i midten. Helt til høyre har man et vindu med en katt, denne katten er noe vi kaller en sprite. Ved bruk av katten kan vi animere og få den til å utføre handlinger ved å skrive et program med blokkene. Under kategorien hendelser finner vi blokker som kan brukes til å starte programmet, den mest grunnleggende blokken i denne kategorien er *når grønt flagg klikkes*. Blokken *gjenta n ganger* ble ofte brukt av elevene i undervisningsopplegget som ble utprøvd, og er en løkke funksjon der blokkene inne i denne blokken blir gjentatt n ganger. Siden Scratch er et visuelt språk, med vinduet med katten som utfører det programmet som elevene har skrevet, innbyr det til grafisk programmering. Dette kan vi bruke til å utforske matematiske konsepter, og spesielt geometri er her et felt som særlig er lett å utforske med bruk av Scratch (Haraldsrud, Sveinsson, & Løvold, 2020). I Scratch er bevegelse og bruken av kategorien penn sammen med kategoriene hendelse og styring viktig for at katten skal tegne geometriske figurer. I kategorien bevegelse brukes blokkene gå n skritt og snu n grader for å bestemme hvor lang linjen skal være og hvor mange grader katten skal snu seg. Kategorien *penn* brukes for å tegne og slette linjene som katten tegner, den kan også brukes til å endre linjetykkelse og linjefargen.



Figur 1 Skjermbilde av Scratch. hentet fra https://scratch.mit.edu/projects/editor/?tutorial=getStarted

2.2.2 Algoritmisk tenkning

Algoritmisk tenkning blir nevnt i kunnskapsløftet 2020 som viktig i prosessen med å utvikle strategier og fremgangsmetoder for å løse et problem, og er en del av kjerneelementet utforskning og problemløsning (Utdanningsdirektoratet, 2020). Det eksisterer flere likelydende begreper som omhandler det å tenke med og om algoritmer, som Gjøvik & Torkildsen (2019) mener betyr mer eller mindre det samme. På engelsk er det snakk om både algorithmic thinking og computational thinking innenfor programmering. Direkte oversatt er algorithmic thinking det som blir mest likt algoritmisk tenkning. Utdanningsdirektoratet har satt algoritmisk tenkning som den norske oversettelsen for computational thinking (Utdanningsdirektoratet, 2019) og det er denne oversettelsen jeg også vil vise til når jeg bruker algoritmisk tenkning.

Algoritmisk tenkning i praktisk programmering er å bryte en løsningsstrategi eller plan til mindre sekvenser med mulige handlinger og definisjoner av passende variabler (Kaufmann & Stenseth, 2020). Ved bruk av algoritmisk tenkning vurderer man hvilke steg som trengs for å løse et problem, og ved det bruke sin teknologiske kompetanse for å få en datamaskin til å løse problemet (Utdanningsdirektoratet, 2019), Wing (2017) legger til at mennesker også skal kunne løse problemet og at algoritmisk tenkning ikke kun handler om problemløsning, men også problemformulering. Algoritmisk tenkning kan sies å bestå av

de seks kjerne begrepene; abstraksjon, algoritmebehandling, automatisering, dekomponering, feilretting og generalisering (Bocconi, Chioccariello, Dettori, Ferrari, & Engelhardt, 2016). Abstraksjon vil si å trekke ut essensen fra flere eksempler og se bort ifra irrelevante opplysninger. Den viktigste og vanskeligste begrepet i algoritmisk tenkning er abstraksjon (Wing, 2017). Abstraksjon er nøkkelen til algoritmisk tenkning og for å kunne designe effektive algoritmer må man kunne bruke abstrakte data, Wing (2017) definerer dette som abstrakte verdier og handlingen for å behandle disse verdiene. Med bruk av abstraksjon kan man endre størrelse og håndtere kompleksitet, da kan man lage mye større saker utfra en basesak (Wing, 2017). Ved algoritmebehandling følger og forklarer man trinnvise sekvenser og regler, når man forstår en algoritme trenger man ikke å lage den fra bunn av for et nytt lignende problem (Csizmadia, et al., 2015). Automatisering er å implementere løsningen av problemet i programmeringsspråk og gjøre de menneskelige beregningene mindre ved å bruke en datamaskinens raske regnekraft (Lee, et al., 2011). Dekomponering er å gjøre et stort problem om til mindre og mer håndterlige problemer ved å se på komponentene som gir det hele (Csizmadia, et al., 2015). Hver kode i et program blir her hver for seg forstått, løst, utviklet og evaluert. Feilretting er å systematisk analysere og evaluere ved bruk av testing, sporing og logisk tenkning for å forutsi og bekrefte utfallet (Csizmadia, et al., 2015). Å generalisere er å kjenne igjen sammenhenger og mønstre, og lage metoder og regler for å enklere kunne løse andre lignende problem (Csizmadia, et al., 2015). Csizmadia, et al (2015) mener at å stille spørsmål som «Er dette problemet likt et tidligere problem?» og «Hva er forskjellen på problemene?» kan hjelpe med å se sammenhengen av mønstrene og strategier som blir brukt.

2.2.3 Algoritmiske Metoder

Brennan & Resnick (2012) peker på at det er lite enighet om strategier for å analysere og vurdere algoritmisk tenkning. Med interesse om programmering og ønske til å utvikle algoritmisk tenkning har Brennan & Resnick (2012) i løpet av noen år utviklet et rammeverk for algoritmisk tenkning. Et av de tre hoveddimensjonene i dette rammeverket er *Algoritmiske metoder* (computational practices) (Brennan & Resnick, 2012). Algoritmiske metoder beskriver konstruksjonsprosessen og setter fokus på tankegang og læring for å se hvordan

elevene lærer (Brennan & Resnick, 2012). Gjennom studie kommer de fram til fire algoritmiske metoder:

1. **Inkrementell og iterativ**. Å lage et program er ikke en ren sekvensiell prosess der man først identifiserer hva programmet skal gjøre deretter lage en plan for programmet før man implementerer kodene til programmet. Det er en prosess der man går stegvis fram og må tilpasse og endre planene for å komme nærmere en løsning. Elevene utvikler programmet med å kode litt og litt, tester kodene ut for deretter å utvikle programmet videre basert på erfaring de oppnådde under testing og eventuelle nye ideer de har kommet opp med.
2. **Testing og Feilsøking**. Det går sjeldent som planlagt i starten av en programmeringsprosess og det er en av årsakene til at det er viktig for elevene å utvikle strategier for å håndtere og forutse potensielle problemer. Elever utvikler denne metoden ut fra å prøve og endre, hva de har lært fra andre oppgaver eller ved støtte fra en annen kunnskapsrik elev eller lærer. Noen metoder elevene brukte i Brennan & Resnick sin studie var; å identifisere problemet, lese igjennom kodene, eksperimentere med kodene, skrive kodene på nytt, finne eksempler på programmer som fungerer, spørre noen om hjelp og ta en pause.
3. **Gjenbruk og blande**. Det å bygge videre på andre sine programmer er vanlig praksis hos programmerere og har blitt forsterket av internett som har økt bruken av denne metoden. Gjenbruk og blanding hjelper til i utviklingen av kritisk kode-tolkning og provoserer viktig spørsmål om forfatterskap og eierskap. I Brennan & Resnick sin studie brukte en elev dette to ganger ved at ideen om programmet kom fra en spillside hun hadde sett og senere ved å bytte sprite (figur som utfører programmet) til en med jetpack hentet fra en tilleggsfunksjon.
4. **Abstraksjon og modulisering**. Denne metoden karakteriseres av å lage noe stort med å sette sammen små samlinger, og er en viktig metode for alle som driver med programmering og problemløsning. Elevene bruker abstraksjon og modulisering på flere nivåer i studien, fra starten av arbeidet med å tolke begrepene i problemet til å oversette dette til individuelle sprites og stabler av koder.

Disse fire metodene til Brennan & Resnick (2012) gjør det enklere å kategorisere metodene elever benytter seg av når de jobber med programmering i skolen. Brennan & Resnick (2012) sine metoder kan vi koble til Bocconi med flere (2016) sine begreper for algoritmisk tenkning. I inkrementell og iterativ metode vises begrepene algoritmebehandling og

dekomponering ved at elevene bryter ned programmet til mindre deler som løses trinnvis, det blir også brukt feilretting i testingen av sekvensene. Med metoden testing og feilsøking bruker elevene automatisering og feilretting når de prøver og endrer programmet. Når elevene prøver å finne ut hva som må endres brukes dekomponering og generalisering ved å se på kodene og bruke tidligere oppgaver for å kjenne igjen sammenheng og mønster. Ved gjenbruk og blande metoden bruker elevene dekomponering og feilretting når de vurderer et annet program og avgjør om programmet kan brukes ved å evaluere de ulike kodene brukt i programmet. Algoritmebehandling kan også vises i gjenbruk og blande ved å følge trinnvis et annet program, eller generalisering ved å se sammenhenger og mønstre i et annet program for å utvikle et eget. I metoden abstraksjon og modulisering brukes abstraksjon når man setter sammen flere samlinger til et mer komplekst program. Det blir også brukt dekomponering, algoritmebehandling og generalisering når eleven bygger en forståelse over de mindre samlingene i programmet og sammenhenger, mønster og regler dannes. Feilretting kan også vises om eleven bygger forståelse av samlingene ved testing og endring. Automatisering vises i abstraksjon og modulisering ved benytte datamaskinens regnekraft for å utføre det komplekse programmet.

2.2.4 Nivåer av argumentasjon i programmering

Å argumentere vil si å føre frem bevisgrunner eller argumenter, det kan også bety å tale for en sak (Store Norske Leksikon, 2018). Lavy (2006) refererer i sin studie til diskursive utvekslingen mellom elever i ferd med å formulere en viss matematisk regelmessighet som argumentasjon, og data, krav og resonnement som elevene bruker kaller hun for argument. Når elevene diskuterer for hva som burde endres i programmet for at programmet skal bli riktig, vil jeg si at de argumenterer. I Lavy (2006) sin studie bruker elevene programmeringsspråket Logo der de så etter matematiske regelmessigheter i et geobrett program som hadde forskjellige antall punkter elevene kunne knytte sammen for å lage ulike figurer. Målet med oppgavene var å undervise elevene i tallteorikonsepter i en interaktivt datastyrt setting. Eksperimentet hadde en varighet på ti økter der elevene utviklet sine talteorier og argumentasjon. Lavy satte søkelys på to elever som jobbet sammen i par, disse to ble valgt siden de ofte tenkte høyt.

Lavy (2006) utviklet fire forskjellige nivåer av argumentasjon da hun studerte to elever som jobbet i et datamiljø. Hun fulgte disse elevene gjennom deres utvikling over flere timer der elevene videre utviklet argumentasjonen.

Det første nivået er *grunnleggende argument*. Her er nivået på argumentasjonen lokalt, som betyr at antagelsene som danner argumentet er et resultat av testing av noen uformelle løsninger (Lavy, 2006). Elven gjør noen få tester med liknende program der kodene er nesten like og ut ifra hva eleven observerer kan han argumentere for en bestemt handling vil gi et bestemt resultat. For eksempel kan en elev som jobber med Scratch se hvordan et kvadrat program med gjenta n (n for tilfeldig tall) ganger ser ut. Eleven ser at dette er et program for kvadrat, men ser ikke på kodene hva de gjør i programmet.

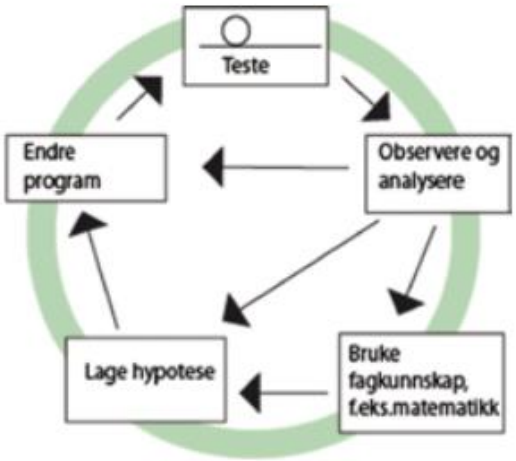
Nivå to er *sammensatt argument*. I dette argumentet er resonnementet sammensatt av konkrete eksempler og matematiske betraktninger. Det har noen likheter med det første nivået ved at det dannes en generalisering basert på noen tester, men her vil eleven også bruke matematiske betraktninger knyttet til tallenes egenskaper. For eksempel vil eleven se i Scratch at antall skritt i et program som skal tegne et kvadrat, bestemme størrelsen på kvadratets omkrets og areal.

