UiA Universitetet i Agder

Master

Programmering i matematikk – muligheter og utfordringer

En studie rundt innføringen av programmering som del av matematikkfaget i den norske skolen og hvilke argumenter som taler for eller imot innføringen.

RENATE ELISE JAKOBSEN

VEILEDER
Niclas Larson

Universitetet i Agder, 2019
Fakultet for teknologi og realfag
Institutt for matematiske fag

Forord

At jeg nå kan sitte og skrive forordet som markerer slutten på mastergradsarbeidet mitt var ikke en selvfølge da jeg for 3,5 år siden startet på påbyggingsstudiet i master i matematikk ved Universitetet i Agder. Reisen har vært lang og tøff, men samtidig utviklende og svært lærerik. Jeg er utrolig stolt over å ha klart å fullføre dette studiet, i noen perioder med full jobb ved siden av. Det er mange som har bidratt til at denne reisen har vært mulig, og som jeg ønsker å takke.

Først og fremst vil jeg takke mine informanter som brukte av sin tid og stilte opp til intervju for å dele av sine tanker, meninger og erfaringer om programmering. Uten dem ville aldri denne oppgaven sett dagens lys, så tusen takk til Cathrine Wahlstrøm Tellefsen, Andreas Drolsum Haraldsrud, Tom Louis Lindstrøm, Tor Espen Kristensen og Børre Stenseth.

Jeg ønsker videre å rette en stor takk til min veileder Niclas Larson som har fulgt meg gjennom hele prosessen med masteroppgaven. Selv om vi aldri har møtt hverandre personlig, har dine konstruktive innspill, dine detaljerte kommentarer og tilbakemeldinger vært til uvurderlig hjelp i arbeidet med denne oppgaven.

At det ble mulig for meg som er bosatt i Finnmark å få studere mastergrad i Kristiansand, kan jeg takke rådgiver Trine Engeland og alle lærerne ved masterstudiet ved Universitetet i Agder for. De har vært svært fleksible og flinke til å tilrettelegge for at jeg som langveisfarende student skulle kunne gjennomføre dette studiet. Det har også min egen arbeidsgiver vært, og jeg ønsker også å rette en takk min personalsjef Birgit Simensen som har støttet meg gjennom hele denne reisen.

Takk til min samboer Thomas som har oppmuntret meg og holdt ut med meg disse årene. Du har vært min private kokk og har til tider servert både middag og kvelds til meg mens jeg satt konsentrert foran PC-skjermen. Jeg vil også takke mine to flotte barn som har vært tålmodige med meg, selv om det nok har vært vanskelig å forstå hvorfor jeg så ofte sitter ved PC-en og jobber, istedenfor å tilbringe tiden med dem.

Jeg vil takke Guri og Elin på kontoret mitt som har pushet og støttet meg hele tiden. Dere var aldri i tvil om at skulle jeg klare dette, og dere har vært til uvurderlig hjelp i form av gode matematiske diskusjoner underveis.

Videre vil jeg takke min medstudent Natalie som jeg var så heldig å bli kjent med på mine turer til Kristiansand. Du og jeg har delt oppturer og nedturer, frustrasjon og glede da vi har vært i «samme båt» med både jobb og barn, i tillegg til masterstudiet. Turene til Kristiansand ville ikke blitt det samme uten deg og snart kan vi feire at vi begge er ferdige!

Til slutt vil jeg takke øvrige familie, venner og kollegaer for å ha tenkt på meg underveis og vist interesse for mitt arbeid. Det har vært en stor motivasjon for meg.

Alta, 2019

Renate Elise Jakobsen

Sammendrag

I forbindelse med fagfornyelsen og utarbeidelsen av kjerneelementer og de nye læreplanene, har det vært muligheter til å gi høringsinnspill på utkastene både til kjerneelementene og selve læreplanen. Etter at det første utkastet til kjerneelementer i matematikk ble sendt på høring, kom det frem at programmering nå skulle bli en del av matematikkfaget gjennom arbeid med algoritmisk tenkning. Dette skapte stor debatt, og oppsummeringen av høringsinnspillene viste at det var stor motstand mot dette i mange av de matematiske fagmiljøene i Norge. Norge er likevel sent ute med å innføre programmering i forhold til mange andre land i Europa, og både Sverige og Finland har allerede innført programmering i sine læreplanene. Målet med denne studien var å undersøke hvilke muligheter og utfordringer innføringen av programmering i matematikkfaget kan gi, og hvilke forutsetninger som er viktige for at implementeringen av programmering skal lykkes. På bakgrunn av dette ble forskningsspørsmålene til denne oppgaven:

1. *Hvilke argumenter kan brukes for eller imot innføringen av programmering i matematikkfaget?*

2. *Hvilke forutsetninger er viktige for å lykkes med implementeringen av programmering i matematikk?*

Jeg har brukt generisk kvalitativ metode for å besvare forskningsspørsmålene mine. Fem informanter med erfaring innen matematikk og programmering eller var med i utarbeidelsen av kjerneelementene og læreplan ble intervjuet. Intervjuene ble gjennomført som dybdeintervjuer der et dokumentstudie av høringsinnspill, stortingsmeldinger, diverse utredninger og forskning dannet grunnlaget for utarbeidelsen av intervjuguiden. For å analysere intervjuene og høringsinnspillene ble dataene transkribert og ved bruk av programmet Nvivo ble de kodet og kategorisert ved hjelp av tematisk analyse. Funnene inneholder relevante utsagn og sitater som belyser forskningsspørsmålene.

Studien viser at programmering kan gi lærere og elever mange nye muligheter i matematikk-faget. Der elevene i dag bruker ferdiglagde programmer, kan de nå programmere noen av disse programmene selv, noe som kan bidra til større dybdelæring. Programmering krever algoritmisk tenkning, problemløsningskompetanse og samarbeid, og disse kompetansene ses på som viktige kompetanser for fremtiden. Programmering vil kunne føre til stofftrengsel i matematikkfaget og det kommer til å kreve tid å lære seg verktøyet, både for elever og lærere. Aller helst burde programmering kommet som eget fag, noe både informantene og utredninger påpeker. Mange elever på ungdomsskolen og i den videregående skole vil nå få opptil fire digitale verktøy å forholde seg til i matematikkfaget, noe som kan bli utfordrende. Det trekkes likevel frem at programmering kan være engasjerende, og på denne måten bidra til økt motivasjon og utholdenhet hos elevene.

For å lykkes med implementeringen er det lærerens faglige og didaktiske kompetanse som er avgjørende, dette ble understreket av alle informantene. Lærerkompetansen innenfor programmering er svært lav, da dette ikke har vært en del av lærerutdanningen. For å øke denne kompetansen kreves en massiv etterutdanning. Det er viktig at lærerne ser nytteverdien av programmering i matematikkfaget, og at bruk av dette verktøyet kan bidra til å øke elevenes

faglige utbytte. Eksamen ses på som en viktig faktor som påvirker om lærere vil bruke tid til å arbeide med programmering i matematikktimene. Dette kalles washbackeffekt, og erfaring med innføringen av verktøyet Geogebra viste at eksamen hadde en avgjørende betydning for i hvilken grad verktøyet ble brukt i undervisningen. Det er svært viktig at skoleledelse og utdanningsdirektoratet nå setter av tid og midler til at lærere får mulighet til å både lære seg verktøyet, slik at lærerne og skolene lykkes med å oppfylle kravene til de nye læreplanene og kjerneelementene som nå er fastsatt.

Abstract

Elaboration and renewal of the Norwegian curricula have given opportunities to provide feedback through consultation inputs on the drafts for both the core elements and the curriculum itself. After the first draft of the core elements of mathematics was submitted for consultation, it was stated that programming would become part of the mathematics subject. This created a lot of debate, and the summary of the consultation input showed that there was great opposition to this in many of the academic communities in Norway. However, Norway is late in introducing programming in the curriculum compared to many other countries in Europe, e.g. both Sweden and Finland have already introduced programming in their curriculum. The aim of this study was to investigate what opportunities and challenges the introduction of programming in mathematics can provide, and what preconditions should form the basis for the success of programming. With this background, the research questions for this thesis were:

1. *What arguments can be used for or against the introduction of programming in the mathematics subject?*

2. *What is necessary to succeed with the implementation of programming in mathematics?*

I have used a generic qualitative method, to prosess my research questions. I have undertaken a documentary study where consultation input, parliamentary reports, various studies and research were studied. The document study led to the designing of the interview guide, and five informants who are experienced in mathematics and programming and/or were involved in the preparation of the core elements and the design of the curriculum, were interviewed. The interviews were conducted as in-depth interviews. To analyze the interviews, the data were transcribed, and by use of Nvivo program, were coded and categorised using thematic analysis. The findings contain relevant statements and quotations that pertinent to the research questions.

The study shows that programming can give teachers and students many new opportunities in the mathematics field. Students often use pre-made programs, but now they can build some of these programs themselves, such as derivation which can contribute to a greater depth learning. Programming requires algorithmic thinking, problem solving skills and collaboration, which are considered as important competencies for the future. Programming will nevertheless lead to content overload in the mathematics profession and it will require time to develop the tool, both for students and for teachers. Most preferably, programming should come as a separate subject, something both informants and investigators point out. Many students in lower and upper secondary school will now have up to three digital tools to deal with in the mathematic class which can be challenging. Nevertheless, it was emphasised that programming can be engaging, and in this way contribute to increased motivation and perseverance for the students.

In order to succeed with the implementation of programming, it was emphasised by all informants that it is the teacher's professional and didactic competence that is crucial. Teacher competence in programming is very low, as this has not been part of teacher education. To increase this competence, a massive coursing and education is required. It is important that teachers see the value of using programming in the mathematics subject and that using this tool can help to increase students' academic performance. The written exam is also an important

factor that influences how much time teachers will spend on working on programming in math lessons, this is called the washback effect. Experience with the introduction of the tool Geogebra showed that the exam was crucial to whether the teacher implemented Geogebra the classroom or not. The fact that school management and the Directorate of Education now devote time and resources to allow teachers to learn the tool is very important, so that teachers and schools succeed in meeting the requirements of the new curriculum and the core elements that are now set.

Innhold

Figurliste:

Tabelliste:

1 Innledning

Samfunnet rundt oss er i stadig endring og det er viktig at både skolen og lærere holder tritt med utviklingen. Hverdagen blir stadig mer preget av ulik bruk av teknologi, og barn og ungdom som vokser opp i dag er allerede svært aktive brukere av digital teknologi. Men selv om norske barn og ungdom er storforbrukere av digitale medier som snap, instagram, facebook, etc., betyr ikke det at de nødvendigvis har gode grunnleggende digitale ferdigheter. Gjennom innføringen av Læreplan for Kunnskapsløftet (LK06) i 2006, ble fem grunnleggende ferdigheter innført. Digitale ferdigheter, sammen med lese-, skrive-, regne-, og muntlige ferdigheter er ferdigheter som barn og unge skal utvikle gjennom grunnskolen og videregående skole. Dette er ferdigheter som ses på som forutsetninger for læring og utvikling i skole, arbeid og samfunnsliv (Kunnskapsdepartementet, 2006). Rapporten fra ICILS[1] viste at elevene i Norge scorer bedre enn mange andre land innenfor digitale ferdigheter, men likevel har fortsatt mange elever for svake digitale ferdigheter (Ottestad, Throndsen, Hatlevik & Rothagi, 2014).

De siste årene har det kommet kritikk at elevene i dagens skole først og fremst læres opp til å bli forbrukere av digitalt innhold og digitale løsninger og ikke produsenter. I boka *Program and be programmed* argumenterer Rushkoff (2010) at det er urovekkende at så få egentlig forstår teknologien som finnes overalt. Sanneutvalgets rapport, *Teknologi og programmering for alle,* understreker at alle elever trenger en utdanning som gjør dem i stand til å forholde seg til teknologi i hverdagen. Elevene må kunne ta stilling til dilemmaer som den teknologiske utviklingen genererer, og forstå hvordan teknologi kan fremme eller hemme en bærekraftig utvikling. De skal ikke bare fungere i samfunnet, men også sette preg på, utvikle og gi retning til framtidens samfunn. Utvalget anbefalte å innføre teknologi som et eget obligatorisk fag i grunnskolen. Et slikt nytt teknologifag skulle inkludere programmering og digital teknologi som en betydelig komponent (Sanne m.fl., 2016).