Det tredje nivå for argument er *utdypet argument*. Her bygger eleven videre på det som ble oppdaget i sammensatt argumentasjon. Dette nivået blir utviklet når konklusjonen eleven har i sammensatt argumentasjon ikke lenger passer, eller et nytt problem dukker opp der tidligere argumentasjon ikke blir gyldig (Lavy, 2006). På dette nivået vil eleven endre argumentet til å enten ikke gjelde det nye tilfellet eller inkludere også det nye. Eksempelvis om eleven med scratch trodde at programmet for kvadrat gjaldt alle firkanter, men så skal han lage et rektangel og ser at da gjelder ikke dette programmet for alle firkanter, men bare for kvadrater. Eleven kan si at dette er et program for et kvadrat eller endre programmet til et som vil passe et rektangel også. I likhet med sammensatt argumentasjon skjer utprøvingen i utdypet argumentasjon med kontrollert testing og færre forsøk i motsetning til hos grunnleggende argumentasjon (Lavy, 2006).

Det fjerde og siste nivået kalles *generalisert argument presentert som spesifikt* (Lavy, 2006). Dette nivået er ifølge Lavy (2006) basert på sammensatt og utdypet argumentasjon, og begrunnelsene her baseres argumenter gjeldende til det spesifikke tilfellet med den generelle påstanden og matematiske betraktninger. Elevene bruker her tidligere konklusjoner og nye eksempler fra en oppgave når de skal argumentere for hvordan neste oppgave skal løses. Eksempelvis vil en elev som jobber med Scratch kunne se at for å kode et rektangel med utgangspunkt fra et program som tegner et kvadrat, se at det kan legges til en kode med n skritt og en kode med 90 grader i løkkefunksjonen for å lage et rektangel.

Elevene i Lavy (2006) sin studie var i de første to øktene fasinert over den grafiske effekten å kjøre store tall inn i kodene, elevene ble her oppmuntret til å prøve med mindre tall og jobbe mer systematisk for å hjelpe dem i å finne interessante matematiske relasjoner. Grunnleggende argument ble vist i løpet de tre første øktene og sammensatt argument ble sett fra den tredje økten. I løpet av fjerde og femte økt viste elevene utdypet argument og generalisert argument presentert som spesifikt.

Korreksjon eller feilretting, er en elementær del av arbeidet med programutvikling (Stenseth, Kaufmann, & Forsstøm, 2019). Første gang man skriver og tester et program er de nesten aldri «riktig». Det kan være feil i syntaksen eller kjøretiden, eller programmet kan gi et uønsket svar. Dette kan bidra til at programmering oppfattes som rigid og vanskelig i starten siden det ofte blir underkommunisert i programmerings undervisning. Denne prosessen kan imidlertid snus til noe positivt fordi feilretting underbygger resonnementer som innebærer hypotesetesting og hypoteseformulering. Når elevene skal prøve å rette opp i en feil, må de forstå programmet for å analysere feilen og lage hypoteser om hvordan feilen kan rettes (Kaufmann & Stenseth, 2020). Kaufmann & Stenseth (2020) nevner også at det er en risiko for at feilretting kan bli en målløs syklus der elevene gjør endringer basert på lite gjennomtenkte hypoteser. Feilretting medfører en gjentakende innfallsvinkel til problemløsning som vil si at arbeidet går i en syklus lik den som vist under i figur 1.



Figur 2 Arbeidssyklus i feilretting fra «Programering og matematikk» av Stenseth, B., Kaufmann, O. T., & Forsstøm, S. E. (2019, Februar). Tangenten s.8

Kaufmann & Stenseth (2020) bruker Lavy (2006) sitt rammeverk om argumentasjon når de ser på hvordan tre elever jobber med programmering og korrigerer kodene. Arbeidsmetodene elvene bruker er gjentagende sykluser illustrert i figur 1. Det man kan se i denne figuren er at

ikke alle som jobber med feilretting er innom alle punktene i arbeidssyklus. I Kaufmann og Stenseth (2020) er oppgaven at elevene skal endre variablene i et allerede skrevet program i programmeringsspråket Processing. De kategoriserer tre nivåer for argumentasjon der elevene jobber i gjentakende sykluser:

1. *Intuitive og kodefokuserte argument*. I denne arbeidssyklusen kjøres programmet, og deretter observeres hva som skjer for så å gjøre seg en enkel analyse. Dette kanskje før en litt tilfeldig endring gjøres og programmet kjøres på nytt. I denne kategorien går eleven fra teste til å observere og analysere til å endre program og tilbake til teste i figur 2. Her bruker den intuitive og kodefokuserte tidligere eksempler for å begrense antall tilfeldige forsøk lignende det man ser på nivået grunnleggende argument. Argumentene baserer seg på en variabel og en hypotese dannes av hva endringen i variabelen vil gjøre. Eksempelvis vil en elev se at man kan gjøre et kvadratet større om man øker antall skritt i programmet.
2. *Argumenter basert på matematiske betraktninger*. Matematikken blir her brukt til å analysere hva som skjer når programmet kjøres og til å danne en hypotese. Det er her man vil at elevene skal komme siden det er her elevene kan visualisere matematiske hypoteser mener Kaufmann & Stenseth. I forhold til den første kategorien så vil elevene her lage en hypotese som også bruker matematiske betraktninger. Måten det argumenteres på her er lik det vi ser i sammensatt og utdypet argumentasjon. Eksempelvis vil elevene programmere et rektangel basert på at rektangelet skal ha to og to sider som er like lange bruke dette når de skal legge inn riktige koder i en løkkefunksjon.
3. *Argumenter basert på generalisering.* Arbeidsprosessen og argumentene her er basert på en utvidelse i et gitt problem til andre lignende problem. Disse argumentene består av flere argumenter fra lignende problemer som sammen er tilstrekkelige til å støtte opp et nytt argument, ligner på generelle argument presentert som spesifikke.Her brukes alle punktene i figur 1. Elevene vil eksempelvis her bruke matematisk fagkunnskap til å programmere et program som kan konstruere flere forskjellige geometriske figurer.

I Lavy (2006) sin studie oppnår elevene det siste nivået av argumentasjon, men i Kaufmann & Stenseth (2020) sin studie kommer ikke elevene opp på det siste nivået. Noe av grunnen til dette kan være forskjellen i tidsrammene av studiene Lavy (2006) sin studie gikk over ti økter, imens Kaufmann & Stenseth (2020) sin studie gikk over en uke med to økter der en økt

var to undervisningstimer på 45 minutter. Elevene i Lavy (2006) sin studie oppnår det fjerde nivået i slutten av den fjerde økten og i løpet av den femte økten, så det kan være mulig at elevene i Kaufmann & Stenseth (2020) sin studie hadde klart det om de hadde fått noen økter til. Det var nesten lik alder på elevene i studiene, i Kaufmann & Stenseth (2020) gikk de i 8.trinn og i Lavy (2006) gikk de i 7.trinn. Elvene fra begge studiene deltok også i fag som var rettet mot problemløsning og/eller programmering. Oppgavene elevene fikk var forskjellig og kan ha betydning for resultatene. I Lavy (2006) sin studie brukte elevene programmeringsspråket Logo og kunne endre to variabler som tegnet forskjellige polygoner og stjerner. I Kaufmann & Stenseth (2020) sin studie fikk elevene et ferdig program, der de måtte endre variablene for å få riktig løsning. Det var dermed mere data i Kaufmann & Stenseth (2020) og derav flere distraksjoner som kunne føre elevene på villspor.

3 Metode

I dette kapittelet presenterer jeg valg av metode, utvalg og gjennomføring av observasjon og intervju. Jeg presenterer også undervisningsopplegget og gjennomføringen av dette. Videre drøfter jeg studiens relabilitet, validitet og overførbarhet, og vil vise til forskningsetiske prinsipper ved studien.

3.1 Metodisk tilnærming

Denne masteroppgaven handler om hvordan elever arbeider og løser programmeringsoppgaver i matematikk. Problemstillingen er: «Hvilke løsningsmetoder og argumenter bruker elever i programmeringsoppgaver i matematikk?». For å svare på spørsmålet gjennomfører jeg en casestudie der formålet er å få kunnskap om hvilken løsningsstrategi og argumenter elevene benytter seg av når de jobber med programmeringsoppgaver.

Utfra denne problemstillingen velger jeg en kvalitativ tilnærming. Postholm og Jacobsen (2018) forklarer at den kvalitative metoden brukes når intensjonen er å forstå og beskrive hva bestemte mennesker gjør i sitt hverdagsliv. Hovedformålet med denne metoden har alltid vært å forstå og beskrive «den andre». Den kvalitative forskningen benytter seg av et åpent og fleksibelt design (Corbin & Strauss, 2015). Gjennom sin fleksibilitet gir en kvalitativ tilnærming gode muligheter for relevante tolkninger av data fra hver enkelt undersøkelse enhet (Bjørndal, 2015). Corbin og Strauss (2015) nevner å utforske den indre opplevelsen eller hvordan meninger blir dannet og transformert som noen av grunnene til at forskere velger kvalitativ metode. Kvalitative metoder kjennetegnes av at man forsøker å få mye data om et mindre antall informanter (Christoffersen & Johannessen, 2012).

For å kunne undersøke og få en dypere forståelse for hvilke løsningsmetoder elever bruker i programmeringsoppgaver, har jeg begrenset antall deltagere og tidsramme. Utvalget mitt ble derfor en liten gruppe elever innenfor en bestemt tidsramme. Denne forskningen er da basert på en casestudie, som belyses og tolkes gjennom semi-strukturerte par-intervjuer og observasjon. I tillegg planlegger og gjennomfører jeg undervisningen som elevene skal ha i det gitte tidsrommet.

3.2 Casestudie

Casestudie er en samlebetegnelse for flere forskningsdesign som har noen enkelte variasjoner (Postholm & Jacobsen, 2018). Christoffersen og Johannessen (2012) skriver at casestudier brukes mye innenfor utdanningsforskning. Denne type studie kan knyttes til kvalitative tilnærminger, som intervju og observasjon. De peker videre på at i en casestudie ønsker forskeren å samle mye data fra noen enheter over en lang eller kort tidsperiode gjennom en omfattende og detaljert datainnsamling. Enheter kan eksempelvis være elever, elevgrupper, klasser eller skoler, og disse enhetene steds- og tidsavhengige. En case har to klare kjennetegn; oppmerksomheten er avgrenset mot den spesielle casen og en mest mulig detaljert beskrivelse. Casen undersøkes detaljert og grundig for å få mest mulig data, slik som i min studie.

En enkelcasestudie har som målsetting å presentere grundige forståelser av en enkelcase (Creswell & Poth, 2018). Forståelse av det som finner sted innenfor en spesiell kontekst vil ofte være utgangspunktet. Postholm og Jacobsen (2018) forklarer at man kan for eksempel som forsker gå inn i en enkelt klasse for å forstå akkurat disse elevene og hvordan disse elevene handler, tenker og skaper. Da er da akkurat disse elevene i akkurat den klassen som står i sentrum. Videre kan slike studier også ha en forklarende hensikt der fokuset er å få rede på ulike prosesser som skaper et spesielt resultat eller tilstand. Om forskeren ønsker mer generelt å belyse en sak, kan hun eller han velge en case for å belyse nettopp denne saken. Stake (1995) kaller en slik case for en instrumentell casestudie. En enkelcasestudie vil i utgangspunktet produsere det som kalles «lokal kunnskap» (Postholm & Jacobsen, 2018).

3.3 Observasjon

Begrepet observasjon stammer fra latin og betyr å undersøke eller å iaktta (Bjørndal, 2015). Å observere er noe alle mennesker gjør i sin hverdag, men begrepet er i faglige kontekster vanlig å oppfatte som noe snevrere. Observasjon i pedagogisk sammenheng blir for eksempel forstått som «oppmerksom iakttakelse», som vil si at man forsøker å observere noe som har pedagogisk betydning på en konsentrert måte. Christoffersen og Johannessen (2012) mener at observasjon egner seg godt som metode når forskeren ønsker direkte tilgang til det han eller hun undersøker, og den er innenfor et avgrenset og overkommelig geografisk område. I flere sammenhenger kan den eneste måten å innhente gyldig kunnskap på være å være til stede i en setting.

I min studie velger jeg observasjon som metode fordi det gir meg mulighet til å være til stede i settingen jeg forsker på. Ved å være til stede i settingen får jeg gode forutsetninger til å beskrive og samle inn hvordan elevene jobber med oppgavene. Jeg som forsker vil møte forskningsfeltet med min teoretiske bakgrunn og mine antakelser. Teorien og antakelsene vil ifølge Christoffersen og Johannessen (2012) filtrere på mange måter min observasjon. Observasjonen vil føre til at jeg utvider min teoretiske kunnskap for stadig å få en bedre oversikt over praksisfeltet. En utfordring med observasjon er at jeg må være åpen for det som ligger i dataene for ikke å øke risikoen til at jeg overser temaer jeg ikke har tenkt på før datainnsamlingen (Christoffersen & Johannessen, 2012). Siden jeg før studien hadde litt generell kunnskap om programmering og det er begrenset hvor mye kunnskap man får lest seg opp på, var dette noe jeg prøvde å være bevist på under min observasjon.