1.1 Bakgrunn for studien

Gjennom fagfornyelsen skal grunnskolen og den videregående skole fra høsten 2020 ta i bruk nye læreplaner. Utdanningsdirektoratet har tre hovedargumenter for behovet for en fagfornyelse i den norske skolen:

1. Det elevene og lærlingene lærer skal være relevant. Samfunnet og arbeidslivet endrer seg med ny teknologi, ny kunnskap og nye utfordringer. Vi trenger barn og unge som reflekterer, er kritiske, utforskende og kreative.
2. Elevene og lærlingene skal få mer tid til dybdelæring. Mange av læreplanene har vært for omfattende. For å legge gode rammer for dybdelæring, kan vi ikke bare fylle på med nytt innhold. Vi må gjøre tydelige prioriteringer.
3. Det skal bli bedre sammenheng i og mellom fagene og de forskjellige delene av læreplanverket skal henge bedre sammen (Utdanningsdirektoratet, 2018a, s. 1-2)

Et viktig prinsipp for læreplanene blir derfor å gi elevene tid til å gå i dybden i fagene, se sammenhenger mellom fagområder og utvikle evnen til å reflektere og tenke kritisk. Det siste året har over 100 lærere og andre fagfolk jobbet sammen med Utdanningsdirektoratet for å komme frem til kjerneelementene. Det har vært tre brede innspillsrunder hvor det kom inn til sammen 6 700 innspill som direktoratet har lyttet til (Kunnskapsdepartementet, 2018b). Neste

[1] ICILS (International Computer and Information Literacy Study) er en studie av ungdomskolelevers digitale ferdigheter. I alt har 18 land og to provinser i Canada deltatt I ICILS 2013.

steg i prosessen er utarbeidelsen av læreplaner som fastsettes høsten 2019 og høsten 2020 skal planene iverksettes på Vg1 i den videregående skole. Når det gjelder førsteutkastet til kjerneelementene i matematikkfaget kom det inn nesten 100 høringssvar fra mange ulike instanser, blant annet skoler, lærere, privatpersoner, universiteter, høyskoler og organisasjoner. Det temaet som opptok flest respondenter var innføring av programmering som en del av matematikkfaget, istedenfor som eget fag, slik Sanneutvalget anbefalte (Sanne m.fl., 2016). Selv om responsen ikke var entydig, var det gjennomgående sterk motstand mot innføringen. Mange understreket at de er for innføring av et eget programmeringsfag, gjerne kombinert med teknologi, men at det ikke er rom for programmering i matematikkfaget innenfor dagens rammer. Mange av respondentene var bekymret for at innføringen ville kunne føre til ytterligere stofftrengsel i et fag som allerede er for stort, og dermed motvirke, istedenfor å stimulere til dybdelæring. Lav lærerkompetanse innen emnet var også noe som ble påpekt (Kjerneelementgruppa for matematikk, 2017b).

1.2 Forskningsspørsmål

Det er klart at innføringen av programmering som del av matematikkfaget vil få konsekvenser for min og mine kollegaers arbeidshverdag. Jeg jobber i den videregående skole og vår matematikkseksjon var en av skolene som ga høringsinnspill på kjerneelementene. Flere av lærerne i fagseksjonen, inkludert meg, var skeptiske til innføring av programmering som del av matematikkfaget. Få lærere som underviser i matematikk har erfaring med programmering, eller har det som en del av utdanningen sin. Noen av de spørsmålene som kom opp når dette ble diskutert i matematikkseksjonen var: Hva er begrunnelsene for å innføre programmering i matematikkfaget, når motstanden blant lærerne og andre fagpersoner i utgangspunktet er så stor? Hvilke konsekvenser får dette for matematikklærere? Hvordan skal vi lykkes med dette?

I starten av arbeidet med denne oppgaven ønsket jeg derfor å forsøke å danne meg et bilde av hva som var bakgrunnen til at programmering endte opp i matematikkfaget, istedenfor som eget fag som Sanneutvalget anbefalte. Videre ønsket jeg å finne argumenter for eller imot programmering, da hovedsakelig argumenter med faglige begrunnelser rettet mot matematikkfaget i skolen. Det vil også være naturlig å se på hvilke forutsetninger som bør ligge til grunn for at vi som faglærere skal lykkes med å implementere et helt nytt og ganske stort emne i matematikkfaget, i tillegg til det som allerede er i læreplanen. Jeg ønsket å finne ut hvilke muligheter og utfordringer matematikklærere i skolen vil kunne møte når vi skal starte opp arbeidet med programmering med elevene fra høsten 2020. På bakgrunn av dette ble forskningsspørsmålene for denne studien som følger:

1. *Hvilke argumenter kan brukes for eller imot innføringen av programmering i matematikkfaget?*

2. *Hvilke forutsetninger er viktige for å lykkes med implementeringen av programmering i matematikk?*

For å besvare forskningsspørsmålene satte jeg meg først inn i prosessen frem mot innføringen av programmering. Gjennom et dokumentstudie gjennomgikk jeg stortingsmeldinger, rapporter fra ulike utvalg og fant frem til avisartikler skrevet av fagpersoner knyttet til matematiske fagmiljø i Norge som omhandler temaet programmering. I tillegg har jeg gått gjennom mange av innspillene i høringsprosessen i utarbeidelsen av kjerneelementene i faget og læreplanen. Som

informanter etablerte jeg kontakt med fem personer med matematikkfaglig eller programmeringsfaglig bakgrunn for å få frem synspunkter omkring innføringen av programmering i matematikkfaget. Disse er: Cathrine Wahlstrøm Tellefsen, Andreas Drolsum Haraldsrud, Tom Louis Lindstrøm, Tor Espen Kristensen og Børre Stenseth. En bredere presentasjon av informantene kommer i metodedelen.

Gjennom denne oppgaven ønsker jeg hovedsakelig å finne de faglige argumentene som brukes i debatten, og undersøke de begrunnelsene som ligger bak argumentene for eller imot innføringen av programmering. Med *faglige* argumenter menes her argumenter eller påstander som hevder at programmering enten kan bidra til endring i matematikkompetansen til elevene, eller argumenter som hevder at programmering gir nye muligheter eller utfordringer for læreren.

Jeg håper denne oppgaven kan bli et bidrag til å kunne se hvilke muligheter og utfordringer vi som matematikklærere står ovenfor når programmering skal implementeres i undervisningen fra høsten 2020.

1.3 Oppbygging av oppgaven

Denne masteroppgaven inneholder følgende kapitler: Programmering – et tilbakeblikk, teori, metode, funn, diskusjon og avslutning. Tatt i betraktning at det på 1980-tallet var stor tro på programmering i skolesammenheng starter jeg oppgaven med et historisk tilbakeblikk på programmeringen i skolesammenheng, før jeg presenterer relevant litteratur i teorikapitlet.

I metodekapitlet vil jeg presentere og begrunne hvilke metodiske som ble gjort i forbindelse med forskningsdesign, forskningsmetoder og analysemetoder, samt diskutere metodens validitet og relabilitet. Videre presenteres funn gjort i forbindelse med dokumentstudie og intervju, før funnene analyseres, diskuteres og knyttes opp mot relevant teori.

Avslutningen oppsummerer og besvarer forskningsspørsmålene og inneholder også tanker og forslag til videre forskning innenfor dette spennende området.

2 Programmering – et tilbakeblikk

Programmering er ikke noe nytt i skolesammenheng, og i den forbindelse vil jeg gi leseren et innblikk i programmeringens utvikling gjennom de siste 50 årene. Dette er viktig for å danne et bilde av hvorfor dette er så omdiskutert som det er i dag. Jeg vil også gjøre rede for hva man legger i det å programmere, samt se på hvordan andre land har løst utfordringen med å innføre programmering i læreplanene. Jeg vil starte med en kort innføring i datamaskinens historie, for så å definere begrepet algoritmisk tenkning, da dette begrepet er meget sentralt i det å forstå hva programmeringsprosessen går ut på. Jeg vil også forsøke å definere skillet mellom programmering og koding, da dette er to begreper som kan forveksles med hverandre.

2.1 Fra kuleramme til datamaskin

I dag kan en datamaskin kjøre milliarder av beregninger hvert sekund, og det er ikke tvil om at datamaskinens inntog innen matematikk og naturvitenskap har åpnet opp for å kunne løse problemer man for få tiår siden ikke trodde var mulig å løse. Hjelpemidler for å gjøre matematikkberegninger er ikke noe nytt fenomen. Kulerammen er et av de første kjente regneverktøyene og er omtalt i historiske dokumenter allerede for over 2 000 år siden[2]. John Napier er kjent for bruken av logaritmer for å gjøre beregninger med multiplikasjon enklere, da addisjon av eksponenter er enklere enn multiplikasjon av tall. I 1630 ble den første regnestaven basert på denne teknikken laget. Regnestaven ble videreutviklet i 1850 og ble brukt helt frem til 1960-tallet (Rossing, Asphjell & Aas, 2001). William Schickard var en tysk professor og er kjent for å ha designet en av de første maskinene som kunne gjøre beregninger med alle fire regne-artene ved hjelp av tannhjul og det som kalles et Napiers bone. Dette var et manuelt hjelpemiddel for multiplikasjon av tall, ved hjelp av addisjon, se figur 1.

Figur 1: Kopi av Schickards kalkulator. Hentet fra: https://www.computerhistory.org/revolution/calculators/1/47/194

For å kunne kalles en datamaskin må enheten kunne programmeres (Haraldsrud, 2018). Charles Babbage kalles «datamaskinens far» og han designet den første programmerbare enheten kalt *Analytical Engine* allerede i 1837. Babbages konstruksjon var dampdrevet og skulle lese data og instruksjoner fra hullkort, men den ble dessverre aldri ferdigstilt (Rossen & Dvergsdal, 2019). Det skilles ofte mellom analoge og digitale datamaskiner. En analog maskin baserer seg på endringer i fysiske fenomener som strøm, eller mekaniske deler, men ulempen var at denne typen datamaskiner var lite fleksible (Haraldsrud, 2018). Claude Shannon beskrev i sin mastergrad i 1938 hvordan man kunne erstatte mekanikken i datamaskinen med elektronisk styrte strømbrytere som kunne kobles sammen til logiske kretser. Ved hjelp av boolsk algebra

[2] https://matematikk.net/side/Abakus

kunne man nå konstruere datamaskiner som brukte binære tall for å utføre matematiske operasjoner (Rossen & Dvergsdal, 2019). Datamaskinene utviklet seg etter dette raskt og under 2. verdenskrig fikk datamaskinene stor betydning. Alan Turings *The Bombe*, var en dekodingsmaskin som ble brukt for å lese kodene tyskernes kodemaskin *Enigma* sendte ut. På denne måten klarte engelskmennene å knekke kodene til tyskerne i rekordfart, noe som var med på å bidra til slutten av 2. verdenskrig (Holvik, 2014). Etter hvert ble datamaskinene stadig mer digitale, men de tok svært stor plass. På 1960-tallet ble transistorer, som var mye mer effektive enn radiorør, forminsket og satt sammen i integrerte kretser, som etter hvert ble miniatyrisert til microchiper. Microchipene er hjernen i datamaskinene, som også kalles prosessoren. I dag kan integrerte kretser inneholde flere millioner transistorer, og dagens datamaskiner tar lite plass og har en enorm beregningsevne i forhold til før (Haraldsrud, 2018).

I tritt med datamaskinenes utvikling, ble det også utviklet ulike programmeringsspråk som gjør oss i stand til å programmere datamaskiner. I dag er Python blitt et populært språk i undervisningssammenheng. Dette er et tekstbasert programmeringsspråk som er fleksibelt, gratis og lett å lære.

```
from random import randint

tall1 = randint(2, 12)
tall2 = randint(2, 12)

print('Hva er ' + str(tall1) + ' ganger ' + str(tall2) + '?')
svar = input()

if svar == tall1 * tall2:
    print('Ja, svaret er ' + svar)
else:
    print('Nei, det riktige svaret er ' + str(tall1 * tall2))
```

Figur 2: Eksempel på programmeringskode med bruk av Python. Hentet fra kodeklubben.no

Med fokuset sentrert rundt programmering er det også blitt utviklet ulike blokkprogrammeringsspråk, slik at barn og unge lettere kan lære seg programmering. Scratch og Lego Mindstorm er eksempler på populære programmeringsspråk som bruker blokker, i stedet for å skrive tekstkoder.

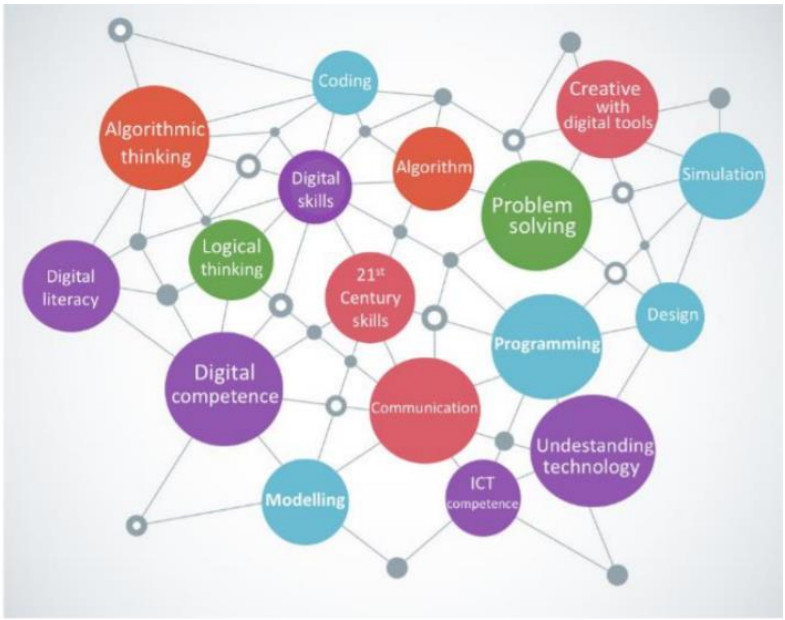
Figur 3: Eksempel på programmering i scratch. Hentet fra kodekubben.no

2.2 Algoritmisk tenkning

Algoritmisk tenkning finner vi som en del av kjerneelementet «utforskning og problemløsning» og spesifikt står det at: «Algoritmisk tenking er viktig i prosessen med å utvikle strategier og fremgangsmåter» (Kunnskapsdepartementet, 2018b, s. 15). Kjerneelementgruppen i matematikk skriver i artikkelen *Kjerneelementer i matematikk, men hvorfor programmering?* at: «For å gi liv til algoritmisk tenkning er det derfor viktig at det kombineres med læring av programmering. Når vi har skrevet inn algoritmisk tenkning som et kjerneelement betyr det derfor også at matematikkfaget skal gi grunnleggende programmeringsopplæring allerede fra småtrinnet» (Kjerneelementgruppa for matematikk, 2017b).

Det er viktig å skille mellom begrepene algoritme og algoritmisk tenkning, da det siste begrepet er en kompetanse som inneholder mye mer enn bare algoritmer. Ordet «algoritme» kommer fra den persiske matematikeren Al-Khwārizmī, og er oversettelsen av navnet hans fra det arabiske alfabetet til det latinske alfabetet (Tesch, 2019). I matematikk og i datasammenheng er algoritme en fullstendig og nøyaktig beskrivelse av fremgangsmåten for løsning av en beregningsoppgave og innebærer å kunne bryte ned et problem i delproblem som kan løses systematisk. Algoritmen angir de enkelte skrittene i oppgaveløsningen og rekkefølgen av dem ved ord, matematisk symbolikk og/eller skjematisk fremstilling av arbeidsgangen (Dahlum, 2018). I et datamaskinprogram er en algoritme uttrykt i et programmeringsspråk det som får datamaskinen til å utføre det vi ønsker. I dagligtalen brukes ofte algoritme synonymt med en oppskrift. For å bruke en algoritme trenger elevene kun evnen til å følge oppskriften stegvis, og de trenger ikke å forstå det logiske fundamentet bak oppskriften (Biggs, 1990).

Algoritmisk tenking (AT) er den norske oversettelsen av det engelske begrepet «computational thinking» (CT). Mange mener det norske begrepet ikke er like dekkende som det engelske og i Sverige brukes oversettelsen «datalogisk tänkande», som kan sies å være en litt mer presis oversettelse enn den norske. I begge begrepene blir bl.a. kompetanse som har opprinnelse fra programmering, sett i en bredere sammenheng (Sanne m.fl., 2016). Algoritmisk tenkning er utviklet som ett nytt konsept for å forbedrede barn og ungdom for fremtidens utfordringer i et stadig mer digitalt samfunn. Rapporten *The Nordic approach to introducing computational thinking and programming in compulsory education*, ser nærmere på utdanningspolitiske dokumenter om algoritmisk tenkning og programmering i de nordiske landene. Denne rapporten knytter kjernekonsepter som abstraksjon, algoritmisk tenkning, automatisering, dekomponering og generalisering til CT-begrepet, og begrunner at disse konseptene er relatert til et sett av holdninger og ferdigheter som også kan knyttes til CT, se figur 4 (Bocconi, Chioccariello & Earp, 2018).

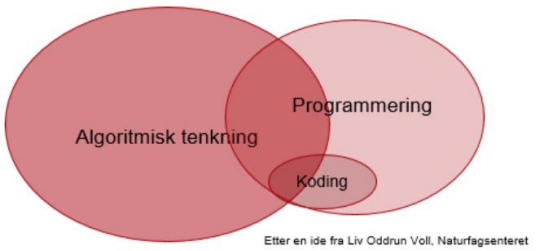


Figur 4: CT-relaterte termer brukt i utdanningspolitisk dokumentasjon i de nordiske landene. (Bocconi m.fl., 2018)

Hovedforståelsen av CT kan sies å være todelt:

1. Å løse problemer og åpne oppgaver
2. Digital kompetanse (som å skape digitale løsninger og være en kritisk bruker)

Ifølge Bocconi m.fl. (2018) er begrepene knyttet til algoritmisk tenkning relatert til sett av holdninger og ferdigheter som inkluderer det å skape noe med datamaskiner (programmer, spill etc.), testing og feilsøking/debugging, samarbeid og kreativitet og evnen til å håndtere åpne oppgaver og problemer. Algoritmisk tenkning utgjør kjernen i naturvitenskapelig praksis og i matematikk, og selv om begrepet er relativt nytt så har skolen lenge arbeidet med elevers evne til algoritmisk tenkning, selv om delen med digital kompetanse ikke er har vært like tydelig (Tellefsen, 2018). Figur 5 viser hvordan programmering både innebærer algoritmisk tenkning og koding, men også mye mer.



Figur 5 Sammenhengen mellom AT, programmering og koding. Etter en ide av Liv Oddrun Voll, Naturfagsenteret.

I rapporten fra Sanneutvalget (2016) ser man på algoritmisk tenkning som det å kunne abstrahere, gjennomgå informasjon systematisk, lære å lese og å forstå forskjellige representasjonsformer, modularisere og resonnere i iterative og parallelle strukturer. Utvalget er tydelig på at å lære algoritmisk tenkning er viktig for å kunne møte fremtidens behov for kompetanse (Sanne m.fl., 2016).

2.3 Programmering – mer enn bare koding

For mange er programmering synonymt med det å lage koder for å få datamaskinen, eller andre gjenstander, til å gjøre det man ønsker. Men begrepet programmering inneholder mye mer enn dette. For å kunne diskutere innføringen av programmering i matematikk, og andre fag, er det viktig å danne seg et moderne og detaljert bilde av hva programmering er og hva det innebærer. I rapporten *programmering i skolen* skrevet av Senter for IKT i utdanning beskrives begrepet slik:

> Programmering, slik det er brukt i dette dokumentet, omfatter mer enn å bare skrive programkode som kan kjøres på en datamaskin, det inkluderer også prosessen med å komme fram til denne koden. Det vil si prosessen fra å identifisere et problem og tenke ut mulige løsninger på problemet, til å skrive kode som kan forstås av en datamaskin, og å feilsøke og kontinuerlig forbedre denne koden (Selvik m.fl., 2016, s. 9).

Sanneutvalget beskriver begrepet programmering i samme åndedrag som algoritmisk tekning i sin rapport *Teknologi og programmering for alle*:

> Det å løse et problem gjennom å spesifisere en presis sekvens av kommandoer kalles algoritmisk tenkning eller algoritmisk problemløsing, og en slik presis sekvens av kommandoer kalles en algoritme. En algoritme kan sammenlignes med en matoppskrift, en strikkeoppskrift eller en regneoppskrift i matematikk. Programmering handler om å utforme algoritmer i et programmeringsspråk som en datamaskin kan tolke (Sanne m.fl., 2016, s. 18).

Videre beskriver Sanneutvalget programmering som en ferdighet, en måte å forstå de grunnleggende mekanismene i digital teknologi og en tilegnelse av algoritmisk tekning som har likhetspunkter med matematisk logikk.

Knut Mørken, professor ved Universitetet i Oslo (UiO), er sitert i artikkelen *Programmering er ikke det samme som koding*, og har denne beskrivelsen av hva koding, programmering og algoritmisk tenking er:

> Den største jobben er å finne ut hva du ønsker å gjøre og formulere det på en presis måte. Koding, selve det å taste inn et dataprogram, er bare en liten del av programmeringen. Viktigere er algoritmisk tenkning, måten en må tenke på for å oversette et problem uttrykt i ord til en logisk rekke av operasjoner som kan kodes inn i en datamaskin (Lynnenbakken, 2018).

Kort oppsummert kan det sies at programmering handler om å bryte et gitt problem ned i et sett av kommandoer og så få en datamaskin til å utføre disse kommandoene. Selv om ferdigheter innen koding er en sentral del av selve programmeringsprosessen, krever det å kunne programmere flere ferdigheter enn bare det å skrive koder, og algoritmisk tekning og feilsøking er en sentral del av denne prosessen.

2.4 Programmeringens historie i skolesammenheng

«Venter vi bare ett år med å gjøre noe, vil vi ikke bare være bak mål, vi vil være langt oppe på tribunene»[3].

Sitatet over er hentet fra et oppslag i Norsk Skoleblad fra 1981 i forbindelse med bruk av PC i skolen og debatten om bruk av teknologi i skole synes nesten å være like gammel som bruken selv. Bruk av datamaskiner og informasjonsteknologi er ikke noe nytt, og det har vært brukt mye lengre i undervisningssammenheng enn mange tror. Den aller første undervisningen i elektronisk databehandling (EBD) ble gitt av Herman Ruge ved forsøksgymnasiet allerede i 1968 (Bostrøm, Bø, Langmyhr & Rydland, 2008). Også informasjonsteknologi som eget fag har en lengre historie i videregående skole enn de fleste er klar over, med ulike fagbetegnelser (EDB, databehandling, informasjonsbehandling, informasjonsteknologi). Disse fagene hadde fra begynnelsen hovedsakelig et programmeringsfokus. Et illustrerende eksempel på dette er at læreplanene for data- og informasjonsbehandling på handels- og kontorfag fra 1974, blant annet inneholdt fagene Cobol-programmering, filbehandling (med programmering) og assemblerprogrammering (Bostrøm m.fl., 2008).

Ved innføringen av mønsterplanen M87 ble bruk av datamaskiner i skolen et eget krav. Dette kom som en direkte følge av en stortingsmelding (Stortingsmelding nr. 39 1983-84) som kom som svar på stadig sterkere krav fra næringsliv og politikere om at «skolen ikke måtte bli datatapere». Det ble bevilget 44 millioner til innkjøp av datautstyr, og Jamissen og Nyhus skriver i boka *EDB i grunnskolen* at «EDB maskiner kjøpes inn og tas i bruk i forskjellige, mer eller mindre velbegrunnede måter. Plutselig er det ikke alltid så nøye hva innholdet i undervisningen er, bare de får prøve seg litt» (Jamissen & Nyhus, sitert i Jåtten, 2006, s. 17).

[3] Norsk skoleblad nr. 20, 1981

Inspirert at Jean Piagets teorier om barns utviklingstrinn utviklet professoren Seymour Papert et av verdens mest kjente programmeringsspråk, LOGO, som skulle få barn til å kommunisere med datamaskinen på en naturlig måte. I et intervju med Mark Goldberg i 1991 sier Papert: «The spirit, is to make the computer an expressive medium for a child to use in a natural way. In a way that's like a producer, not a consumer» (Goldberg, 1991). I 1980 gav Papert ut boka *Mindstorms* som av mange ble sett på som et paradigmeskifte innen pedagogikken, og boka fikk stor suksess i pedagogiske miljøer. Der beskrev han at han ville bort fra at barna lærte fra datamaskinen, men at barna i stedet skulle lære opp maskinen. Med denne kunnskapen kunne barna senere være med på endre tankemønstrene sine, som igjen kunne ha positiv påvirkning på andre fagområder. Papert argumenterte for at barn som lærte å programmere utviklet kognitive ferdigheter innenfor planlegging og problemløsning, og utviklet evnen til å reflektere rundt de kognitive prosessene i seg selv (Papert, 1980).

Ethvert nytt tankesett vil etter hvert bli kritisert, og en av Paperts største kritikere var den amerikanske professoren Roy Pea. Han hevdet at overføringsverdien fra barns læring gjennom å programmere i LOGO, og over til andre fag, var svært liten. Dette aspektet var viktig i Paperts teorier, og gjennom flere vitenskapelige undersøkelser viste det seg at overføringsverdien faktisk var vanskelig å påvise (Pea & Kurland, 1984). Siden den pedagogiske revolusjonen som flere hadde sett for seg gjennom programmeringsarbeidet uteble, ble forsøkene med programmering i skolen færre og færre. Da internett kom for fullt som den nye teknologiske «bølgen», forsvant fokuset på programmering i skolen mer eller mindre helt. Dette ble en aktivitet kun for de spesielt interesserte helt frem de siste årene.