Gold (1958) definerte fire observatørroller, disse fire rollene kan plasseres langs de to ulike dimensjonene «Grad av avstand» og «Grad av deltakelse». De fire rollene kaller Gold «deltaker som observatør», «fullstendig deltaker», «fullstendig observatør» og «observatør som deltaker». Postholm og Jacobsen (2018) beskriver at i rollen som fullstendig observatør, vil forskeren på ingen måte samhandle med det som observeres, og forskeren har ingen tilknytting til situasjonen som blir observert. I rollen observatør som deltaker deltar ikke forskeren i aktiviteten og er mest observatør, her kan forskeren, om han er i klasserommet, ikke svare på spørsmål som gjelder undervisning, men kan svare vennlig på spørsmål på hvem han er og hva han gjør. I denne rollen må forskeren ikke være en del av det som observeres. Når forskeren inntar deltaker som observatør rollen har han en tydeligere observatørrolle enn når han er fullstendig deltaker, her har forskeren liten grad av avstand og liten grad av deltakelse. Den siste rollen er fullstendig deltaker. Her er forskeren en del av det som observeres og kan for eksempel være en lærer som observerer egen undervisning. Det kan være en utfordring å få observert mens man samtidig underviser, når man da må skrive observasjonsnotater rett etter undervisningen er ferdig (Postholm & Jacobsen, 2018). Min observatør rolle ble fullstendig deltaker. Dette ettersom det var ønskelig for læreren at jeg styrte undervisningen da læreren hadde lite erfaring med programmering og scratch. For å sikre at dataene mine fra observasjon ble mest mulig korrekt noterte jeg underveis samtidig som jeg skrev et mer utfyllende observasjonsnotat etter at undervisningen var over. Jeg prioriterte også å fokusere mest av min oppmerksomhet på fire av elevparrene for å øke kvaliteten på mine observasjoner.

3.4 Intervju

Den andre kvalitative metoden jeg valgte å bruke er intervju. Dette fordi jeg ønsket detaljert og utfyllende data fra elevene på hva de gjorde og på hvilke argumenter de førte når de jobbet med oppgavene. Den mest brukte metoden for å samle inn kvalitative data på, er å bruke kvalitative intervjuer (Christoffersen & Johannessen, 2012). Metoden er fleksibel og kan brukes bortimot overalt, metoden gjør det også mulig å få detaljerte og fyldige beskrivelser. Samtale er viktig for mennesker, det er gjennom samtaler man kan forstå hverandre, kommentere hverandres utsagn eller handlinger, svare på hverandres spørsmål. Samtaler brukes også for å beskrive intensjoner de har, hva de føler, mener og tenker. Skal jeg tolke betydningen av hvordan elevene jobber med programmeringsoppgaver, er det viktig at det kvalitative intervjuet får frem beskrivelser av prosessene elevene hadde når de jobbet med oppgavene. Dataene i kvalitative intervju er svar på forskerens spørsmål, som regel blir dette registrert på lydbånd, digital diktafon, minidisk eller iPod og deretter skrevet ut. Jeg valgte å bruke en digital diktafon siden dette gjorde det enklere å sikre at ingen andre meg kunne skaffe seg tilgang til lydfilene, med tanke på forskningsetiske prinsipper. Christoffersen & Johannessen (2012) nevner at når forskeren har behov for å gi informanten mer frihet til å utrykke seg enn det som er mulig i et strukturert spørreskjema, er det henskiktsmessig å velge kvalitativ intervju som metode. Når infomanten kan være med å bestemme hva som tas opp i intervjuet kommer menneskets erfaringer og oppfatninger best fram. Jeg velger derfor et semi-strukturert intervju for at elevene skal kunne få uttrykke seg best mulig om metodene og argumentasjonen de hadde i undervisningen. Ved å bruke et semi strukturert intervju har jeg en intervjuguide som hjelper meg å holde den røde tråden, samtidig som jeg kan være fleksibel med muligheten til å følge på med oppfølgingsspørsmål på tema eller svar som elevene kommer med og variere rekkefølgen på spørsmålene.

En intervjuguide blir ofte benyttet for å sikre at man får med seg alle aspektene man ønsker å få svar på (Christoffersen & Johannessen, 2012). Jeg formet en intervjuguide som ville hjelpe meg i å svare på forskningsspørsmålet samtidig som jeg la til rette for oppfølgningsspørsmål og andre spørsmål basert på observasjonen. I intervjuguiden er de første spørsmålene introduksjonsspørsmål ment som en oppsummering av timen, slik at elevene får enkle spørsmål og en myk start. Introduksjonsspørsmål brukes ofte og hjelper elevene med å rette oppmerksomheten mot tema (Christoffersen & Johannessen, 2012). Videre i intervjuguiden kommer nøkkelspørsmålene, hensikten med disse spørsmpålene er å få den informasjonen jeg ønsker utfra forskningsspørsmålet og formålet (Christoffersen & Johannessen, 2012). Mine

nøkkelspørsmål har som hensikt å avdekke hvilke løsningsstrategier elevene brukte og hvordan de arguementerte. I avsluttningen ble elevene spurt om de ville gjort noe annerledes om de skulle løst oppgavene på nytt. Dette for å gi elevene en mulighet til å komme med tanker som de ikke fikk vist tidligere i undervisningen og intervjuet.

3.5 Utvalg

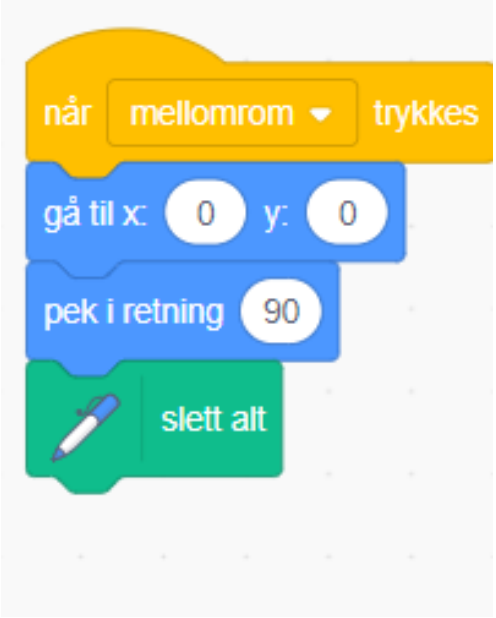
For å velge ut informanter til forskningsspørsmålet mitt, ønsket jeg en klasse på mellomtrinnet eller ungdomsskolen, der alle elevene har tilgang til minst en pc på deling av to elever. Jeg la ingen kriterier for hvor mye tidligere erfaring elevene skulle ha med programmering og scratch, men noen forhåndskunnskaper var å foretrekke. Elevene i klassen jeg forsket på hadde tilgang til en pc hver og hadde litt erfaring med scratch fra før. Utvalgets størrelse blir også begrenset av tiden jeg har tilgjengelig, på grunn av Covid-19 pandemien ville jeg prøve å minske tiden min på skolen for å minske risikoen for smitte. Covid-19 var også grunnen til at jeg ikke ønsket å forske på flere skoler. Klassen ble valgt fordi deres mattelærer ønsket å bidra til min forskning da dette er et område lærerne mente det er behov for mer kunnskap om. Læren bistod med valg elevpar. Parene ble valgt utfra kunnskap på elevenes samarbeidsevne. Vi ønsket elevpar som normalt arbeidet godt sammen. Elevene som ble valgt ut til intervju var de samme fire parene som ble fokusert under observasjon.

3.6 Undervisningen

Opplegget besto av to undervisningsøkter i 6.trinn som ble gjort etter hverandre med en pause på 15minutter imellom øktene. I begge undervisningstimene ble programmeringsspråket scratch brukt, siden dette er et blokk-basert programmeringsspråk kan elevene raskt lære seg den grunnleggende bruken. Ut ifra funnene Dohn (2020) sin studie valgte jeg å gå for et undervisningsopplegg med åpen struktur, dermed kunne jeg tilpasse undervisningen underveis siden jeg ønsket å skape høy interesse for oppgavene. En åpen struktur gjør det mulig å tilpasse undervisningen til elevene og med høy grad av uavhengighet og fantasi, for ved det å minske risikoen for lav interesse av oppgavene.

Den første økten var en undervisningstime på 45 minutter. Siden jeg ikke hadde noen kjennskap til elevenes tidligere erfaringer i programmering og scratch, var hovedmålet for denne timen at elevene skulle klare å bruke grunnleggende ferdigheter i scratch. Undervisningsopplegget ble sendt noen uker før til klassens lærer for å sikre at undervisningen ville passe elevene. En endring som ble anbefalt av læreren var å endre gruppe størrelsen fra tre elever til to elever, dette var elevene mer vant med. Formålet med

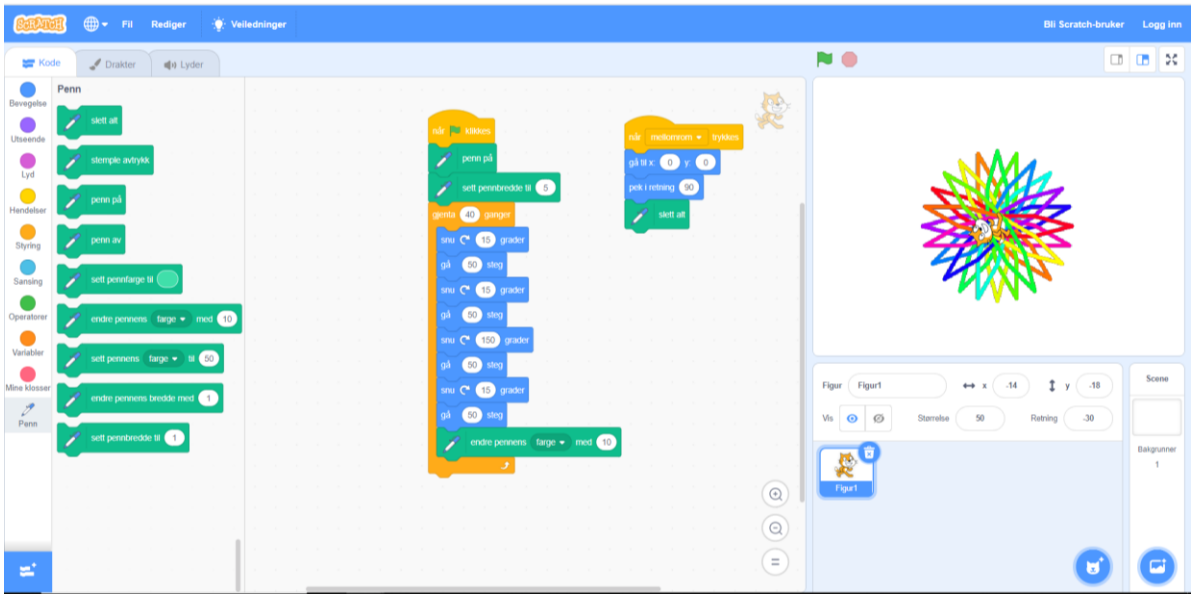
den første økten var at elevene skulle ha tilstrekkelig grunnlag for undervisningsøkten etter pausen. Læringsmålet for den første timen «klare å bruke grunnleggende funksjoner i Scratch» ble skrevet på tavla for å gi elevene et mål for timen. Den første økten ble startet med en introduksjon av meg og hvorfor jeg var på besøk. Deretter ble elevene inndelt i par eller grupper på to til tre elever, dette var læringspar som kontaktlæreren hadde bestemt og var grupper som elevene var komfortable med. Da elevene hadde satt seg i gruppene og fått opp en PC hver og kommet seg inn på scratch, startet jeg med en gjennomgang av hva et blokk-basert programmeringsspråk er og hvordan scratch leser blokkene. Videre viste jeg på smartboardet foran i klasserommet hvordan man legger inn blokkene i programmet samtidig som elevene imiterte det i gruppene. På smartboardet viste jeg elevene hvordan man legger inn penn funksjonen, og bruken av bevegelse, hendelser og styrings kodene i scratch. Det ble også vist hvordan man lager et program som resetter det tidligere program har tegnet, dette programmet ble kalt for viskelæret.



Figur 3 Viskelæret.

Etter en kort gjennomgang av det grunnleggende, som de hadde behov for i scratch, fikk elevene bruke den resterende tiden av økten på å utforske bruken av scratch, dette for at de skulle bli kjent med programmet gjennom lek og testing der de hadde fritt for å programmere det de ønsket. Når elevene er i utforskingen gikk jeg og læreren rundt for å veilede elevene med eventuelle problemer som dukket opp. Etter at elevene har prøvd seg frem i 10 minutter demonstrerte jeg hvordan man kan lage en fargerik blomst. Dette programmet kopierte

elevene og utforsket videre. På slutten av timen fikk elevene vise frem hva de hadde laget til resten av klassen.