2.5 Programmering som skolefag i andre land

I dag er fokuset på programmering større enn noen gang tidligere. Bevegelsen *Lær kidsa koding*, har spilt en sentral rolle i å sette programmering på den utdanningspolitiske agendaen og har bidratt til etablering av en rekke kodeklubber over hele landet. Verktøyene for å programmere er mer tilgjengelige enn noen gang, og ofte gratis. Siden 2014 har Kodetimen (Hour of Code) og Codeweek blitt gjennomført i en rekke land. Etter å ha deltatt på kodetimen i 2014 ble president Obama kalt den første amerikanske presidenten som skrev en linje med kode (Selvik m.fl., 2016). European Schoolnet (EUN) startet i 2014 *The Coding Initiative* på oppdrag fra EU-kommisjonen. Målet er å fremme programmering og informatikk i Europa. EUN publiserte i oktober 2015 rapporten *Computing our future*, som gir en oversikt over status for programmering i skolen i 21 europeiske land, og den viser at allerede da hadde 16 land programmering som en del av pensum (Balanskat & Engelhardt, 2015). England, Finland, Sverige og Estland har alle innført programmering som del av pensum i skolen, men på ulike måter.

England – programmering som del av eget IKT-fag

Fra høsten 2014 innførte England programmering fra første trinn gjennom det nye faget Computing, som erstattet faget ICT (engelsk for IKT). Det nye faget Computing setter konkrete mål for elevenes programmeringskompetanse på de ulike trinnene, fra å forstå hva en algoritme er, til å kunne benytte minst to programmeringsspråk og designe gode løsninger. I forbindelse med det nye faget er det utviklet en rekke ressurser for å støtte lærere i overgangen, og flere gratis nettbaserte kurs (også kalt MOOC[4]) tilbys fra en rekke universiteter (Selvik m.fl., 2016).

[4] MOOC forkortelse for Massive Open Online Courses

Finland – programmering som del av matematikkfaget

Høsten 2016 innførte Finland nye læreplaner der programmering er tatt med i flere fag. I læreplanen for matematikkfaget blir programmering beskrevet som en aktivitet og et mål fra 1. til 9. trinn under hovedområdet «Matematisk tenkning og matematiske metoder». På 7. til 9. trinn skal elevene utvikle sin algoritmiske tenkning og sine programmeringsferdigheter ved å lage enkle programmer. Elever i Finland har 9-årig grunnskole, ikke 10-årig slik som Norge. Elevenes anvendelse av prinsipper for algoritmisk tenkning og programmeringsferdigheter teller ved sluttvurderingen etter ungdomsskolen (Selvik m.fl., 2016).

Sverige – programmering som del av grunnskolens læreplaner.

I Sverige innføres innføres det programmering i hovedsak i fagene matematikk og teknikk, og det gis MOOC-kurs tilpasset lærere som underviser i disse fagene. Det står i den svenske kursplanen for matematikk at elevene etter 9-årig grunnskole skal kunne vite hvordan algoritmer kan skapes, anvendes, testes og forbedres gjennom programmering for matematisk problemløsning. De skal i tillegg kunne programmere i ulike programmeringsmiljøer (Skolverket, 2016).

Estland – teknologi som tverrfaglig emne

I Estland er programmering et tverrfaglig emne i grunnskolen kalt «Teknologi og innovasjon». Dette emnet gir en tverrfaglig tilnærming til IKT og annen teknologi i skolen, og dette krever at lærere bruker teknologi i undervisningen i alle fag. Det stilles ikke krav til hva slags teknologi som skal benyttes, det er opp til lærerne selv å bestemme. Det er dannet en egen stiftelse HITSA (The Information Technology Foundation for Education) som både arbeider med kompetanseheving av lærere og utarbeidelse av ressurser, samt sikrer at de som går ut av den estiske skolen har den digitale kompetansen de trenger (Selvik m.fl., 2016).

Som det fremkommer av gjennomgangen av disse fire landene, kan det se ut som at våre politikere har sett over til våre nordiske naboer når Norge har fastsatt hvordan programmering skal innføres i skolen. Det bør likevel nevnes at innføringen har skapt debatt også i våre naboland. Forskere og lærere ved bl.a. Malmö universitet, som har en av Sveriges største lærerutdanninger, går ut i media og advarer mot at verken skolene eller lærerne er klare for å innføre programmering i skolen enda (Ekelund m.fl., 2018). I Norge er det bestemt at programmering av skal inn som del av pensum i flere fags læreplaner, da med hovedvekt i matematikk som er faget som skal ha ansvaret for opplæringen. De siste årene har flere ungdomsskoler hatt prøveordninger gjennom å ha programmering som valgfag, og i 2016 var det hele 150 skoler som tilbydde dette faget. Det er laget en egen forsøkslæreplan der det vektlegges at elevene gjennom valgfaget skal utvikle sin algoritmiske tankegang.

2.6 Forskning på overføringsverdier fra programmering

Forskningen som fulgte i kjølvannet av LOGO-programmeringen klarte ikke å vise en tydelig overføringsverdi fra kompetanse i programmering til kompetanse på andre områder, og programmering ble derfor en aktivitet for spesielt interesserte i en lang periode. Programmering er igjen «allemannseie», og populariteten skyldes blant annet at det nå på nytt hevdes at kompetanse i dataprogrammering forbedrer kognitive ferdigheter, inkludert kreativitet, resonnering og matematiske ferdigheter (Scherer, Siddiq & Viveros, 2018). Bare de siste årene er det skrevet flere mastergrader som omhandler programmering, og i desember 2018 ble det publisert to større undersøkelser som omhandler programmering.

Den første er en stor undersøkelse som omhandler overføringseffekter og kognitive fordeler gjennom arbeid med programmering. Undersøkelsen er en metaanalyse gjort av 105 ulike studier, og er publisert i det anerkjente tidsskriftet *Journal of Educational Psychology*. Studiene som ble undersøkt hadde alle en sammenligningsgruppe, noe som styrker de statistiske resultatene som ble generert i undersøkelsen. Forskerne delte effektene inn i hvorvidt de har innvirkning på oppgaver som involverer programmering eller ligner programmering (near-transfer) eller om de er på mer generell problemløsning eller faglige ferdigheter (far-transfer). Resultatene er oppsiktsvekkende, og viser tydelig at kompetanse i dataprogrammering har positiv effekt på bl.a. kreativ tenkning, matematiske ferdigheter og metakognisjon (Scherer m.fl., 2018). Det som er verdt å bemerke er hvordan sammenligningsgruppene har prestert når det gjelder «far transfer» oppgavene. Hvis man ser på de 41 studiene der den ene gruppa jobbet med programmeringsaktiviteter og sammenligningsgruppa ikke fikk programmeringskompetanse, men istedenfor arbeidet systematisk med læringsstrategier, problemløsning og PC, så ser man ikke lengre tydelig forskjell på gruppene. Programmeringsgruppa er ikke lengere signifikant forskjellig fra sammenligningsgruppa. Begge gruppene fikk en positiv effekt på bl.a. kreativ tenkning, matematiske ferdigheter og metakognisjon, og det kan likevel utfra denne undersøkelsen konkluderes med at kompetanse i dataprogrammering gir overføringsverdi til kompetanse på andre områder (Scherer m.fl., 2018).

Den andre undersøkelsen er utført av Sanna Erika Forsström og Odd Tore Kaufmann ved Høgskolen i Østfold. De har gjort en litteraturanalyse av 15 artikler for å finne forskningsbaserte begrunnelser for innføringen av programmering i matematikkfaget i andre land. Gjennom denne litteraturanalysen identifiserte de fire temaer som gikk igjen: Motivasjonen til å lære matematikk, studentenes prestasjoner i matematikk, samarbeid mellom elever og den endrede rollen til læreren. De fant i analysen at arbeid med programmering i matematikk både øker elevenes motivasjon til å lære matematikk og forbedrer studentenes prestasjoner i faget (Forsström & Kaufmann, 2018). De konkluderer med at på den ene siden kan analysen av artiklene vise til at programmering gir økt motivasjon til å lære matematikk og forbedrede prestasjoner i matematikk, men på den andre siden er det ikke like tydelig at resultatene kan generaliseres til matematikkfaget i skolen. Dette er et område som de anbefaler at det bør forskes mer på, og det påpekes at de konkrete fordelene med å innføre programmering i matematikkfaget avhenger av mange faktorer som må ses i sammenheng med integreringen. Læreren får en annen rolle i undervisningen enn tidligere, og samarbeid mellom studentene er viktige for læringsprosessen. I tillegg er det viktig å endre undervisningsmetoder, da programmeringsaktiviteter ofte skiller seg fra tradisjonell matematikkundervisning (Forsström & Kaufmann, 2018).

3 Teori

For å få en bedre forståelse av hvorfor innføringen av programmering i matematikk er et så omdiskutert tema, mener jeg det er viktig å sette seg inn i matematikkens egenart og hvorfor det skjer endringer i matematikkopplæringen. I forbindelse med fagfornyelsen ble kompetansebegrepet oppdatert og dybdelæring er blitt et viktig begrep. Dette kapitlet vil belyse hvilke kompetanser som anses som viktige for fremtiden, og da spesielt innenfor matematikkfaget, og se på hva som menes med dybdelæring i matematikkfaget.

3.1 Matematikkens egenart

Mange har stilt spørsmålet: «Hva er matematikk?», og det finnes mange ulike oppfatninger av hva matematikk egentlig innebærer. For noen er matematikk et positivt ladd ord, men dessverre gir dette ordet for mange der ute en forbindelse til lav mestring og vonde følelser. Ordet matematikk stammer fra gresk og kan oversettes med «læren om det som kan innses». Pytagoreerne var et filosofisk brorskap som ble stiftet av den greske matematikeren Pytagoras rundt 530 f.Kr og for dem var matematikk et samlebegrep for fire vitenskaper: aritmetikk, geometri, sfærikk og musikk. Felles for disse er at tall var selve grunnlaget for å beskrive fenomenene som oppstår i hver av disse grenene (Nygaard, Hundeland & Pettersen, 1999). Det å forstå og få en presis beskrivelse av naturfenomener har vært en viktig drivkraft for utviklingen av matematikk. I tillegg til den praktiske siden, har den intellektuelle siden av matematikk også vært viktig. I boka *What's math got to do with it* beskriver Jo Boaler matematikk både som en menneskelig aktivitet og et sosialt fenomen. Hun ser på matematikk som et sett av metoder for å kunne belyse verden og som en del av kulturen vår. Matematikk er også en måte å uttrykke forhold og ideer på, både numerisk, grafisk, symbolsk, verbalt og billedlig (Boaler, 2008). Brekke og Gjone (2001) velger å beskrive matematikk som det som kan utrykkes eller representeres som et formelt system, som gjerne kan kalles et byggverk. Matematikk er i tillegg den *aktiviteten* som vi utfører når vi på ett eller annet nivå arbeider innenfor dette formelle systemet eller byggverket.

Hvis man spør en elev i skolen hva matematikk er, vil nok svarene være ganske ulik de som er nevnt over. For mange er ordet knyttet til regning, symboler og formler, heller enn finne mønster og se etter sammenhenger. Ofte tenker ikke elever over hva symbolene egentlig representerer, men manipulerer dem etter visse regler, løsrevet fra andre forhold eller «virkeligheten» (Brekke & Gjone, 2001). Dette kan ha sammenheng med at den tradisjonelle opplæringen i matematikk som ofte praktiseres i skolene er av en slik art at matematikk ikke oppleves som verken praktisk, intellektuelt eller sosialt.

3.2 Matematikkopplæring i Norge

Matematikk, både som vitenskap og som undervisningsfag har en lang historie, og tidligere ble grunnleggende kunnskaper i matematikk overført fra en generasjon til en annen. Den greske matematikeren og filosofen Euklid forfattet et av verdens mest kjente matematikkverk *Elementer* om lag 300 år f.Kr., der han samlet og systematisere hellensk geometri. Verket var strengt aksiomatisk oppbygd, dvs. basert på grunnleggende sannheter, også kalt aksiomer. Måten Euklid presenterte geometrien i Elementer, ble stående som et forbilde for hvordan man skal presentere et matematisk system på. Helt frem til vår tid har han hatt stor innflytelse på undervisningen i faget i skolen (Brekke & Gjone, 2001). I 1739 fikk Norge en lov om utdanning av «almuen», som utviklet seg til lov om folkeskole og videre til lover og grunnskoler (Brekke &

Gjone, 2001). Matematikk som eget fag fikk en plass i norsk skole allerede i den første skoleloven som Stortinget vedtok i 1827, men da under navnet regning. Det var lagt opp til at elevene skulle få regneferdigheter, og at de sikkert, raskt og praktisk skulle kunne løse dagligdagse oppgaver, og arbeid med tall og regneartene hadde stor plass. Litt senere kom også måling og geometriske beregninger med, men fagets innhold var lenge relativt stabilt, helt til ca. 1950, og var i hovedsak orientert mot ferdigheter (Breiteig & Venheim, 1998). Siden 1950 har vi hatt mange ulike læreplaner, og i neste delkapittel skal jeg se litt nærmere på hvorfor det skjer endinger i læreplanene for faget matematikk og gå litt innpå de siste læreplanene som har vært i Norge frem til Læreplanen for Kunnskapsløftet 2006.

3.2.1 Endringer i faget – hvorfor?

Samfunnet er i rask utvikling, og Stortinget forsøker gjennom utdanningsreformer å legge et grunnlag for å imøtekomme de kravene som samfunnet til enhver tid stiller til utdanningssystemet vårt. Hvis man ser kun 60 år tilbake i tid har samfunnet endret seg drastisk, spesielt gjennom utvikling av ny teknologi. Teknologien fører til nye muligheter og endring av behov for kompetanse, og den teknologiske utviklingen har skjedd svært raskt. Ifølge Alseth, Breiteig og Brekke (2003) har den teknologiske utviklingen spilt en viktig rolle for fagets planer, ved at innføring av tekniske hjelpemidler har påvirket både fagets innhold og undervisningsmetoder. Tilgangen på digitale resurser i skolen og i samfunnet er allerede stor, men det spås å øke også fremover da samfunnet blir mer og mer digitalisert (Erfjord & Haara, 2018). Kunnskap om hvordan digital teknologi fungerer, er vesentlig i et digitalt samfunn og mennesker vil stadig trenge ny kompetanse og mulighet til å utvikle endringskompetanse.

Mange av dagens skoleelever vil i fremtiden være med på å utvikle teknologi som er viktig for samfunnet, og de trenger derfor utdanning som gjør dem i stand til dette, deriblant er grunnleggende ferdigheter innen programmering viktig (Kaufmann, Stenseth & Holone, 2018). At den teknologiske utviklingen også påvirker matematikkfaget er klart. Ny teknologi og programvare gir matematikklærere nye muligheter i undervisningen, men også utfordringer som ikke fantes før. Matematikk er et fag der forventningen om bruk av digitale verktøy og digitale ressurser er store, noe som også vises når man ser på utviklingen som har vært innen bruk at digitale verktøy på eksamen i matematikk (Erfjord & Haara, 2018). Oppgaver der bruk av regneark, Geogebra og CAS er gitt til eksamen på ulike trinn både på ungdomsskolen og i den videregående skole. Den rivende utviklingen som har vært innen matematikkdidaktikk har vært med på å skape ny teori og kunnskap om læring av matematikk, er et annet argument for endringer i læreplaner, noe som vil belyses senere i kapitlet.

3.2.2 Utviklingen av norske læreplaner

Før kunnskapsløftet kom i 2006 har vi siden 1959 hatt flere læreplaner og jeg vil her ser på de viktigste hovedtrekkene til disse og se læreplanutviklingen i sammenheng med ulike læringssyn som var i vinden da endringene kom.

Fra 1960-1997

I 1959 ble grunnskolen utvidet fra 7 til 9 år og faget regning fikk da navnet matematikk gjennom 59-planen (Alseth m.fl., 2003). 1950- og 1960-tallet var preget av en reformbevegelse, spesielt påvirket av USA, der det behavioristiske synet på læring sto sterkt. Dette synet går ut på at man kan overføre læring, og kunnskap er et bredt spekter av faktakunnskaper og ferdigheter. Ved et slikt syn på kunnskap vil øving være sentralt, og lærebok blir sett på som et av de viktigste

redskapene i undervisningen. Ved bruk av data som verktøy vil drilloppgaver og øvelser stå i fokus (Alseth m.fl., 2003). Den midlertidige mønsterplanen av 1971 (M71) fikk derfor en mer formalistisk retning av faget, og dette endte opp i mer algebraisk og abstrakt fremstilling av matematikk. Reformbevegelsen fikk en motbølge og dette resulterte i at mønsterplanen av 1974 (M74) likevel fikk en mer tradisjonell form, med vekt på grunnleggende ferdigheter i regning og matematikk (Breiteig & Venheim, 1998). De første lommeregnerne, og den personlige datamaskinen (PC) kom inn i skolematematikken på denne tiden og begge disse fikk store konsekvenser for undervisningen. I mange land ble det det et betydelig satsning utdanning og opplæring innen databehandling og i første halvpart av 1980-tallet var mye av fokuset på programmering av datamaskin (Brekke & Gjone, 2001).

På 1970-tallet kom et konstruktivistisk syn på læring for fullt inn i opplæringen, og elevene skulle «konstruere» sin egen kunnskap (Alseth m.fl., 2003). Jean Piagets teori om hvordan vi danner ny kunnskap gjennom de kognitive prosessene assimilasjon og akkomodasjon kommer inn under dette læringssynet. Ved assimilasjon tilpasser vi ny erfaring i de kunnskapsstrukturer vi allerede har, og ved akkomodasjon blir disse kunnskapsstrukturene omstrukturert (Solvang, 1992). Undervisningen skulle i denne tidsperioden bygges på elevenes forkunnskaper og forsterke og utfordre disse, blant annet gjennom kognitive konflikter ved misoppfatninger. I tillegg ble det lagt vekt på aktiviteter, problemløsning og utforskning (Alseth m.fl., 2003). Mønsterplanen av 1987 (M87) var preget av dette læringssynet og datalære og problemløsning kom for første gang inn som egne hovedområder i læreplanen i matematikk. Disse områdene hadde ikke skolen noen tradisjoner i, og dermed lite å bygge på, noe som skapte en del utfordringer for lærerne (Breiteig & Venheim, 2004).

Læreplanen av 1997

På 1990-tallet var det sosiokulturelle læringssynet (også kalt sosial konstruktivisme) en av bidragsyterne til endringene av læreplanene. Vygotskys teori om at læring skjer gjennom sosial aktivitet gjorde at ansvaret for læringen ble spredt på alle deltakerne i prosessen. Organiseringen av opplæringen skulle ha som mål å etablere et læringsfelleskap, der problembasert læring og læringen som en prosess, ble ansett som viktig i undervisningen (Alseth m.fl., 2003). Etter 10 år med M87, ble *Læreplanverket for den 10-årige grunnskolen,* eller bare læreplanen av 1997 (L97) iverksatt. Nå skulle seksåringene starte på skolen, og skolen ble utvidet til et 10-årig løp. Det ble utarbeidet egne mål for hvert trinn som beskrev hva elevene skal ha mulighet til å ha vært i gjennom. L97 hadde klare rammer for hvilke undervisningsmetoder lærerne skulle bruke, med konkrete beskrivelser av disse, og elevenes egenaktivitet var viktig i denne læreplanen. I matematikk var fokuset på praktisk matematikk sentralt, og arbeid med oppgaver og problemer der eksperimentering og utforskning skulle vektlegges. Slik fikk elevene muligheten til å konstruere egen kunnskap gjennom aktivitet. «Matematikk i dagliglivet» ble et eget målområde for opplæringen på alle trinn, i tillegg til «tall og algebra» og «geometri». «Behandling av data» ble eget målområde fra 5. trinn til 10. trinn (Breiteig & Venheim, 2004).

Kunnskapsløftet LK06

I 2004 la departementet frem Stortingsmelding nr. 30 (2003–2004) *Kultur for læring,* som presenterte utdanningsreformen Kunnskapsløftet. Målet med reformen var å gjøre elever og lærlinger bedre i stand til å møte kunnskapssamfunnets utfordringer og i 2006 kom «Læreplanverket for kunnskapsløftet», også kalt LK06. Gjennom LK06 kom grunnleggende

ferdigheter inn i læreplanene, og dette var ferdigheter som skulle gå igjen i alle fag og som skulle integreres i kompetansemålene. Det ble fastsatt kompetansemål for 2. trinn, 4. trinn, 7. trinn og 10. trinn. Kompetansemålene ble utformet slik at de fleste skal kunne nå disse, men i ulik grad av måloppnåelse. Der L97 gikk langt i å beskrive arbeidsformer og timefordelinger, er det i LK06 lagt til rette for at hver enkelt kommune og skole kunne avgjøre dette lokalt. Kommunene og skolene fikk nå mulighet til å utvikle egne lokale læreplaner, utfra de nasjonale, noe som ga lærere og rektorer større frihet både til å velge læreverk og undervisningsmetoder (Kunnskapsdepartementet, 2006). Stor metodefrihet stiller større krav til lærerne som underviser, og selv om friheten ble større, var fortsatt fokuset på at elevenes egen aktivitet er avgjørende for læring sterkt til stede. Lærerne skulle bidra til å skape gode situasjoner for elevenes læring, og læringsstrategier var et viktig begrep i LK06.

Oppsummert er det tydelig at det er behov for fornyelse av læreplanene med jevne mellomrom, både fordi forskning på hvordan elevene lærer gir nye didaktiske perspektiver, og på grunn av teknologiske fremskritt og samfunnets endrede kompetansebehov.

3.2.3 Fagfornyelsen: Kunnskapsløftet 2020

Fagfornyelsen blir den største endringen i skolen siden LK06. Lærere og rektorer har i lengre tid gitt tilbakemeldinger om at dagens læreplaner har for mange temaer, og at de ikke får tid nok til å gå i dybden på det viktigste i fagene. I 2015 og 2016 satte regjeringen i gang et omfattende arbeid for å fornye innholdet i skolefagene. Gjennom NOU-rapporten *Fremtidens skole – fornyelse av fag og kompetanser* (NOU 2015: 8), og stortingsmelding nr. 28 *Fag – Fordypning – Forståelse* (Kunnskapsdepartementet, 2015-2016) er vi mot en fornyelse av kunnskapsløftet der det faglige innholdet skal bygge på de grunnleggende prinsippene i LK06. I læreplanene skal det komme tydeligere frem hvordan de grunnleggende ferdighetene er en del av det faglige innholdet og en forutsetning for elevenes læring i faget (Kunnskapsdepartementet, 2016). Kompetansemålene som vi har i LK06 skal videreføres, men det skal bli kompetansemål etter hvert trinn slik som L97 hadde. Gjeldende fag- og timefordeling skal beholdes, men det skal skje en omfattende strukturendring og effektivisering av fagene. Matematikk vil fortsatt være det nest største faget i norsk skole, kun morsmål har et større timetall i skolen. I matematikkfaget har læreplanene vært lagt opp til det som kalles spiralprinsippet, der samme fagstoff gjentas i læreplanen på ulike trinn. Dette ønsker man i matematikk å gå bort i fra i den nye læreplanen. *Kjerneelementene* blir det viktigste og mest sentrale elevene skal lære i hvert fag, og gir retning og prioriteringer for de nye læreplanene som skal lages, og er det som skal gi føringer for kompetansemålene. (Kunnskapsdepartementet, 2018a).

3.3 Kompetanse

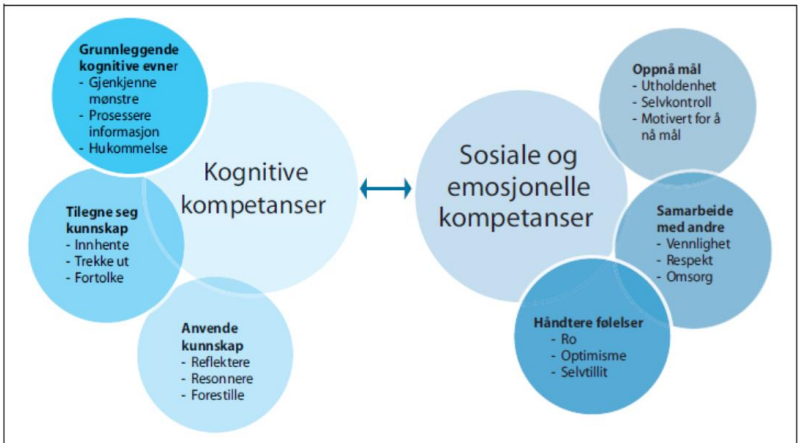
Kompetanse er et begrep som har utviklet seg gjennom ulike læreplaner, og i NOU 2015: 8 definerer Ludvigsenutvalget kompetanse slik:

> Kompetanse betyr å kunne mestre utfordringer og løse oppgaver i ulike sammenhenger og omfatter både kognitiv, praktisk, sosial og emosjonell læring og utvikling, inkludert holdninger, verdier og etiske vurderinger (NOU 2015: 8, s. 14).