Figur 4 Blomsten med koder som ble vist til klassen

Den andre undervisningsøkten varer i 90min der elevene får mulighet for å ta en luftepause underveis om de trenger det. Økten starter med en liten gjennomgang av den første økten før elevene får oppgave å programmere tre forskjellige geometriske figurer. Hva som er en geometrisk figur blir diskutert med elevene og eksempler som kvadrat, rektangel og sirkel blir trukket fram av elevene. Når elevene jobber med oppgaven går jeg rundt i klasserommet og veileder elevene, samtidig som jeg observerer hvordan de jobber med oppgavene. Elevene som løser oppgaven før de andre får tilleggsoppgaver ut ifra hva slags figurer de har laget fra før. Her vil oppgaven bli å programmere noe som krever videre utvikling av hva de har programmert fra før, elevene som blir ferdig får oppgaven om å programmere et rektangel. I slutten av økten blir det oppsummert og vist fram hvordan ulike figurer ble programmert og elevene blir utfordret på hvordan programmet kan kodes annerledes.

Siden jeg ønsket å se på hvilke løsningsstrategier og argumentasjon elevene brukte ønsket jeg ikke å påvirke arbeidet mer en det å lære dem å bruke Scratch. Derfor forsøkte jeg etter beste evne å ikke komme med veiledning som kunne påvirke prosessene for mye. Min hjelp bestod for det meste av rent tekniske råd når elevene kjørte programmet helt fast. Noen råd som kunne hjelpe dem på veien ble allikevel gitt der elevene så ut til å miste motivasjonen helt, eller sette seg fast i en stor blindvei.

3.7 Analysemetode

Kvalitative analysemetoder har først som hensikt å sortere innsamlet datamateriale, for så å kunne gjøre materialet forståelig. Mine teoretiske antakelser fungerer som filter når jeg skal analysere datamaterialet. Når jeg analyserer dataen tar jeg utgangspunkt i de teoretiske antakelsene, fordi dette gir struktur og retning (Christoffersen & Johannessen, 2012). Analysen skal gi mening til både det første inntrykket og til den avsluttende samlingen (Stake, 1995). Min dataanalyse starter i det jeg entrer klasserommet jeg skal ha undervisningen min i, som blir mitt forskningsfelt (Stake 1995; Christoffersen & Johannesen 2012). En analyse vil som regel ta utgangspunkt i en tekst som i mitt tilfelle er transkribert intervju og data fra observasjon, og skape en forståelse av tekstene (Christoffersen & Johannessen, 2012). Jeg ser videre på deler og elementer i tekstene, disse delene og elementene påvirker min mening som jeg finner i helheten som igjen vil påvirke min forståelse av de ulike delene og elementene (Christoffersen & Johannessen, 2012). Mitt mål med analysen er avdekke et mønster om hvordan elever arbeider med programmeringsoppgaver.

I en teoristyrt case som dette er, blir det brukt allerede eksisterende teori på området for å tolke funnene (Christoffersen & Johannessen, 2012). Teorien blir brukt mot funnene i analysen, og teorien vil hjelpe meg til å forstå praksis eller danne grunnlag for ny teori. De teoriene jeg bruker for å finne ut av hvordan elever jobber med programmeringsoppgaver er; Schoenfeld (1989) med støtte av Polya (1957) for å beskrive problemløsningsprosessen med Schoenfeld sine seks faser, 1)lese, 2) analysere, 3)utforske, 4)planlegge, 5) implementere og 6) sjekke. Jeg bruker Bocconi med flere (2016) for å se hvilke deler av algoritmisk tenkning elevene bruker med begrepene; abstraksjon, algoritmebehandling, automatisering, dekomponering, feilretting og generalisering. Når jeg ser på hvilke metoder eller strategier elevene bruker for å løse oppgaven, bruker jeg Brennan & Resnick (2012) sitt rammeverk om algoritmiske metoder med metodene, inkrementell og iterativ, testing og feilsøking, gjenbruk og blande, og abstraksjon og modulisering. For å tolke elevenes diskusjon og argumentasjon bruker jeg Kaufmann & Stenseth (2020) sine tre nivåer intuitiv og kodefokusert argument, argument basert på matematiske betraktninger og argumenter basert på generalisering, sammen med Lavy (2006) sine fire nivåer grunnleggende argument, sammensatt argument, utdypet argument og generalisert argument presentert som spesifikt.

Under min analyse av observasjoner og transkriberte intervju skrev jeg ned mine tanker, memos, om hvilke nivåer og begreper elevene brukte i undervisningen. Memos er et analytisk

verktøy og er mer enn bare et analyselager (Corbin & Strauss, 2015). Når jeg analyserer sammenligner, stiller spørsmål, kommer med påstander for en mening og foreslår mulige relasjoner mellom begreper, kan memos hjelpe meg her med å skape en dialog med meg selv, noe som er til ekstra stor nytte for meg som analyserer alene for å få utviklet en dypere analyse.

3.8 Relabilitet

Relabilitet vil si hvor pålitelige dataene som blir samlet inn er, og knyttes til nøyaktigheten av undersøkelsen ved; hvordan dataene er samlet inn, hvilke data som brukes og hvordan de bearbeides (Christoffersen & Johannessen, 2012). For å teste relabiliteten til dataene kan gjenta sammen undersøkelse med 2-3 ukers mellomrom og om resultatene blir like er det høy grad av relabilitet (Christoffersen & Johannessen, 2012). Postholm & Jacobsen (2018) påpeker at i en kvalitativ studie vil det være svært vanskelig å replikere fordi møtet mellom forsker og elever vil fortone seg forskjellig. Det vil fortone seg forskjellig siden ulike forskere bringer med seg sin subjektive individuelle teori og alle mennesker er i utvikling (Postholm & Jacobsen, 2018), dette vill spesielt gjelde elever som er i konstant utvikling. Man kan også si at relabiliteten er høy om andre forskere som undersøker samme fenomen får samme resultat (Christoffersen & Johannessen, 2012). Man kan se lignende trender fra lignende studier (se kap. 5.2) som selv om det ikke blir brukt likt undervisningsopplegg så hever det relabiliteten til studien. Men i denne studien vil man ikke få samme resultat om man hadde gjort samme undersøkelse noen uker etterpå. Dataene vil bli annerledes siden problemløsning og programmeringsoppgaver baserer seg mye på tidligere erfaringer med tema, siden elevene ville hatt mer erfaring ved en andre gjennomgang av undervisnings opplegget ville dataene mest sannsynlig bli ulike. Postholm & Jacobsen (2018) mener at å reprodusere en kvalitativ studie med hensyn av å måle studiens relabilitet har ingen betydning.

Postholm & Jacobsen (2018) peker på at man burde knytte relabilitet til refleksjon over hvordan undersøkelsen og forskeren kan ha påvirket resultatet. I starten av undervisningen forklarte jeg hvorfor jeg var på besøk og fortalte at jeg ikke skulle se på hva som var rett og galt, men på hvordan de jobbet og løste oppgavene der det ikke var noen fasit, dette ble også sagt i intervjuet og at det er de som har det riktige svaret og ikke meg. Selv om dette ble sagt så vil det fortsatt være en mulighet for at elevene tilpasset seg etter hva de trodde at jeg ønsket å få av svar og Postholm & Jacobsen (2018) påpeker at dette er noe man heller ikke kan kontrollere fult. Med bruk av observasjonsnotater underveis og rett etter timen, sammen med

transkribert intervju der intervjuet ble tatt opp med diktafon øker jeg relabiliteten til dataene ved å sikre at informasjonen ikke blir endret av min hukommelse og det blir lettere å friske opp hukommelsen under analyse arbeidet.

I min studie ønsker jeg å se på hvordan elever argumenterer og strategier de bruker for å løse programmeringsoppgaver, siden programmering er nytt i skolen har de fleste elever lite erfaring og kunnskap om dette fra før er dataene mine av interesse.

3.9 Validitet

Intern validitet dreier seg om i hvor stor grad det er samsvar mellom den virkeligheten jeg påstår at jeg studerer og analyserer, og teorier jeg benytter for å beskrive denne virkeligheten (Postholm & Jacobsen, 2018). Ved å se på hvordan tidligere forskning ser på hvordan elever jobber med problemløsnings oppgaver og type argumentasjon elever bruker når de løser programmeringsoppgaver måler jeg hvilke løsningsmetoder og argumentasjon elevene bruker i programmeringsoppgaver. Schoenfeld (1989) med støtte av Polya (1957) beskriver problemløsnings prosessen som kan forklare hvorfor noen av parene brukte forskjellige løsningsmetoder. Brennan & Resnick (2012) sitt rammeverk om algoritmiske metoder kategoriserer ulike metoder som elever bruker i programmeringsoppgaver, dette rammeverket bruker også jeg når jeg skal kategorisere elevenes løsningsmetoder. Bocconi med fler (2016) viser begreper innenfor algoritmisk tenkning som kan forklare argumentasjonen elevene bruker innenfor programmering. Kaufmann & Stenseth (2020) og Lavy (2006) sin studie lager et rammeverk for å kategorisere nivåer av argumentasjon, disse rammeverkene bruker også jeg for å kategorisere elevene sine argumentasjons nivåer. Det at jeg kan knytte mine data mot rammeverkene til Kaufmann & Stenseth (2020), Lavy (2006) og Brennan & Resnick (2012) øker graden av validitet i min studie.

3.10 Overførbarhet

Grad av overførbarhet går ut på om funn fra en konteks kan generaliseres eller overføres til andre kontekser (Postholm & Jacobsen, 2018). Min studie tar bare utgangspunkt i noen elever fra en klasse og kan ikke generaliseres til å gjelde alle elever på 6.trinn i Norge siden alle elevene i Norge ikke har like erfaringer som elever i min studie. Men likheten mellom min studie og tidligere studier viser at man kan forvente å få noe lignende resultat om man hadde gjennomført samme undervisning i en annen klasse.

3.11 Forskningsetikk

I et forskningsprosjekt er det viktig at forskeren tar hensyn til de etiske retningslinjene som gjelder for forskning. Retningslinjene er utarbeidet av den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora (NESH), og nevner blant annet informert samtykke, anonymitet og personvern (NESH, 2016). Under vil jeg ta for meg disse hensynene og beskrive hvordan det ble ivaretatt i denne studien.

Før jeg dro til skolen for å gjøre forskningen min meldte jeg studien min til Norsk senter for forskningsdata (NSD), og fikk godkjent studien. Informert samtykke skal være frivilig, uttrykelig og informert erklæring fra den som blir forsket på om at vedkommende godtar å bli forsket på, ved informert menes at den som skal delta skal ha nødvendig opplysninger om undersøkelsen (Christoffersen & Johannessen, 2012). Siden elevene var under 15 år lagde jeg et samtykke skjema der både eleven og elevens foresatte skulle signere. Jeg sendte samtykke skjema noen uker i forveien slik at alle elevne og foresatte skulle få god tid på å vurdere om de ville være med. En skisse av undervisningen ble også sendt til rektoren på skolen og klassens kontaklærer, slik at de skulle få innblikk i hva som skulle bli gjort. Læreren tok i forkant av besøket å informerte og forklarte elever og foresatte om timen og forskningen for at de skulle bli best mulig informeret. Elevene blir anonymisert og intervjuene blir slettet etter endt studie for at min forskning ikke kan spores tilbake til dem.

4 Resultat

Målet med studien er å belyse hvilke løsningsstrategier elevene bruker og hvordan de argumenterer. I dette kapittelet vil jeg presentere funn og analyse av fire elevpar fra undervisningstimen. Presentasjonene av parene er fra observasjon og intervju.

4.1 Funn fra observasjon og intervju

Observasjon ble gjort i den andre undervisningstimen, det var også denne timen som var hovedfokuset under intervju, ettersom det var data fra denne timen som ville hjelpe meg å svare på forskningsspørsmålet. I denne delen vil jeg belyse funn fra hva elevene gjorde i undervisningen, hvordan de løste oppgavene og hvordan de argumenterte i timen. En gjenganger hos mange elever som jeg vil forklare før funnene, var at de kalte en figur med flere kanter en 8 for en sirkel da det lignet på en sirkel. Denne sirkelen så ut som en sirkel der den ble tegnet mens i programmet ser man at det ikke er en sirkel, siden denne mangekanten ikke har like lang avstand til sentrum fra alle punktene er dette ikke en sirkel, men vil bli kalt sirkel som en samling på mangekanter elevene kalte sirkel.