Utvalgets definisjon legger vekt på prosessen bak en handling, det praktiske, sosiale og emosjonelle i en læringsprosess. Kompetanse blir sett på som et stort og komplekst begrep som er sammensatt av mange menneskelige holdninger og verdier, ikke bare ferdigheter og

kunnskap. Utvalget anbefaler fire kompetanseområder som grunnlag for fornyelse av skolens innhold: *fagspesifikk kompetanse, kompetanse i å lære, kompetanse i å kommunisere, samhandle og delta, kompetanse i å utforske og skape* (NOU 2015: 8, s. 10-11).

Det første området, fagspesifikk kompetanse, er fundamentet i skolefagene. De tre siste kompetanseområdene er det utvalget kaller *fagovergripende*, det vil si at de er relevante for mange ulike fag og kunnskapsområder. I tillegg til kompetanse som er fagspesifikke eller fagovergripende, har elever i norsk skole behov for å utvikle kompetanser som å kunne lære, kommunisere, samarbeide, delta, utforske og skape. De siste tiårene har det blitt utviklet et robust kunnskapsgrunnlag om hvordan elever lærer i skolen og på andre læringsarenaer, og denne kunnskapen bør ha betydning for hvordan skolen tilrettelegger for læring og læringsmiljøet. Definisjonen av kompetanse viser at det ikke bare er de kognitive kompetansene i fag som er viktige å utvikle, læringskompetanse og sosiale og emosjonelle kompetanser må også gis oppmerksomhet (NOU 2015: 8, s. 9). *Sosiale og emosjonelle kompetanser* går under ikke-kognitive kompetanser, og er kompetanser som involverer det å nå sine mål, samarbeid med andre og håndtere egne følelser (OECD, 2015). Sosial kompetanse handler om de ferdighetene som elevene tar i bruk når de skal omgås andre og emosjonell kompetanse er elevenes evne til å gjenkjenne og regulere følelser (Lindbäck & Glavin, 2015). Forskningen viser at elevenes aktive deltakelse i og refleksjon over egne læringsprosesser fremmer læring. For å tilrettelegge for dybdelæring i opplæringen trekker Ludvigsenutvalget frem disse konseptene som særlig viktige: metakognisjon, selvregulering, kreativitet og innovasjon (NOU 2014: 7, s. 11; NOU 2015: 8, s. 21). Metakognisjon og selvregulering handler om elevenes kompetanse til å lære og begge disse begrepene er eksempler på kompetanser der kognitiv, sosial og emosjonell læring spiller sammen, se figur 6 (NOU 2014: 7, s. 37).



Figur 6: Kognitive og sosiale og emosjonelle kompetanser. Oversettelse hentet fra NOU 2014: 7, s. 38

3.3.1 Fagspesifikk kompetanse

Matematikk, naturfag og teknologi ses på som et eget kompetanseområde i tillegg til språk, samfunns- og etikkfag og praktiske og estetiske fag. Ludvigsenutvalget mener det er viktig at elevene utvikler kompetanse innen disse fire områdene også i fremtiden for at elevene skal ha et fundament som de kan bygge videre på i de utdannings- og yrkesvalgene de gjør (NOU 2015: 8, s. 23). Teknologiutvikling og digitalisering fører til endringer i innhold og metoder innen fagområdene, og hva som kjennetegner kompetanse i de ulike områdene vil derfor videreutvikles kontinuerlig. Det er bred enighet om at matematikkompetanse er viktig siden kunnskapsutvikling i andre vitenskapsfag er avhengig av matematikk, og de fleste vil være

avhengig av matematikk både i hverdagsliv, utdanning og i arbeidssammenheng. Innenfor fagspesifikk kompetanse regnes etisk vurderingsevne, engasjement, holdninger til fag og egen læring i fagene som de sentrale sosiale og emosjonelle kompetansene (NOU 2013: 8, s. 22).

3.3.2 Fagovergripende kompetanser

Kompetanse i å lære

Læringskompetanse handler om metakognisjon og selvregulering. Læring defineres i NOU-rapporten *Elevers læring i fremtiden skole* som en aktivitet der en person tilegner seg ny eller endrer og forsterker eksisterende kunnskap, adferd, ferdigheter, verdier eller preferanser og kan involvere og kombinere ulike typer informasjon (NOU 2014: 7, s. 32). Ifølge rapporten har mye av tidligere forskning handlet om kognitiv læring, mens nyere forskning også tar innover seg at læring ikke bare involverer tankeprosesser/kognitive prosesser, men også personers følelser, motivasjon, sosiale ferdigheter og relasjoner og at læring skjer ved at tenkning, følelser og motivasjon utvikles gjennom et samspill (NOU 2014: 7, s. 32). For at en elev skal opparbeide seg god læringskompetanse ser man en sammenheng med elevers metakognisjon og selvregulering. Elever som klarer å regulere sin egen læringsprosess og klarer å styre denne i positiv retning, klarer i større grad å tilegne seg ny eller forsterke kompetanse. OECD prosjektet *Education in social progress* (ESP) har som mål å forstå hvordan utdanning og kompetanse/ferdigheter (engelsk: skills) har betydning for individets velvære og sosiale fremgang.

Bruk av relevante læringsstrategier er en del av dette, og det er også elevens tro på egen mestring, motivasjon for å lære og evne til å fortsette et arbeid når det butter imot (NOU 2014: 7, s. 11). I 2014 ble det utført en fokus-undersøkelse gjennomført av OECD, der det bl.a. ble gjort analyser av PISA-studien. Undersøkelsen viser sammenhenger mellom egenskaper som motivasjon og utholdenhet og elevenes resultater i matematikk og det kommer tydelig frem at elever som er utholdende gjør det bedre i faget. Norske elever scorer lavere enn gjennomsnittet i OECD-landene, og i undersøkelsen fremheves det spesielt at norske elever skårer lavere på utholdenhet enn OECD-snittet. De utsetter vanskelige oppgaver i større grad enn OECD-snittet og de gir lett opp når de må bryne seg på en vanskelig oppgave (Ertesvåg, 2014).

Kompetanse i å kommunisere, samhandle og delta

Kommunikasjon, samhandling og samarbeid blir ofte sett i sammenheng, både i forskning og i utredningsarbeid om kompetanser for det 21. århundre (NOU 2015: 8). Behovet for kompetanse innenfor kommunikasjon og samarbeid er økende på mange områder og i dag er ulike typer av samarbeidsformer brukt både i skole og i arbeidslivet. Elever og arbeidstakere må kunne samhandle og kommunisere på tvers av ulikheter i bakgrunn, verdier og synspunkter. Å kunne utføre aktiviteter og oppgaver sammen med andre er viktig i arbeidslivet, og mange vil samarbeide om kompleks oppgaveløsning, ofte på tvers av yrker eller fagfelt (NOU 2015: 8). Det er viktig at elever utvikler gode holdninger, og lærer å vise respekt og omsorg for andre. Programmeringsarbeid fordrer til samarbeid, og det er viktig å kunne kommunisere og arbeide mot et felles mål gjennom f.eks. parprogrammering. Ifølge Stahl (2015) kan elevene gjennom verbalisering i gruppearbeid med teknologi som støtte, engasjere seg i å skape mening i den informasjonsstrømmen de sitter i, og den kan fremme en forståelse som vokser fram over tid. Dette er en viktig del av forskningsområdet datastøttet samarbeid (Computer Supported Collaborative Learning), som studerer læring som en sosio-kognitiv prosess. (Stahl, referert til i Sanne m.fl., 2016, s. 31)

Kompetanse i å utforske og skape

Kompetanse i å utforske og skape handler om kreativitet og innovasjon, kritisk tenkning og problemløsning (NOU, 2015: 8, s. 31). Ludvigsenutvalget påpeker videre at det viktigste med å utvikle kompetanse er å kunne anvende den. De beskriver dette som at man innehar kapasitet til å ta i bruk kunnskaper og ferdigheter til å mestre utfordringer og løse oppgaver. Elevenes kunnskap om og forståelse av det de har lært, hvordan de kan bruke det de har lært, og når de kan bruke det, er viktig for å oppnå kompetanse (NOU, 2015: 8, s. 10). Å kunne arbeide med komplekse problemer krever ofte kreativitet og nytenkning. Man må kritisk vurdere den informasjonen man har fått, og velge de problemløsningsstrategiene som er relevante. Kunnskaps- og teknologiutvikling og høye forventninger til at komplekse problemer skal løses, gjør at kreativitet og innovasjon blir viktig i samfunnet og i arbeidslivet fremover (NOU 2015: 8, s. 31).

3.3.3 Kompetanser for fremtiden

I boka *Elefanten i klasserommet at* skriver Jo Boaler (2015) at å kunne matematikk er ikke bare en av de viktigste ferdighetene for fremtidige arbeidstakere, det er også avgjørende for å få et velfungerende og suksessfullt liv. Det er estimert at det vil være over 20 millioner jobber for mennesker med kompetanse innen matematisk problemløsning i det 21. århundret, derfor vil det bli et stort behov for matematikk også i fremtiden. Når Boaler snakker om matematikk, er det ikke bare det å kunne lese grafer og tabeller, eller kunne prosentregning hun trekker frem. Hun påpeker også viktigheten av å kunne resonnere og løse problemer, være fleksible i metodebruk i nye situasjoner, tenke logisk og kunne sammenligne og analysere både tall og datamateriale (Boaler, 2015). Boalers tanker om fremtidens behov er i tråd med det Partnership for 21st Century learning (P21) har beskrevet i sitt rammeverk *21st Century learning.* P21 er en ideell organisasjon med medlemmer fra hele verden og medlemmene representerer i hovedsak myndigheter, skoleledere og lærere. Rammeverket *21st Century Learning* har som mål å fremme hvilke ferdigheter, kompetanser og kunnskap elever bør lære i fremtiden, og skiller mellom 21st Century skills og 21st Century themes[5]. 21st Century skills er ferdigheter som elever bør lære for å kunne fungere i et fremtidig samfunn. Rammeverket deler ferdigheter i tre ulike områder: 1) Livs- og karriereferdigheter, 2) Lærings- og innovasjonsferdigheter, 3) Informasjons-, Media- og Teknologiferdigheter. I tillegg inneholder rammeverket hvilke kjernefagområder og temaer som er viktig for det 21. århundret. Det er fokusert spesielt på det som kalles de fire C-er, som er knyttet til læring- og innovasjonsferdigheter. Disse fire C-ene er: Kritisk tenkning (Critical thinking), Kreativitet (Creativity), Kommunikasjon (Communication) og Samarbeid (Collaboration). Applied Educational Systems utdyper disse begrepene på sin nettside (www.aeseducation.com) slik: Kritisk tenkning ses i sammenheng med problemløsning og det å kunne være kritisk til for eksempel kilder fra internett. Det er viktig at elever lærer hvordan arbeide utforskende, være engasjert og stille spørsmål. Kreativitet ses på som kompetanse i å «tenke utafor boksen», og være innovativ. Kommunikasjon er viktig for å dele ideer og elevene må utvikle kompetanse i å være tydelige og effektive når de kommuniserer, og det er her samarbeid kommer inn som en sentral faktor. Det å lære å samarbeide for å nå et felles mål, som f.eks. løse en kompleks oppgave er essensen i det som er viktig for fremtiden (Applied Educational Systems, 2019). Ved å bruke programmering og algoritmisk tenkning for å løse problemer må elevene ofte ta i bruk alle de fire C-er for å finne en løsning.