4.1.1 Par 1

Observasjonen viser at par 1 var de eneste som klarte å lage et avlangt rektangel. Av geometriske figurer laget de kvadrat, en mangekant som de kalte sirkel og et avlangt rektangel. Kvadratet ble programmert på denne måten: penn på – gjenta 4 ganger – gå 50 steg – snu 90 grader. «Sirkelen» ble programmert med kodene: penn på – gjenta 20 ganger – gå 30 steg – snu 20 grader, denne figuren ble en mangekant som lignet på en sirkel, men har ikke egenskapene til en sirkel. Den ble godkjent av læreren som en sirkel siden tegningen så veldig lik ut en sirkel. De første to figurene hadde Petter og Gunnar lite utfordringer med å programmere og det ble ikke brukt mye testing for å få til programmet, de hadde laget et kvadrat i den første undervisningstimen og på sirkelen måtte de bare få den til å passe inn i vinduet det ble tegnet på ved å endre skritt. Programmet til rektangelet ble skrevet slik: pen på – gå 100 steg – snu 90 grader – gå 50 steg – snu 90 grader – gå 100 steg – snu 90 grader – gå 50 steg – snu 90 grader. Alle programmene ble startet med hendelse blokken «når grønt flagg trykkes». Fra mine observasjoner av par 1 så det ut som de hadde godt samarbeid og diskuterte underveis hvordan programmet skulle kodes. Dette ble også bekreftet under intervjuet:

K: Hvordan jobbet dere med oppgavene?

Petter: Vi dro in blokkene og trykket på flagget.

K: Hva gjorde dere om det ikke ble riktig?

Petter: Vi snakka litt også endret vi et tall.

Gunnar: Vi endret skrittene og testet det ut på nytt

K: Hvorfor endret dere skrittene?

Petter: Den måtte bli lengre

På kvadrat og «sirkelen» brukte de løkke funksjonen «gjenta n ganger», men på rektangelet brukte de ikke denne funksjonen etter som det var lettere for dem å ikke bruke den.

K: Hvorfor var rektangelet vanskelig?

Petter: Vi måtte bruke flere linjer

Gunnar: Vi kunne bruke gjenta på kvadrat.

K: Hvorfor brukte dere ikke gjenta på rektangelet?

Gunnar: Det var lettere uten

Par 1 brukte litt ulike løsningsmetoder, der gjenta ble brukt la de inn alt de trodde skulle gi hele figuren også kjørte programmet, på rektangelet gikk de mer stegvis fram. Her lagde de først en linje også en 90 grader etterfulgt av en kortere linje en den første linjen. Par 1 så da at de var halvveis og la videre inn det motsatte av det de gjorde først.

K: Hvordan laget dere rektangelet?

Petter: Vi tok litt og litt.

Gunnar: Vi tok først den lange streken under også prøvde vi å få vinkelen opp igjen.

K: Tok dere stegvis hele veien?

Gunnar: Nei når vi kom halvveis gjorde vi de samme bare motsatt.

4.1.1.1 Analyse av par 1

Par 1, Petter og Gunnar brukte flere forsøk før de fikk den geometriske figuren de var ute etter, når de endret kodene hadde de en tanke om at denne endringen kan gi ønsket resultat. De brukte kort tid på å lese oppgaven og brukte mest tid på implementering (Schoenfeld, 1989). Det ble brukt litt analyse når de gikk fra kvadrat til sirkelen. Da måtte de teste ut forskjellige versjoner med variablene, men dette tok ikke noe stor del av tiden sammenlignet med implementering og sjekking, her bruker de metoden testing og feilsøking (Brennan & Resnick, 2012). I starten av undervisningen jobber Petter og Gunnar likt hvordan elevene i Schoenfeld (1989) sin studie, ved å bruke lite tid på å lese og analysere oppgaven og heller implementerte og testet. Da de skulle programmere rektangelet derimot måtte Petter og Gunnar bruke mer tid på å analysere, utforske og planlegge, her bruker de mere tid på analyse og utforskning sammenlignet med de to figurene før. De så etter hvert at det å programmere et rektangel ut ifra programmet til kvadratet ikke vil fungere med bare å endre lengden siden dette bare gjør at kvadratet øker i størrelse. De koder derfor et nytt program og bruker metoden inkrementell og iterativ (Brennan & Resnick, 2012) uten bruk av løkkefunksjonen, når de har kommet halvveis ser elvene at dette vil fungere og at de kan gjøre det «motsatte» å få et rektangel. Petter og Gunnar hadde lite utfordringer med å programmere kvadratet og «sirkelen» og når de snakket om hva de måtte gjøre av endringer i intervjuet, snakket de om arbeidet med rektangelet. Modifiseringene de gjør når de skal programmere rektangelet viser at de vurderer kodene i programmet lager en hypotese på at man kan programmere et rektangel uten løkke og finner en annen retning for å komme fram til løsningen.

Petter og Gunnar er innom flere av kjernebegrepene i algoritmisk tenkning (Bocconi er al, 2016), man av variabel grad. De automatiserer når de benytter metoden testing og feilsøking (Brennan & Resnick, 2012), ved å legge inn kodene for programmet og lar programmeringsspråket gjøre jobben, men graden av automatisering minsker ved rektangelet når de må bruke inkrementell og iterativ metode (Brennan & Resnick, 2012). Når Petter og Gunnar ser at de er halvveis i programmeringen av rektangelet følger de samme oppskrift for neste halvdel og her ser vi bruken av algoritmebehandling ved at de følger instruksjonene til den første halvdelen for den andre halvdelen. Par 1 dekomponerer når de bryter ned programmet til kvadratet og ser at de trenger noe annet for at rektangel. De programmer kode for kode uten bruk av løkke funksjonen for å gjøre arbeidet mindre komplisert eller «lettere» som Gunnar sier. Der er ikke stor grad av abstraksjon selv om de ser at programmet til kvadrat kan brukes til «sirkelen» ved å endre variablene og dermed kan lage mange figurer ut

ifra base kodene til kvadratet. Så ser de ikke det større bilde når de er halvveis i programmeringen av rektangelet at de kan bruke løkkefunksjonen. De derimot generalisere godt med kvadrat programmet når de ser at de kun kan endre på variablene for å få en «sirkel».

I starten holder Lars og Petter seg til den algoritmiske metoden testing og feilsøking (Brennan & Resnick, 2012). Gjennom å endre variabler i kvadrat programmet opp mot en sirkel, de har god forståelse over hva variablene gjør og utforsker seg gradvis med flere kanter. Det er også gjennom denne metoden at de endrer metode til inkrementell og iterativ når de skal kode et rektangel. Her går de stegvis fram med strek for strek fram til de er halvveis.

Når de jobber med feilrettingen, bruker de sammensatt argument av Lavy (2006) sine fire nivåer. Petter og Gunnar ser og forstår hva kodene i programmet gjør med programmet og hvilke variabler som utfører den gitte oppgaven i programmet, de endrer kvadrat programmet til et sirkelprogram, men de klarer ikke å utvikle seg videre til nivå tre når de jobber med rektangelet ved å utvikle programmet til å også kunne brukes på et kvadrat. Siden de ikke oppnår utdypet argumentasjon (Lavy, 2006) som er en del av argumenter basert på matematiske betraktninger til Kaufmann og Stenseth (2020) viser Petter og Gunnar intuitiv og kodefokusert argumentasjon (Kaufmann & Stenseth, 2020). Petter og Gunnar analyserer hva kodene gjøre og hvordan variablene påvirker programmet, endringene de gjør før de tester på nytt har en hypotese og de gjør derfor ingen tilfeldige endringer

4.1.2 Par 2

Par 2 jobbet veldig raskt med de tre første figurene, de laget figurene kvadrat, 8-kant (oktogon) og en mangekant som de kalte sirkel. De kjørte mange tester over kort tid der de gjorde små endringer mellom vært forsøk. Siden disse to var veldig raske med å lage 3 figurer, fikk de en ekstra utfordring med å lage et rektangel. De tre første figurene ble programmert ganske likt. Kodene de brukte var penn på – gjenta n ganger – gå 50 steg – snu n grader. På kvadratet ble det gjentatt fire ganger og snu 90 grader. På 8-kanten ble det gjentatt 12 ganger og snu 45 grader. «Sirkelen» ble programmert i lik stil som par 1 bare med å gjenta 200 ganger, gå 15 steg og snu 10 grader. På rektangelet prøvde de å bygge på koden fra kvadratet, men de kom ikke fram til en løsning.

K: Hvordan prøvde dere å lage rektangelet?

Lars: Vi prøvde å gjøre den firkanten vi hadde lengre, men den ble bare større.

Anna: Den var litt vanskelig.

De løste oppgavene med å kode hele programmet uten noe diskusjon eller hypotese på hva programmet ville tegne. Ble ikke programmet en geometrisk figur, endret de den til de fikk en figur.

K: Gjorde dere noe før dere kjørte kodene i programmet

Anna: Nei egentlig ikke, vi testet med å trykke på det grønne flagget også om det ikke funket så vi igjennom koden for å se hva som måtte skiftes.

K: Hva var det dere endret på kodene?

Lars: Vi testet ut med å skifte lengdene.

Lars og Anna viste hvordan et typisk rektangel skulle se ut men det ble vanskelig å kode programmet til å få to parallelle linjer med forskjellige lengde ut i fra at de prøvde å programmere med kun to koder i løkken, siden det hadde funket på alle de andre figurene.

K: Prøvde dere på rektangel?

Lars: Jeg prøvde, men fikk det ikke til.

Anna: Nei, det var litt vanskeligere

K: Hvorfor var rektangelet vanskeligere?

Anna: Fordi det er ikke det samme utover hele, det er jo forskjellige lengde.

Lars og Anna satt lenge for å prøve å få til rektangelet, men dette ble etter hvert kjedelig og de mistet motet, de fikk da beskjed om at de kunne prøve å lage flere figurer istedenfor.

4.1.2.1 Analyse av par 2

Lars og Anna jobbet veldig raskt med oppgavene, og på de tre første figurene endret de stort sett bare på variablene for å få flere antall kanter. De gjør en liten endring i programmet tester programmet og gjør en liten endring på nytt før det testes igjen. For hver testing nærmer figuren seg en sirkel og de bruker denne erfaring til å fortsette å endre til figuren blir en sirkel. Denne løsningsstrategien er lik Brennan og Resnick (2012) sin metode testing og feilsøking.

Her går det rett fra å lese oppgaven til implementering og sjekke resultatet (Schoenfeld, 1989), denne syklusen holder de seg til i utviklingen av de tre første figurene. Når de får en ekstra utfordring med å lage et rektangel bremses det raske tempoet og de blir nødt til å analysere og utforske mer av hva som skjer i kodene. Det som tidligere fungerte, fungerer ikke lengre. Anna og Lars prøver å se hva som kan endres for å lage et rektangel, men etter noen forsøk med endringer av variablene i programmet, mister de interessen og gir opp.

De har høy grad av automatisering (Bocconi et al 2016) når de gjør testing og justeringer i bruk av metoden testing og feilsøking (Brennan & Resnick, 2012). Det er et høyt antall forsøk som gjøres over kort tid, der de bruker utregningshastigheten til programmeringsspråket som støtte i dette tempoet. Når de jobber seg fra et kvadrat, til oktogon og videre til sirkelen endrer de litt på en av variablene før de sjekker resultatet. Par 2 jobber seg oppover med flere og flere kanter med å øke gjentatte ganger og grader snu, på sirkelen måtte de også justere antall skritt for at figuren ikke skulle bli for stor for vinduet, bruk av endring og justering her viser hvordan de jobber med feilretting. De generaliserer designet av programmet og endrer kun variablene når de skal bygge videre mot en ny figur. Lars og Anna er innom dekomponering når de forsøker å gjøre et kvadrat om til et rektangel og ser på hver enkelt kode og hva de gjør, men klarer ikke å se problemet med at det trengs flere koder i programmet. Graden av automatiseringen er høy på grunn av bruken av løkke funksjonen som gjør testingen mer effektiv.

Anna og Lars bruker stort sett den algoritmiske metoden testing og feilsøking (Brennan & Resnick, 2012). På veien fra kvadrat til sirkel bygger de seg oppover mot flere og flere kanter ved å endre litt og litt på alle variablene. Strategien de kommer med her er å øke antall ganger løkken skal gjentas samtidig som grader det skal snus og antall skritt minsker, denne strategien utviklet de ved å eksperimentere seg fram.

Argumentasjonen til Anna og Lars går fra grunnleggende argument over til sammensatt argument (Lavy, 2006). De danner seg en generalisering på at slik som dette programmet ser ut kan vi lage alle figurer. Videre i veien mot sirkelen ser de at variablene må endres for at det skal komme nærmere og bruker tallenes egenskaper, dette løfter dem opp på et sammensatt argument. Utviklingen til utdypet argument stoppes ved at de ikke klarer å utvikle strategien sin med å legge til flere koder.

4.1.3 Par 3

Par 3 hadde et godt samarbeid og syntes det var spennende og utfordrende å bruke scratch til å lage geometriske figurer. Figurene de lagde var et kvadrat og en mangekant som lignet en sirkel, i tillegg prøvde de seg på et rektangel. På et kvadrat brukte de – penn på – gjenta 4 ganger – gå 50 skritt – snu 90 grader. På «sirkelen» brukte de – penn på – gjenta 50 ganger – gå 15 skritt – snu 10 grader. De fikk ikke til rektangelet, men prøvde å bygge fra et kvadrat med å bruke løkkefunksjonen gjenta 4 ganger og kun to koder inne i løkken, de prøvde også å legge til noen koder etter løkken og fikk noe som lignet på en propell.

K: Hva var det som var vanskelige oppgaver?

Per: Å finne ut hvordan man skulle tegne rektangelet.

Line: Ingen visste jo hvordan det skulle være.

K: Hvorfor var det vanskelig?

Per: Jeg vet ikke, vi prøvde å lage noe også ble det noe helt annet.

K: Åja hva fikk dere da?

Per: Vi fikk et kvadrat ved siden en propell av noe slag.

Par 3 løste oppgavene med å teste dem ut og endre på en eller flere av kodene når programmet ikke ble riktig, her ble alltid et komplett program kodet før de testet det ut. Det de endret på i kodene var som regel skritt og grader, de avtalte sammen hva de kunne endre og kjørte programmet på nytt. Jeg observerte også at de prøvde å legge til koder under kvadrat programmet, dette ga propellen som Per nevner. Bortsett fra forsøket som endte med propell var endringene rettet mot skritt og grader.

K: Når dere skulle kode, snakket dere noe om hvordan koden skulle se ut *før dere kodet eller testet dere koden underveis.*

Per: Vi testet dem

Line: Ja vi testet dem litt ut.

Per: Vi testet og testet litt også kom vi fram til svaret etter hvert, så vi brukte noen forsøk på å få det til.

K: Hva var det dere gjorde av endringer når dere testet?

Line: Antall skritt og grader

4.1.3.1 Analyse av par 3

Par 3 hadde et godt samarbeid og de hadde en god balanse når det kom til forslag som kunne hjelpe programmet. Selv om par 3 hadde et godt samarbeid og virket vant med å argumentere sammen så gikk de raskt fra å lese oppgaven til implementering og testing (Schoenfeld, 1989). Her bruker de datamaskinen og programmeringsspråket sin raske regnekraft til å få fortgang på arbeidet så de får testet endringene raskt. Kvadratet lager de raskt og jobber seg mot ved å endre på variablene der figuren blir mer og mer lik en sirkel. Programmets design er like med kun endringer av variabler. Det meste av testingen på de første to figurene var på sirkelen der den måtte bli rund nok og ikke for stor. «Gjenta 50 ganger» er mer enn de trenger til denne figuren der de hadde holdt med «gjenta 36 ganger», men det viktigste for dem var at figurer skulle bli hele. Når de skal programmere rektangelet og ser at endring av variabler ikke er nok analyserer de koden og prøver å legge til koder for å bygge videre på rektangelet. Det at de må bruke mere tid på å analysere og utforske blir for Line og Petter en brems i prosessen mer enn en motiverende utfordring og elevene gir opp.

Den algoritmiske metoden Line og Per benytter seg mest av er testing og feilsøking (Brennan & Resnick, 2012). De baserer sitt neste forsøk på erfaringer fra tidligere forsøk og gjør endringer deretter. De kjører flittig med tester noe som tyder på at de ikke tenker så mye over hva som kommer til å skje ved den spesifikke endringen i variabelen, men lar heller datamaskinen gjøre jobben for dem. Line og Petter prøver å bygge videre på programmet de har, men ender opp med en propell. Her gir ikke svaret på endringene noe klart svar på hva som gikk galt siden det virket som for mye gikk galt for å bygge videre på ideen.

Line og Per benytter seg mye av datamaskinens regnekraft og har høy grad av automatisering (Bocconi et al, 2016), alle forsøkene inneholder bruk av løkke funksjonen som gjør at de kan bruke færre koder (blokker) i programmene de tester. Ved feilretting så tester de ut programmet, endrer variabler ofte flere samtidig. Elevene lager seg først en generalisering med at programmet til kvadratet må fungere for alle figurer siden også fungerte på sirkelen. Generaliseringen her holder ikke når de prøver seg på et kvadrat. Løkke programmet kan også sies å gi dem noe grad av algoritmisk behandling på grunn av kopieringen av programmet fra

kvadrat til sirkel. Line og Per klarer ikke å dekomponere som også brukes i testing og feilsøking (Brennan & Resnick, 2012), mangelen på dekomponering kan være grunnen til at de ikke får programmert rektangelet.

Par 3 starter med grunnleggende argument (Lavy, 2006) i timen ved holde seg til et program design på to figurer og mener da at dette vil da fungere på andre figurer også, dette blir også innenfor intuitiv og kodefokusert argument (Kaufmann & Stenseth, 2020). De møter på problemer når de skal prøve seg på et rektangel og endringer av på variablene ikke er nok. Line og Per oppnår også sammensatt argument (Lavy, 2006) siden de bygger videre fra grunnleggende argument ved å forstå hva variablene gjør. De endrer gradene og ser at for å få sirkelen så må det snus få grader for at den skal se rund ut. Par 3 endrer også antall skritt for å minske størrelsen til sirkelen. Det er ved rektangelet at utviklingen stopper når de ikke klarer å endre programmet til å også gjelde et kvadrat.

4.1.4 Par 4

Paret hadde litt tettere oppfølging fra læreren en fra meg og jeg fikk observert de mindre en de andre parrene. Dette paret valgte å ikke bruke løkke funksjonen og programmerte inn strek for strek.

K: Brukte dere løkke funkjsonen?

Bård: Nei

K: Hvorfor brukte dere ikke den ikke?

Bård: Vi hadde vel egentlig ikke bruk for den

Sondre: Det var lettere uten.

I tillegg til tre geometriske figurer lagde de også en sammensatt figur av mange kvadrater. Figuren fant de i læreboka og bestod av 12-17 kvadrater. Når de programmerte denne la de inn koden for et til tre steg i figuren før de testet den.

K: Hvordan lagde dere figurene?

Rune: 90

Bård: Vi brukte 90grader

Sondre: Ja og 20skritt!

Bård: Ja vi la de etter hverandre.

K: Var dere sikre på at kodene var riktige?

Bård: Vi testet kodene hele tiden for å se om streken gikk riktig vei

Sondre: Det ble ganske mye testing og litt slitsomt

K: Hvor mange koder la dere inn før dere testet?

Bård: 1 til 3 koder.

K: Prøvde dere å løse oppgavene på en annen måte?

Sondre: Nei vi bare testet.

De hadde diskusjon underveis på hva som var neste steg for å kunne lage figuren. Diskusjonene drev seg stort sett om hvilken 90 grader som var riktig, ble det feil gjorde de det motsatte siden det ville gi riktig resultat. Det ble gjort færre feil på slutten av timen en det ble gjort på starten av timen.

K: diskuterte dere noe underveis

Sondre: Vi diskuterte litt på hvilken vei streken skulle gå.

Bård: Vi hadde en plan på hvor den skulle gå, men det ble ikke alltid riktig

Par 4 syntes det var tidkrevende å programmere siden de trengte mange blokker til programmet. De var klar over at det fantes andre metoder som kunne være aktuell til en annen gang som kunne gjøre arbeidet enklere

K: Hadde dere løst oppgavene på en enklere måte om dere hadde gjort det på nytt?

Sondre: Ja det hadde vi kanksje

Bård: Ja kanskje

Sondre: Om det kanskje var en enklere måte som ikke tok så lang tid

Bård: Ja det finnes det sikkert.

4.1.4.1 Analyse av par 4

Det som skilte Par 4 fra de andre parene var at de tok seg bedre tid med oppgavene. Dette var det paret som tenkte mest likt matematikeren i Schoenfeld (1989) sin studie. Selv om de tok seg bedre tid med oppgavene, var de fortsatt rask med å lese oppgaven og meste parten av tiden ble også her brukt på implementering og sjekke. Men par 4 tok et valg om å ikke bruke løkke funksjonen og heller ta implementeringen skritt for skritt, også teste underveis før programmet var ferdig kodet. Her analyserer Par 4 problemet og ser for seg hvordan man kan løse problemet med den sammensatte figuren på en tidkrevende men enkelmåte. Det kom også fram at de viste om løkke funksjonen, men valget med å ikke bruke denne var en beslutning de tok før de startet å implementere skrittvis.

Den algoritmiske metoden inkrementell og iterativ (Brennan & Resnick, 2012) bruker par 4 ved alt av implementering. Par 4 går stegvis fram med alle figurene og gjør en eller ingen endring på blokkene. Denne prosessen blir også gjentatt mye. De tester og gjør feilsøking også når de har lagt inn neste steg og tar med seg litt erfaring fra neste steg, det tar litt tid før erfaringene blir tydelige siden de i starten tester mer tilfeldig. Det blir også brukt gjenbruk og blande når de bruker en allerede tegnet figur i læreboken som mal i programmeringen av den sammensatte figuren.

Begrepene rundt algoritmisk tenkning var blant annet feilretting (Bocconi et al, 2016), der de testet ut hvilken vei 90 graden skulle snus, her ble det gjort motsatt endring om det ikke ble ønsket resultat og ingen endring ved ønsket. Siden det også ble enklere for dem mot slutten av timen å kode riktig 90 grader, ser jeg også at de brukte erfaringene sine fra tidligere koder til å få høyere treffsikkerhet med kodingen. Det var også høy grad av algoritmisk behandling siden de brukte en figur fra læreboka når de skulle programmere den sammensatte figuren. De delte også opp denne figuren for å lettere kunne programmere ved å ta «snu grader» blokk og «antall skritt» blokk, her bruker elevene dekomponering. Bruken av disse begrepene er de som vises i inkrementell og iterativ metode.

Diskusjonen eller argumentene elevene i par 4 brukte kan man først knytte til grunnleggende argument (Lavy, 2006), ved å se på hvordan de jobber med å bruke metodene. De ser at de vil klare å lage figurene ved å legge skritt og grade blokkene etter hverandre og holder seg til denne metoden siden de ikke får behov for å endre denne. Det gjøres enkle analyser når de

tester ut programmet siden det eneste av retting de er nødvendig å gjøre er å endre retningen på 90 graderen for at programmet skal bli et steg nærmere den hele figuren. Syklusen og argumentasjonen er her lik intuitiv og kodefokusert argument (Kaufmann & Stenseth, 2020). Elevene her forstår at det kan gjøres med løkker, men de ville holde seg til en enklere løsning. Siden de møter på noe som utfordrer den generaliseringen de har som gjør at det kreves en utvikling av arbeidsprosessen utvikler heller ikke argumentasjonen seg til utdypet argument (Lavy, 2006) og argumenter basert på matematiske betraktninger (Kaufmann & Stenseth, 2020).

5 Diskusjon

5.1 Se parene i sammenheng

De fleste parene begynner med å lese oppgaven raskt og går deretter rett videre til implementering og testing (Schoenfeld, 1989). Unntaket her er Par 4 som tar seg litt tid til å analysere oppgaven og legger en plan der de bestemmer seg for å lage figurene uten å bruke løkke, siden dette vil gi mindre problemer. Par 2 og 3 holder seg stort sett til samme rutine med å gjenta implementering og testing, selv om de analyserer og planlegger litt når de går videre fra kvadrat til sirkel. Den største forandringen ser man hos par 1 når de skal lage et kvadrat. Par 1 startet med å utforske hva som skjer om man endrer skritt i kvadrat programmet, når de ser at det ikke vil løse problemet analyserer de hva som kan løse problemet og går for en ny plan som fungerer. Par 2 og 3 stopper her opp i prosessen og klarer ikke å programmere et rektangel. De utforsker med å endre på variablene i programmet de allerede har og par 3 prøver i tillegg å legge til koder etter løkken. Par 2 og 3 tar seg ikke tid til å analysere.

Par 2 og 3 er også veldig like innenfor begrepene til algoritmisk tenkning der begge bruker automatisering, feilretting, algoritmisk behandling og de lager en lik generalisering (Bocconi med flere, 2016) om programmet. Par 1 har i starten lik generalisering som par 2 og 3 men denne blir brutt og endret i programmeringen av rektangelet. Noe av det som kan være grunnen til at par 1 klarer å endre generaliseringen er at de dekomponerer programmet og gjennom dekomponeringen klarer å forstå bedre hva de ulike kodene gjør (Bocconi med flere, 2016). Par 4 dekomponerer også når bryter ned den sammensatte figuren ned til løsbare problemer. Ulikt fra de andre parene så viser ikke par 4 noe stor grad av generalisering i form av noen program som fungerer for flere figurer, men de generaliserer i at de kan lage det de trenger med metoden de bruker, men dette gjør det ikke noe enklere for dem.