[5] Se også www.battelleforkids.org

3.3.4 Lærernes profesjonsfaglige kompetanse

Elever lærer på ulike måter, og det er derfor behov for å variere undervisningsmetodene. Det finnes ingen universalmetode for god undervisning, og lærerens kompetanse spiller en avgjørende rolle. Det er viktig at læreren er bevisst elevenes læreforutsetninger, målet for undervisningen, rammefaktorer som kan fremme eller hemme læring, det faglige innholdet som skal vektlegges, samt hvordan det skal vurderes (Sylte, 2013). Lærerrollen forutsetter derfor solid kompetanse på flere områder. Profesjonsrettet pedagogisk kompetanse er et samlebegrep som innebærer bredde- og dybdeinnsikt som omhandler både den faglige og den didaktiske delen av yrket. Evne til å omsette teoretisk innsikt til praktisk handling er også et sentralt moment i denne kompetansen (Sylte, 2013). Profesjonsrettet pedagogisk kompetanse innebærer også å kunne planlegge opplæringen innenfor de rammene som er fastsatt i lover, forskrifter og læreplanverket i skolen (Sylte, 2013, s. 25). Som lærer, og i et lærerkollegium, er det å holde seg oppdatert på endringer i læreplaner og utvikle den faglige kompetansen slik at den er relevant i forhold til samfunnets kompetansebehov en viktig del av arbeidet. Ny teknologi er med på å endre måten vi lærer, kommuniserer, underholder oss, finner informasjon og tilegner oss kunnskap på. Dette påvirker også lærerens arbeidsmetoder i pedagogisk og didaktisk sammenheng. Kelentric, Helland og Arstorp (2017) ved Senter for IKT i utdanningen har utviklet et rammeverk for lærerens profesjonsfaglige digitale kompetanse. De deler lærerens profesjonsfaglige digitale kompetanse i et tosidig siktemål: Det ene handler om profesjonsutvikling, det andre om selve profesjonsutøvelsen. Ved å innføre begrepet «profesjonsfaglig digital kompetanse (PfDk)» som begrep i 2012, ønsket Senter for IKT i utdanningen synliggjøre betydningen og den sentrale rollen lærerprofesjonen spiller for realisering av digitalisering av skolen og utviklingen av digitalt kompetente elever (Kelentric m.fl., 2017, s. 5). Senteret deler inn kompetanseområdene som vist i figur 8, men understreker at hvert av kompetanseområdene er viktige, men det er summen av dem som utgjør en profesjonsfaglig digitalt kompetent lærer.

Figur 7: Visualisering av rammeverket for lærerens profesjonsfaglige digitale kompetanse (Kelentric, Helland & Arstorp, 2017)

3.4 Dybdelæring

I den nye læreplanen er dybdelæring og forståelse sentrale begreper. Dybdelæring er ikke et nytt begrep og ble først brukt for over 40 år siden i forbindelse med en studie gjennomført av Ference Marton og Roger Säljö. Marton og Säljö (1976) var interesserte i å undersøke hvordan studenter tilnærmet seg en akademisk tekst og hvordan studentenes læringsutbytte ble påvirket av læringsstrategien de brukte. Undersøkelsen fokuserte på hvordan og hva studentene lærte, «what is learned», fremfor hvor mye studentene hadde lært. De brukte begrepene «surface level processing» og «deep level proceccing» som kan oversettes til overflatelæring og dybdelæring. Når studentene benyttet seg av læringsstrategien overflatelæring, var de opptatt av å pugge regler og fakta, uten å se denne i en større sammenheng, og de var mest opptatt av å huske. Studentene som tilnærmet seg lærestoffet gjennom dybdelæring, var mer opptatt av å se sammenhenger og forsøkte å bygge forståelse. Disse elevene viste seg også å ha en indre motivasjon til å lære og forstå, utover det å prestere bra i skolesammenheng (Marton & Säljö, 1976). Det finnes i dag ulike oppfatninger av begrepet dybdelæring, og begrepet brukes ulikt innenfor litteratur og i ulike deler av læringsforskningen (Gilje, Landfeld & Ludvigsen, 2018). Jeg vil her ta utgangspunkt i Ludvigsenutvalgets bruk av begrepet og tabell 1 beskriver kjennetegnene på dybdelæring og overflatelæring hos elevene.

Dybdelæring	Overflatelæring
Elever relaterer nye ideer og begreper til tidligere kunnskap og erfaringer	Elever jobber med nytt lærestoff uten å relatere det til hva de kan fra før
Elever organiserer egen kunnskap i begrepssystemer som henger sammen	Elever behandler lærestoff som adskilte kunnskapselementer
Elever ser etter mønstre og underliggende prinsipper	Elever memorerer fakta og utfører prosedyrer uten å forstå hvordan eller hvorfor
Elever vurderer nye ideer og knytter dem til konklusjoner	Elever har vanskelig for å forstå nye ideer som er forskjellige fra dem de har møtt i læreboka
Elever forstår hvordan kunnskap blir til gjennom dialog og vurderer logikken i et argument kritisk	Elever behandler fakta og prosedyrer som statisk kunnskap, overført fra en allvitende autoritet
Elever reflekterer over sin egen forståelse og sin egen læringsprosess	Elever memorerer uten å reflektere over formålet eller over egne læringsstrategier

Tabell 1 Egenlaget tabell ut ifra Ludvigsenutvalgets oversettelse. Kilde: NOU 2014: 7 s. 36.

Her kan man kjenne igjen Marton og Säljö sin tilnærming til de samme begrepene. Stortingsmelding nr. 28 *Fag – Fordypning – Forståelse* (Kunnskapsdepartementet, 2016), har følgende definisjon på dybdelæring:

> Dybdelæring betyr at elevene gradvis og over tid utvikler sin forståelse av begreper og sammenhenger innenfor et fag. Elevenes læringsutbytte øker når de gjennom dybdelæring utvikler en helhetlig forståelse av fag og ser sammenhengen mellom fag, samt greier å anvende det de har lært, til å løse problemer og oppgaver i nye sammenhenger (s. 14).

Denne definisjonen vektlegger at elevene skal få *tid nok* til å utvikle sin forståelse, og som nevnt før var nettopp kritikken mot LK06 at læreplanene inneholder for mange temaer. Gilje m.fl.

(2018) ser på den kognitive tilnærmingen til dybdelæring og den sosiokulturelle tilnærmingen til dybdelæring i artikkelen *Dybdelæring –historisk bakgrunn og teoretiske tilnærminger*. De påpeker at dybdelæring handler om å forstå og kunne bruke kunnskapen i nye situasjoner. Å memorere kunnskap vil derfor alene ikke være en hensiktsmessig strategi for dybdelæring (Gilje m.fl., 2018). Det kognitive perspektivet fokuserer på hvordan individet tilegner seg og bygger sin egen kunnskap, og legger bl.a. vekt på å utvikle langtidshukommelsen. Det fagstoffet som skal læres må settes inn i en relevant sammenheng, og knyttes opp mot fagenes kjerneelementer. Hvis ikke læring gjøres faglig relevant, vil det være en risiko for at læringen blir fragmentert, og ikke ses i en større sammenheng, men som isolerte deler.

Det kognitive perspektivet på dybdelæring kan ses i sammenheng med Piagets konstruktivisme og fokuserer på hvordan elevene selv tilegner og bygger sin egen kunnskap. Gjennom en vekselsprosess mellom assimilasjon og akkomodasjon bearbeides ny kunnskap i forhold til de begreper, ord og tanker vi har fra tidligere og ny kunnskap dannes ved at vi utvider eller bygger nye skjemaer (Nygaard m.fl., 1999). Det sosiokulturelle perspektivet og dybdelæring gir en nyansert forståelse hvordan dybdelæring skjer gjennom deltakelse i klasserommet. I den sosiokulturelle læringsteorien ses kognisjon og sosial samhandling i sammenheng, og læring beskrives som prosesser og produkter som må ses i forbindelse med denne sosiale konteksten. Det sosiokulturelle perspektivets styrke er at det kobler sammen elevenes individuelle kognitive utvikling med det sosiale samspillet som læreren kan legge til rette for, og kjerneelementene er i den sammenheng viktige i forhold til det læringsarbeidet læreren legger opp til i klasserommet (Gilje m.fl., 2018).

Gilje m.fl. (2018) mener at dybdelæring handler om elevens evne til gradvis utvikling av forståelse av begreper innenfor et fagområde og gjennom problemløsning, analyser og refleksjon kunne arbeide i og på tvers av fag- eller kunnskapsområder. I dagens digitale samfunn har elevene tilgang til enorme mengder informasjon kun ved få tastetrykk, det betyr likevel ikke at all kunnskap er et «klikk» unna. Evnen til å finne, og kritisk analysere informasjon er viktigere enn å kunne fakta. Å velge ut informasjon for så å skape sammenheng mellom ulike informasjonselementer blir en avgjørende kognitiv ferdighet i dagens samfunn (Ludvigsen., 2016). Gjennom dybdelæring skal eleven gjøres i stand til å møte den samfunnsmessige utviklingen fordi de har lært å overføre og anvende kunnskap fra et fag til nye situasjoner. Dybdelæring kan gi grunnlaget for et sett av viktige kompetanser og samfunnsmessig dannelse (Landfald & Gilje, 2018).

3.4.1 Dybdelæring i matematikk

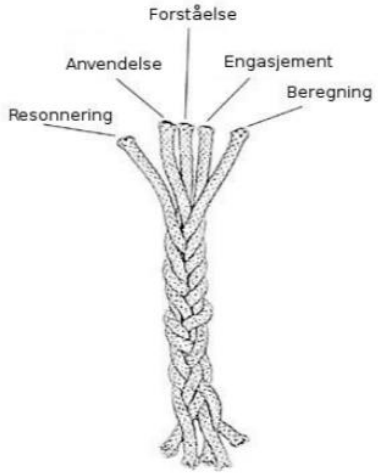
Mona Nosrati og Kjersti Wæge (2018) har med utgangspunkt i tabell 1, funnet fem sentrale komponenter i den matematiske læringsprosessen som beskriver hva dybdelæring i matematikk kan være. Komponentene er hentet fra og satt sammen av forskjellige forskningsbaserte og praksisnære modeller for læring i matematikk. Disse er Kilpatrick, Swafford og Findells trådmodell for matematisk kompetanse, Skemps beskrivelser av relasjonell og instrumentell forståelse, begrepsmessig og prosedyremessig kunnskap beskrevet av Hilbert og Lefevre, og forskningsresultater om metakognisjon (Nosrati & Wæge, 2018). De fem komponentene er:

1. Begrepsmessig forståelse
2. Prosedyrekunnskap
3. Anvendelse

4. Resonnering
5. Metakognisjon og selvregulering

Begrepsmessig forståelse

Dette begrepet er hentet fra trådmodellen til Kilpatrick m.fl. (2001). Kilpatricks modell er delt inn i fem delkomponenter som skal beskrive «mathematical proficiency», som ofte oversettes til matematisk kompetanse på norsk. I tillegg til delkomponenten begrepsmessig forståelse (conseptual understanding), bruker Kilpatrick m.fl. (2001) delkomponentene resonnering (adaptive reasoning), anvendelse (strategic competence), engasjement (productive disposition) og beregning/prosedyrekunnskap (procedural fluency). Figur 9 viser at de fem delkomponentene er tett sammenflettet. Disse fem komponentene støtter hverandre og henger tett sammen, og elevene bør få mulighet til å utvikle alle disse komponentene sammenhengende. På denne måten vil disse forsterke hverandre og elevene vil oppleve å utvikle matematisk kompetanse som er fleksibel, nyttig og relevant (Valenta, 2015).



Figur 8: Norsk versjon av Kilpatricks trådmodell. Hentet fra matematikksenteret.no

Begrepsmessig forståelse handler om evne til å forstå enkeltbegrep, operasjoner og sammenhenger mellom begrep i matematikk, og ikke bare kunne memorere formler og regler. Dette innebærer at evnen til å se de matematiske relasjonene i sammenheng med hverandre, er like viktig som det å forstå de individuelle bitene med fakta og informasjon. Elever som har utviklet begrepsmessig forståelse er i stand til å tolke, forstå og benytte ulike representasjoner, og de kan velge representasjoner som er nyttige i en gitt situasjon (Nosrati & Wæge, 2018). Begrepsmessig forståelse henger tett sammen med det Hiebert og Lefevre (1986) kaller begrepsmessig kunnskap, eller som Skemp (1976) kaller for relasjonell forståelse. Elever som innehar relasjonell forståelse vet ikke bare hvordan en oppgave skal løses, de vet også hvorfor, og har bygget seg mentale strukturer som gjør at de kan løse oppgaver mer effektivt enn elever som ikke har opparbeidet seg dette.

Prosedyrekunnskap

Prosedyrekunnskap, kan vi også finne i trådmodellen, som på norsk ofte er oversatt til beregning. Hiebert og Lefevre (1986) bruker også dette begrepet og beskriver prosedyrekunnskap som kunnskap om regler og prosedyrer for å løse problemer, da ofte i form av oppskrifter eller algoritmer for å manipulere symboler. Prosedyrekunnskap handler om ferdigheter til å gjennomføre prosedyrer fleksibelt, nøyaktig, effektivt og hensiktsmessig (Botten, 2016). Dette innebærer å utvikle og bruke varierte strategier i arbeidet med

regneoperasjoner. Effektivitet og nøyaktighet er viktige elementer i beregning, og det handler også om automatisering av en del regneferdigheter for å frigjøre kapasitet til å arbeide med problemstillinger (Nosrati & Wæge, 2018). Prosedyrekunnskap kan ses i sammenheng med det Skemp (1976) kaller for instrumentell forståelse, der eleven vet hvordan man skal løse en oppgave, men vet ikke hvorfor det er slik. Instrumentell forståelse innebærer en del pugging av algoritmer og regler, og kan ofte oppfattes som kjedelig, men som likevel er viktig for å kunne løse mer komplekse problemer mer effektivt.