Ser vi på par 4 så er de de som bruker den algoritmiske metoden inkrementell og iterativ (Brennan & Resnick, 2012) mest. Par 4 benytter seg også av gjenbruk og blande og testing og feilsøking, men her er det mest gjenbruk av en figur fra boken og testing av sekvensene i koden, siden de jobber iterativt er det lite feilsøking på grunn av det er å endre snu retning som blir gjort av endring. I motsetning til par 4 bruker par 2 og 3 kun metoden testing og feilsøking (Brennan & Resnick, 2012). Begge parene holder seg til testing og feilsøking gjennom hele undervisningen, der de stort sett tester ut forskjellige kombinasjoner av variablene for å programmere forskjellige figurer. Par 1 bruker i første del av undervisningen

testing og feilsøking siden dette fungerer godt for dem i prosessen fra kvadrat til sirkel. Under programmeringen av rektangelet bytter de metode fra testing og feilsøking over til inkrementell og iterativ. Par 4 og par 1 bruker begge disse metodene, men forskjellen her er at par 4 kombinerer testing og feilsøking inn i inkrementell og iterativ metode ved å teste steg for steg. Par 1 går over til å kun bruke inkrementell og iterativ metode der de tar seg bedre tid med hver sekvens og når de kommer halvveis utvikler de resten av programmet basert på erfaringer med første halvdel.

Måten parene argumenterer på og utvikler argumentasjonen er relativt likt mellom alle parene. Parene starter først med grunnleggende argument (Lavy, 2006) og bygger seg videre i timen opp mot sammensatt argument (Lavy, 2006). Par 1 er det paret som først viser sammensatt argument og som er nærme med å oppnå utdypet argument (Lavy, 2006). Hadde par 1 klart å koble kodene inn i en løkke funksjon og da sett at det nye programmet kan brukes til både kvadrat, sirkel og rektangel hadde de vist utdypet argumentasjon og argumenter basert på matematiske betraktninger (Kaufmann & Stenseth, 2020).

Alle parene viser intuitiv og kodefokusert argument, men ingen oppnår argumenter basert på matematiske betraktninger. Par 1 er som sagt nærmest imens de andre parene er litt lengre bak i utviklingen til dette nivået. Lavy (2006) viser til at utdypet argument skjer når konklusjonen eleven har i sammensatt argumentasjon ikke lenger passer eller et nytt problem dukker opp der tidligere argumentasjon ikke blir gyldig. Der dette som skjer når par 1, 2 og 3 skal programmere et rektangel, det som fungerte på kvadrat og sirkel fungerer ikke lengre og en utvikling av programmet blir krevd for å klare å løse problemet. En felles nevner for parene er mangelen på bruk av abstraksjon. Som Wing (2017) viser til så er abstraksjon det vanskeligste å bruke innenfor algoritmisk tenkning og det kan være dette som er grunnen til at elevene ikke kommer seg videre til argumenter basert på matematiske betraktninger. Der par 2 og 3 ikke kommer like langt i som par 1 i programmeringen av rektangelet, kan skyldes at de bruker lite tid på å analysere programmet og ikke dekomponerer programmet nøye nok til å se en ny retning som kan løse problemet. En annen årsak kan være at par 2 og 3 fikk rektangelet som en ekstraoppgave siden de var ferdige med 3 figurer, imens par 1 valgte rektangelet av fri vilje. Siden rektangelet ble selv valgt som en figur av par 1 kan motivasjonen og ønsket til å klare det høyere for dem enn det var for par 2 og 3.

5.2 Sammenligne med tidligere forskning

I problemløsningsprosessen jobber parene veldig likt med hvordan elevene i Schoenfeld (1989) sin studie jobber med problemet. Parene leser raskt over oppgaven og starter med å løse problemet. Når par 1 møter på et vanskeligere problem ved rektangelet, og må finne en helt ny retning, er prosessen deres mer lik matematikeren, selv om jeg vil tro at matematikeren ville forsøkt å lage et mer generaliserbar og effektivt program ved å bruke løkke funksjonen.

Sammenligner man paret i Lavy (2006) sin studiene med parene i min studie, så oppnår paret i Lavy sin studie et høyere nivå av argumentasjon. Elevene i min studie oppnår derimot sammensatt argument tidligere enn hos Lavy, forskjellen her kan ligge i type tema elevene jobber med. Mitt undervisningsopplegg prøver å lære elevene om geometriske figurer, i Lavy sin studie så er tallteori fokuset. I Lavy sin oppgave skal elevene endre variablene i et ferdig program, i min undervisning velger elevene i tillegg hvilke koder som skal bruke og designer programmet helt fra starten av. Det brukes også forskjellige programmeringsspråk i studiene, der det brukes Logo i Lavy sin studie og Scratch i mitt. I likhet med Lavy sin studie så blir flere av parene i starten opphengt av det å bruke store tall i variablene den visuelle effekten, parene ser i min studie er det at katten går mange ganger rundt i sirkler.

I Kaufmann og Stenseth (2020) viser elevene argumenter basert på matematiske betraktninger, par 1 er etter min mening veldig nære å vise det samme, men kommer ikke helt dit. Resten av parene viser intuitiv og kodefokusert argumentasjon. Kaufmann og Stenseth bruker Processing som er et skriftlig programmeringsspråk og jeg bruker Scratch et blokkbasert programmeringsspråk. Oppgaven i Kaufmann og Stenseth sin studie er å endre variablene i et ferdig skrevet program, imens i mitt undervisnings opplegg må elevene velge koder som skal brukes selv og deretter endre variablene.

5.3 Vurdering av undervisningsopplegget

Målet med mitt undervisningsopplegg var at elevene skulle bli engasjert og møte på problemer som ville få dem til å vise fram løsningsstrategier og argumentasjon. Selv om ikke alle figurer som elevene programmerte var et problem for elevene ser man i resultatene at elevene møtte på utfordringer som kan sies å være et problem. Parene varierte i hvor mye de argumenterte, men alle viste et nivå av argumentasjon. Ut ifra undervisningens hovedmål vil jeg si at det fungerte slik det skulle.

Argumenter basert på matematiske betraktninger er nivået Kaufmann og Stenseth (2020) ønsker at elevene skal oppnå. Dette var også et ønske jeg hadde ved at noen av parene skulle oppnå dette. Det kan godt hende at elevene hadde oppnådd dette dersom de bare hadde fått bedre tid ved å fortsette neste dag, men opplegget for timen kunne kanskje også vært bedre tilrettelagt for å veilede elevene til argumenter basert på matematiske betraktninger. Elevene viser blant annet liten grad av abstraksjon. Det er mulig at et ferdig satt opp program hadde fått elevene til å tenke mer over variablene og mindre over hvilke blokker som skulle brukes. Ved å ha flere variabler satt fra før kan det få elevene til å tenke mer over hva de ulike kodene gjør, og hvilke koder som er avgjørende for å tegne en gitt geometrisk figur. En slik oppgave blir også mer lik oppgavene i Kaufmann & Stenseth (2020) og Lavy (2006) sine oppgaver der elevene viser et høyere argumentasjonsnivå. Programmeringsspråket er også annerledes i forhold til Kaufmann & Stenseth (2020) og Lavy (2006) sine studier, men her er også disse to forskjellige fra hverandre. I tillegg ser jeg ikke noen begrensinger med å bruke Scratch sammenlignet med deres programmeringsspråk. Oppgavene kan også ha ført til argumenter basert på matematiske betraktninger uten å bli endret på, om man hadde oppfordret elevene til å ta seg bedre tid med problemløsningsprosessen kunne dette også hjulpet elevene. Et større fokus fra min side på å få elevene til å bruke lengre tid på å analysere og utforske før implementering, kunne fått dem til å løse problemet mer likt matematikeren i Schoenfeld (1989) sin studie. Siden min plan var å ikke påvirke dem for mye, var dette noe jeg prøvde å holde meg unna i undervisningen, men er noe jeg vil fokusere mer på i mine undervisninger framover som lærer.

Undervisningsopplegget kan ha påvirket hvilke algoritmiske-metoder som ble brukt av elevene. Abstraksjon og modulisering var det ingen av parene som benyttet seg av, par 4 kunne ha brukt denne metoden istedenfor inkrementell og iterativ med å koble sammen flere kvadratprogram til en sammensatt figur. Oppgavene var heller ikke designet med tanke på at elevene skulle sette sammen flere mindre programmer til et stort program, min tanke var at elevene først skulle klare å forstå hvordan løkkefunksjonen fungerer før de skulle sette mange sammen til et stort program.

6 Oppsummering av funn

Målet med studien var å svare på forsking spørsmålet:

Hvilke løsningsstrategier og type argumentasjon bruker elever i programmeringsoppgaver?

Ser vi først på løsningsstrategier er det to metoder som viser seg. Den metoden vi ser mest av er testing og feilsøking, etterfulgt av inkrementell og iterativ metode. Det viser seg en sammenheng her med de algoritmiske metodene til Brennan & Resnick (2012) og problemløsningsprosessene Schoenfeld (1989) viser til. Der elevene bruker lite tid på å lese oppgaven, analysere og utforske, bruker elevene også testing og feilsøking. Det vises seg annerledes når elevene tar seg litt tid til å analysere problemet, da ender de opp med å bruke metoden inkrementell og iterativ. En grunn til dette kan være at når elevene analyserer problemet mer, tar de seg bedre tid med å komme i gang med å løse oppgaven, både par 1 og 4 bryter da problemet ned til mindre del mål som blir enklere å programmere. En annen grunn kan være at det at de tar seg tid til å analysere senker tempoet slik at de ikke stresser med å teste ut et helt program, men tar seg tiden til å jobbe trinnvis med testing underveis. Det brukes heller ikke løkkefunksjonen der elevene benytter seg av inkrementell og iterativ.

Argumentasjonsnivåene er ganske like mellom alle elevene, ser man på Kaufmann & Stenseth (2020) sine argumentasjonsnivåer er alle elevene på intuitiv og kodefokusert argumentasjon. Med Lavy (2006) sitt rammeverk viser elevene grunnleggende argument i starten og utvikler seg til sammensatt argumentasjon. Det skiller mer mellom elvene på hvilke av Bocconi med fler (2016) sine begreper innen algoritmisk tenkning elevene viser. Par 1, 2 og 3 viser automatisering, generalisering, feilretting og algoritmisk behandling. Petter og Gunnar i par 1 viser i tillegg dekomponering, som også par 4 viser. Par 1 som viste flest begreper av algoritmisk tenkning, var også nærmest å oppnå Kaufmann & Stenseth (2020) sitt neste nivå argumentasjon basert på matematiske betraktninger. Hadde par 1 fått litt mer tid har jeg god tro på at de ville vist dette nivået.

6.1 Begrensninger og veien videre

Denne studien viser hvordan fire par elever jobber med programmeringsoppgaver. Studiens empiri er hentet ved bruk av observasjon med observatørrollen fullstendig deltaker og et semistrukturert intervju. Data fra observasjonen ble også brukt til å tilpasse spørsmålene til de

forskjellige parene for å bekrefte mine observasjoner og utdypning av de jeg observerte. Med ettertanke, når jeg ser på observasjons notater og de transkriberte intervjuene, kunne jeg fått hentet ut større mengder data ved å bruke videoopptak av elevene i undervisningen. Videoopptak ville hjulpet meg å få mer nøyaktig og objektiv observasjon, og jeg ville fått lik mengde data på alle parene. Et oppgavebasert intervju kunne også vært til nytte for å kunne stille spørsmål til elevene om tankeprosessene samtidig som de er prosessen. Intervjuene ble gjort dagen etter undervisningen og det kunne virke som noen elever ikke husket alt de gjorde i undervisningen.

Denne studien ser på hvordan elever jobber med programmerings oppgaver i to undervisningsøkter, der den økten er en introduksjon til programmeringsspråket Scratch og, den andre økten jobber de med oppgaver. Det kunne vært interessant å følge elevene over flere økter for å se hvordan utviklingen til elevene beveger seg fremover slik studiene til Lavy (2006) og Kaufmann & Stenseth (2020) gjør. På grunn av at skolen var i en annen kommune og pandemien Covid-19, var dette noe jeg ikke hadde muligheten til ettersom mye reising fram og tilbake ville vært tidkrevende og imot anbefalinger om begrensning av smitte.

Til videre forskning kan det være av nytte å se på hvordan man kan hjelpe elevene med å utvikle abstraksjon med bruk av programmering. Det var liten grad av abstraksjon blant elevene. Jeg ser på det å kunne abstrahere som en viktig ferdighet for å kunne utnytte programmering til å øke den matematiske forståelsen. En mulighet her kan være endre oppgave type til et ferdig program slik at elevene blir tvunget til å bearbeide mer informasjon, og der elevene må velge ut hva som er essensiell informasjon. For å styrke funnene i denne studien kunne man også sett på hvordan andre klasser med lignende utgangspunkt opererer. En kvalitativ forskning som dette vil alltid være vanskelig å rekonstruere til å være helt likt, men ved å prøve undervisningsopplegget på flere klasser kunne man fått et mer generaliserende svar og mulig utviklet undervisningsopplegget.

Referanser

Bjørndal, C. (2015). *Det vurderende øyet.* Oslo: Gyldendal Akademisk.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education.* Sevilla: European Commission, Joint Research Centre.

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design.* Vancouver, BC: American Educational Research Association.

Christoffersen, L., & Johannessen, A. (2012). *Forskningsmetode for lærerutdanningene.* Oslo: Abstrakt forlag.

Clark-Wilson, A., Noss, R., Hoyles, C., Saunders, P., & Benton, L. (2019). Key Factors for Successfully Embedding a Programming Approach to the Primary Maths Curriculum at Scale. *Conference: 11th Congress of the European Society for Research in Mathematics Education.* Utrecht, Netherlands: UCL Institute of Education.

Corbin, J., & Strauss, A. (2015). *Basics of Qualitativ Research Techniques and Procedures for Developing Grounded Theory.* Thousand Oaks, CA: Sage Publications.

Creswell, J., & Poth, C. (2018). *Qualitative Inquiry and Research Design: Choosing Among Five Approaches.* Los Angeles : Sage.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woolard, J. (2015). *Computational thinking A guide for teachers.* Computing at School.

Dohn, N. B. (2020). Students' interest in Scratch coding in lower secondary mathematics. *British Journal of Educational Technology (51)(1)*, ss. 71-83.

Gjøvik, Ø., & Torkildsen, H. A. (2019, 3). Algoritmisk tenkning. *Tangenten - tidsskrift for matematikkundervisning* , ss. 31-37.

Gold, R. (1958). Roles in sociological field observations. *Social Forces, 36*, 217-223.

Haraldsrud, A. D., Sveinsson, H. A., & Løvold, H. H. (2020). *Programering i skolen.* Oslo: Universitetsforlaget.

Kaufmann, O. T., & Stenseth, B. (2020, Mars 25). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology*.

Lavy, I. (2006). A case study of different types of arguments emerging from explorations in an interactive computerized environment. *Journal of Mathematical Behavior (25)*, ss. 153-169.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011, Februar). Computational Thinking for Youth in Practice. *ACM Inroads*, ss. 32-37.

Lesh, R., & Zawojewski, J. (2007). Problem Solving and Modelling. I F. K. Lester, *Second Handbook of Research on Mathematics Teaching and Learning* (ss. 763-804). Greenwich: Information Age Publishing.

Mertens, D. (2015). *Research and Evaluation in Education and Psychology: Integrating diversity with quantitative, qualitative, and mixed methods.* Teller Road: SAGE Publications.

NESH. (2016). *Forskningsetiske retningslinjer for samfunnsvitenskap, humaniora, juss og teologi.* Hentet fra https://www.forskningsetikk.no/retningslinjer/hum-sam/forskningsetiske-retningslinjer-for-samfunnsvitenskap-humaniora-juss-og-teologi/

Opplæringslova. (1998). *Lov om grunnskolen og den vidaregåande opplæringa (LOV-1998-07-17-61)*. Hentet fra https://lovdata.no/lov/1998-07-17-61

Pólya, G. (1957). *How to Solve It.* Garden City, NY: Doubleday.

Polya, G. (1981). *Mathematical discovery: On understanding, learning, and teaching problem solving.* New York: John Wiley & sons, Inc.

Postholm, M., & Jacobsen, D. (2018). *Forskningsmetode for masterstudent i lærerutdanning.* Oslo: Cappelen Damm.

Schoenfeld, A. H. (1989). Teaching mathematical thinking and problem solving. I L. B. Resnick, & L. E. Klopfer, *Toward the Thinking Curriculum: Current Cognitive Research* (ss. 83-103). Association for Supervision and Curriculum Development: Alexandria, VA.

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

Stenseth, B., Kaufmann, O. T., & Forsstøm, S. E. (2019, Februar). Programering og matematikk. *Tangenten*, ss. 7-12.

Store Norske Leksikon. (2018, mai 14). *Store Norske Leksikon*. Hentet fra https://snl.no/argumentere

Torkildsen, S. (2017, November). *Matematikksenteret.* Hentet fra Matematikksenteret: https://www.matematikksenteret.no/sites/default/files/media/filer/MAM/Torkildsen%20A%CC%8A%20undervise%20Matematisk%20Probleml%C3%B8sing.pdf

Utdanningsdirektoratet. (2019, 03 27). *Algoritmisk tenkning.* Hentet fra https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2020). *Læreplan i matematikk 1.–10. trinn (MAT01-05).* Hentet fra https://www.udir.no/lk20/mat01-05

Wing, J. M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, ss. 7-14.

Vedlegg 1

NSD NORSK SENTER FOR FORSKNINGSDATA

NSD sin vurdering

Prosjekttittel

Løsningsstrategier hos elever

Referansenummer

556348

Registrert

15.01.2021 av Kristian Berg - kbe106@post.uit.no

Behandlingsansvarlig institusjon

UiT Norges Arktiske Universitet / Fakultet for humaniora, samfunnsvitenskap og lærerutdanning / Institutt for lærerutdanning og pedagogikk

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Kjersti Wæge, kejrsti.wage@ntnu.no, tlf: 91897622

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Kristian Berg, berg-kris@hotmail.com, tlf: 98898236

Prosjektperiode

01.01.2021 - 15.05.2021

Status

29.01.2021 - Vurdert

Vurdering (1)

29.01.2021 - 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 den 29.01.2021 med vedlegg, samt i meldingsdialogen mellom innmelder og NSD. Behandlingen kan starte.

DEL PROSJEKTET MED PROSJEKTANSVARLIG
Det er obligatorisk for studenter å dele meldeskjemaet med prosjektansvarlig (veileder). Det gjøres ved å trykke på "Del prosjekt" i meldeskjemaet.

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://www.nsd.no/personverntjenester/fylle-ut-meldeskjema-for-personopplysninger/melde-endringer-imeldeskjema
Du må vente på svar fra NSD før endringen gjennomføres.

TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til 15.05.2021.

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 ogsamtykker til behandlingen

- formålsbegrensning (art. 5.1 b), ved at personopplysninger samles inn for spesifikke, uttrykkelig angitte ogberettigede formål, og ikke viderebehandles til nye uforenlige formål
- dataminimering (art. 5.1 c), ved at det kun behandles opplysninger som er adekvate, relevante ognødvendige for formålet med prosjektet
- lagringsbegrensning (art. 5.1 e), ved at personopplysningene ikke lagres lengre enn nødvendig for åoppfylle formålet

DE REGISTRERTES RETTIGHETER

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.

Så lenge de registrerte kan identifiseres i datamaterialet vil de ha følgende rettigheter: innsyn (art. 15), retting (art. 16), sletting (art. 17), begrensning (art. 18) og dataportabilitet (art. 20).

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).

OneDrive er databehandler i prosjektet. NSD legger til grunn at behandlingen oppfyller kravene til bruk av databehandler, jf. art 28 og 29.

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

OPPFØLGING AV PROSJEKTET

NSD vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!

Kontaktperson hos NSD: Silje Fjelberg Opsvik

Tlf. Personverntjenester: 55 58 21 17 (tast 1)

Vedlegg 2

Intervjuguide

Intervjuet er et semistrukturert gruppeintervju. Gruppene består av tre elever som tidligere har jobbet med programmerings oppgaver. Det vil bli gjort en observasjon før gruppeintervjuet som vil ha innvirkning på hvordan spørsmålene blir formulert. Dette blir gjort for å hjelpe elevene med å dele sine erfaringer. Fokusspørsmålene skal hjelpe å holde en struktur under intervjuet.

Tidsramme 30-45min

Fokusspørsmål for intervju

1. Hvordan var timen?
    a. Hva lærte dere?
    b. Hvordan var oppgavene?
2. Hva synes dere om programmering?
3. Gjorde dere noe før dere kjørte kodene i programmet?
    a. Snakket dere om hvordan dere skulle løse oppgavene før dere startet?
    b. Brukte dere noe kladd?
    c. Tenkte dere over hva kodene ville gjøre før dere trykket start?
        i. Hvorfor tenkte dere at kodene ville gjøre dette?
4. Hvordan jobbet dere med oppgavene?
    a. Prøvde dere ut forskjellige løsninger?
    b. Var det noe som var vanskelig og hvordan løste dere dette?
5. Hadde dere løst oppgavene på en annen måte om dere skulle gjort det på nytt?
    a. Hvorfor gjort det slik?

Vedlegg 3

Programmerings opplegg 2 timer 6.klasse

1.time 45min

Læringsmål: klare å bruke grunnleggende funksjoner i Scratch

Elevene får en grunnleggende opplæring i bruk av det blokkbaserte programmet Scratch. De skal i denne timen lære:

- Hva som er blokkbasert programmering.
- Hvordan programmet leser blokkene.
- Hvordan man beveger figuren og tegner med den.
- Hvordan man lager løkker og meldinger.

Jeg viser hvordan man bruker scratch på smartboard (eller lignende) samtidig som elevene gjør det på pc (https://scratch.mit.edu/projects/editor/?tutorial=getStarted) gjerne sammen i grupper på 2-3 elever. Etter gjennomgangen skal elevene tegne valgfrie mønster eller figurer ved å bruke det de har lært. Det blir en oppsummering der vi går igjennom hva de har lært.

2.time 90min

Læringsmål: utforske egenskaper til geometriske figurer med scratch.

Elevene jobber sammen i grupper på 2-3.

- Først blir det en muntlig oppsummering av hva vi gjorde forrige time.
- Elevene skal lage tre forskjellige geometriske figurer på Scratch, for eksempel kvadrat, rektangel og likebeint trekant.
- De som blir ferdige kan prøve å tegne en blomst, et hus eller noe annet.

Her vil elevene få mye frihet i å velge hva slags figur de vil tegne. For å klare å tegne geometriske figurer på Scratch må elevene finne ut av egenskapene til figurene. Under tegne prosessen vil de da utforske egenskapene til figurene. Om elevene bruker løkker og

funksjoner vil de bli enklere for elevene å holde orden i kode programmet og de kan lettere feilsøke. Til slutt tar vi en oppsummering sammen i klassen.

Vedlegg 4

Samtykkeerklæring om bruk av datamateriale i forbindelse med masteroppgave

Bakgrunn og formål

Denne undersøkelsen er en del av min mastergradsoppgave for lærerutdanningen 5.-10.klasse ved Universitetet i Tromsø. Formålet med undersøkelsen er å se på hvordan elever løser oppgaver når de bruker et programmeringsprogram i matematikk. Programmering i skolen og matematikk er ganske nytt og det er lite norsk forskning på dette området, denne oppgaven vil være et bidrag til å forbedre undervisningen med programmering i matematikk.

Hva innebærer deltakelsen i studiet?

For å samle inn data om hvordan elever løser programmeringsoppgaver vil elevene bli observert når de jobber sammen i grupper med oppgavene. Etter undervisningen vil noen grupper være med i et intervju på 30-45 minutter.

Tematikken under intervjuet er hvordan elevene jobbet med programmeringsoppgavene i timen. Spørsmålene angår ikke private forhold (f. eks familiehistorie) eller andre spørsmål som vil identifisere elevene.

Hva skjer med informasjonen som samles inn?

Alle personopplysninger vil bli behandlet konfidensielt og anonymiseres. Jeg gjør notater under observasjonen som vil bli transkribert på en beskyttet nettsky og intervjuet blir tatt opp på en digital lydfil, transkribert og lagret på nettskyen. Ingen andre enn meg vil ha tilgang til lydfilen og det er det transkriberte intervjuet som skal brukes i oppgaven.

Prosjektet avsluttes mai 2021 og all informasjon som er innhentet vil bli slettet når oppgaven er levert. Det vil aldri være mulig å identifisere enkeltpersoner i den publiserte oppgaven.

Dine rettigheter

Så lenge du (eleven) kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke personopplysninger som er registrert om deg,

- å få rettet personopplysninger om deg,
- få slettet personopplysninger om deg,
- få utlevert en kopi av dine personopplysninger (dataportabilitet), og
- å sende klage til personvernombudet eller Datatilsynet om behandlingen av dine personopplysninger.

Hva gir meg rett til å behandle personopplysninger om eleven?

Jeg behandler opplysninger om eleven basert på foresattes samtykke

Frivillig deltakelse

Det er frivillig å delta i studien, og eleven(e) kan når som helst trekke seg underveis uten å oppgi grunn.

Eventuelle spørsmål kan rettes til meg Kristian Berg på kbe106@uit.no

Kristian Berg

Universitetet i Tromsø

Eleven(e)s navn:

Foresatte

☐ **Ja,** jeg/vi samtykker i at observasjon og intervju av barnet vårt i forbindelse med dette prosjektet kan brukes slik skissert over.

☐ **Nei,** jeg/vi samtykker IKKE i at observasjon og intervju av barnet vårt i forbindelse med dette prosjektet kan brukes slik skissert over.

Dato:_____ Sted:_____

Signatur foresatte :_____

Elev

☐ **Ja,** jeg samtykker i at observasjon og intervju av meg i forbindelse med dette prosjektet kan brukes slik skissert over.

☐ **Nei,** jeg samtykker IKKE i at observasjon og intervju av meg i forbindelse med dette prosjektet kan brukes slik skissert over.

Dato:_____ Sted:_____

Signatur elev :_____