Både begrepsmessig forståelse og prosedyrekunnskap er begge viktige for å få til dybdelæring. Man bør ha automatisert en del regler for å kunne frigjøre kapasitet i arbeidet med en problemstilling. Ifølge Nosrati og Wæge (2018) er det viktig å understreke at prosedyrekunnskap bør følges av begrepsmessig forståelse, og at disse to begrepene henger tett sammen og støtter hverandre. I undervisningssammenheng knyttes ofte instrumentell forståelse sammen med tradisjonelle undervisningsformer, ofte kalt tavleundervisning. Utforskende og undersøkende undervisningsmetoder er knyttet til relasjonell forståelse, og skiller seg fra tradisjonell undervisning ved at elevaktivitet og problemløsning ofte er i fokus (Nosrati & Wæge, 2018).

Anvendelse

Anvendelse, eller strategisk tankegang som det også kalles, innebærer å kunne gjenkjenne og formulere matematiske problem, representere dem på en hensiktsmessig måte, tenke fleksibelt i utvikling av en løsningsstrategi og vurdere hvor rimelige løsningene er (Valenta, 2016). Dette begrepet knyttes til problemløsningskompetanse, og med matematiske problemer menes her enten problemer fra hverdagene eller samfunnet der matematikk kan anvendes for å finne en løsning, eller mer abstrakte matematiske problemer eller spørsmål. En elev som behersker strategisk tankegang klarer å anvende ulike strategier for å løse et matematisk problem, og kan i tillegg være fleksibel og effektiv i valg av hvilken strategi som er best for å løse problemet (Kilpatrick m.fl., 2001). Siden problemløsning ofte blir sett i sammenheng med programmering velger jeg å belyse akkurat dette emnet i et eget delkapittel, selv om den ikke er tatt med som et eget punkt i Nosrati og Wæges modell av dybdelæring i matematikk.

Resonnering

Resonnering handler om å kunne forklare hvordan man tenker, å kunne følge med i et logisk resonnement og kunne vurdere dets gyldighet. Videre innebærer resonnering å kunne se og begrunne sammenhenger mellom ulike begreper, egenskaper og framgangsmåter (Nosrati & Wæge, 2018). I matematikk er resonnering «limet» som holder alt sammen, og man bruker resonnementer til å navigere gjennom fakta, prosedyrer, konsepter og strategier og for å få en forståelse for hvordan alt henger sammen. Resonnering handler også om å kunne argumentere for gyldigheten av en hypotese eller bevis, ved å utforme et resonnement og kunne både forklare og forsvare dette (Kilpatrick m.fl., 2001). Johan Lithner (2008) skiller mellom kreativ og imitativ resonnering. Kreativ resonnering må oppfylle følgende kriterier: nyskapende, fleksibel, troverdig og ha et matematisk grunnlag. Når han mener det skal være nyskapende, mener han nyskapende for eleven selv. Eleven skal ikke være bundet til en bestemt strategi, men fleksibelt kunne benytte seg av, og tilpasse, ulike strategier for å løse den aktuelle oppgaven (Lithner, 2008). Imitativ resonnering kjennetegnes ved at den er basert på erfaringer og er ikke nyskapende. Lithner (2008) har identifisert to typer imitativ resonnering, memorert resonnering og algoritmisk resonnering. Memorert resonnering er når elevens strategivalg går

ut på å huske et fullstendig svar og utføringen av strategien består i å skrive ned svaret, mens et algoritmisk resonnement innebærer at eleven husker en løsningsalgoritme (Lithner, 2008). Memorert resonnering er verdifullt når man skal besvare faktaspørsmål, og kun har behov for å bruke hukommelse. Algoritmisk resonnering er nyttig når man trenger en bestemt oppskrift, formel eller algoritme, for å løse et problem, og kun regnefeil kan gi feil svar. Hvis man ser paralleller til begrepsmessig forståelse og prosedyrekunnskap, vil eleven ha behov for å kunne resonnere kreativt for å vise at han eller hun har opparbeidet seg en begrepsmessig forståelse, mens prosedyrekunnskap støttes av imitativ resonnering.

Metakognisjon og selvregulering

Siden kognisjon handler om den mentale prosessen med å tilegne seg kunnskap og forståelse gjennom tenking, erfaring og sanser (Wæge & Nosrati, 2018, s. 65), vil ordet *metakognisjon* kunne oversettes til «kognisjon om kognisjon». Wæge og Nosrati (2018) beskriver metakognisjon i sin bok *Motivasjon i matematikk* som det å kunne ta et mentalt «steg» tilbake fra det man holder på med, lærer om eller tenker på, og bevisst tenke gjennom egne fremgangsmåter og kognitive prosesser. I boka viser de til Pintrich's (2002) tre sentrale aspekter ved metakognitiv kunnskap som lærer og elever kan arbeide med:

- Kunnskap om strategier. Dette er kunnskap om forskjellige strategier for å lære, tenke og løse problemer. Det kan handle om å lære noe utenat, f.eks. gangetabellen, eller hvordan man skal forstå og filtrere ut informasjon fra en tekstoppgave, og bruke denne informasjonen til å lage tankekart, tabeller eller andre visuelle hjelpemidler. Det kan også være kunnskap om ulike problemløsningsstrategier (Pintrich, 2002). Noen av disse strategiene er beskrevet i kapittel problemløsning (3.4.2).
- Kunnskap om kognitive oppgaver. Denne kunnskapen handler om å kunne skille mellom enkle og mer komplekse problemstillinger, og det å kunne velge den mest hensiktsmessige strategien (Wæge & Nosrati, 2018).
- Kunnskap om seg selv (selvinnsikt). Dette handler om å ha kunnskap om å ha innsikt i sin egen læringsprosess, og kjenner til sine egne styrker og svakheter i læringsprosessen. Det er viktig å kunne identifisere disse så presist så mulig for å kunne jobbe med disse videre (Wæge & Nosrati, 2018). Det å være bevisst hvilke strategier man velger i ulike situasjoner, er viktig og Pintrich (2002) påpeker at man i tillegg til selvinnsikt, må være bevisst sin egen motivasjon og mestringsevne til f.eks. å fullføre oppgaver.

Oppsummert kan man si at metakognisjon handler om å utvikle *bevissthet rundt* og *kunnskap om* egen kognisjon. Gjennom aktiv deltagelse i matematiske situasjoner kan lærere hjelpe elevene å utvikle dette gjennom aktiv deltakelse i matematiske situasjoner, noe som kan føre til at elevene får større glede i læringsprosessen (Wæge & Nosrati, 2018, s. 67).

Selvregulering handler om hvordan elevene kan gå inn og styre sine egne læringsprosesser, og dette henger sammen med at eleven har utviklet kunnskap i metakognisjon. Ifølge Wæge og Nosrati (2018, s. 68) er det ulike strategier som elever kan bruke for å styre egen læring omfatter blant annet:

- Sette seg mål (dette kan være alt fra en gitt karakter i matematikk til det å få oversikt over et tema som man vet at man ikke helt har forstått)

- Sette seg delmål på veien til et større mål dersom det er nødvendig.
- Overvåke fremgang (nærmer man seg de målene man har satt seg?)
- Endre både lærings- og problemløsningsstrategier, hvis de man har tatt i bruk ikke gir ønsket resultat

Alle disse fem komponentene som er beskrevet over må ses i sammenheng med hverandre og utvikles parallelt og ses i sammenheng med hverandre, i likhet med trådmodellen til Kilpatrick m.fl. (2001). På hver sin måte vil disse komponentene kunne bidra til elevers dybdelæring i matematikkfaget.

3.4.2 Problemløsning

Gjennom læreplanen M87 ble matematikkfaget for første gang forsøkt fremstilt som mer enn regler, formler og fremgangsmåter, og matematisk problemløsning ble ett av elleve hovedemner i læreplanen. Selv om fokus på problemløsning kom sent inn i skolematematikken, er problemløsning i matematikk trolig like gammel som matematikken selv (Solvang, 1992). I litteraturen blir problemløsning ofte kalt heuristikk, som betyr oppfinnelseskunst, og den første vi kjenner til som formulerte tanker omkring problemløsning var den greske matematikeren Pappus, allerede 250 år f.Kr. I nyere tid er det den kjente matematikeren George Pólya, som gjennom boka *How to solve it* gav et viktig bidrag til å forstå hva problemløsning er. Ved problemløsning tar vi i bruk kunnskap og ferdigheter vi har opparbeidet oss, og evnen til problemløsning er viktig ikke bare i skolesammenheng, men også i livet generelt.

Ragnar Solvang (1992) deler opp de utfordringer vi møter til daglig i to:

1. Utfordringer som vi har løsningsmetoder til å mestre. Disse kan kalles øvelse, rutineoppgave etc.
2. Utfordringer som vi ikke har løsningsmetoder til å mestre, slike utfordringene kalles problemer (s. 314).

Videre definerer Solvang (1992) begrepet problem slik: «En utfordring vil for en person være et problem dersom denne personen ikke har noen algoritme som vil gi en løsning når personen konfronteres med utfordringen», og videre defineres problemløsning slik: «Problemløsning er å søke etter handlinger som fører til løsning av et problem» (s. 135). Slik begrepene problem og problemløsning er definert av Solvang (1992), vil oppgaver gitt i undervisningssituasjoner ofte være rutineoppgaver for læreren, men problemer for elevene. En viktig fase i problemløsningen vil da være å forsøke å ta i bruk metoder som har ført til suksess tidligere. Hensikten med å få elever til jobbe med problemløsningsoppgaver er å inspirere dem til arbeid og for å sette i gang tankeprosesser (Alseth & Røsseland, 2014).

Nygaard m.fl. (1999) skiller i boka *Aha* mellom utforsking og problemløsning. De mener at i problemløsning er hovedfokuset å løse problemet, samtidig som veien mot løsningen ikke skal kunne finnes i en algoritme eller prosedyre. I utforskningsaktiviteter er det hele prosessen frem mot løsningen som er viktig, ikke kun løsningen i selg selv. Selv om man kanskje ikke kommer frem til en løsning i det hele tatt kan man ha utført en mengde matematisk arbeid og funnet verdifulle sammenhenger likevel (Nygaard m.fl., 1999). Van De Walle, Karp og Bay-Williams (2014) deler læring av problemløsning i tre ulike deler, som alle er viktige for å være en god problemsløser. Gjennom å lære seg prosedyrekunnskaper som beskrevet i kapittel 3.4.1 læres kunnskaper, ferdigheter og algoritmer som effektiviserer arbeidet med problemløsing, dette

kalles å lære *for* problemløsning. Å lære *om* problemløsning, er å lære seg strategier og prosesser for å komme frem til løsninger, herunder kommer Pólyas strategier som vi skal studere nærmere i neste delkapittel. Til sist kan man lære *gjennom* problemløsing, dvs. at elever lærer matematikk gjennom oppgaver, modeller, situasjoner eller kontekster. Både utforskning og problemløsning er sett på som viktige kjerneelementer i matematikk i den nye læreplan som er under utarbeidelse.

Problemløsningsstrategier

«Det finnes ingen kongevei til problemløsning» (Moses, sitert i Solvang, 1992). Jeg vil her presentere Pólyas fire hovedpunkter som en strategi til problemløsning, for en mer detaljert beskrivelse anbefaler jeg å lese boka til Pólyas bok *How to solve it*. Pólyas bok *How to solve it* gir ingen endelige strategier som kan løse alle problemer, men han har i hovedtrekk strukturert hva det vil si å løse problemer. Pólyas struktur er bygd opp omkring fire hovedpunkter:

1. Å forstå problemet
2. Å legge en plan
3. Å gjennomføre planen
4. Å se tilbake

Under hvert av disse hovedpunktene anbefaler Pólya en rekke spørsmål som problemløseren bør stille seg i forsøket på å løse problemet. Prosessen er ikke lineær, og en må ofte bevege seg frem og tilbake mellom punktene (Pólya, 1957). I punkt 1 er det forutsetningene og betingelsene som gjelder i problemet som er viktige.
Ved hjelp av ledespørsmål vil Pólya ha problemløseren til å tenke gjennom hva situasjonen går ut på, og at elevene helst skal mestre å klargjøre problemets forutsetninger på egen hånd (Solvang, 1992). Hvis ikke eleven mestrer dette selv, har læreren en viktig jobb med å stille de rette spørsmålene, slik at elevers forkunnskaper rundt temaet de skal jobbe med aktiveres. Punkt 2 mener Solvang (1992) ikke bør ses på som absolutt, og mange av spørsmålene en bør stille her kommer før planen er lagt. Dette punktet innebærer å få ideer og innfall, eventuelt se om man kommer opp med liknende, og kanskje enklere, problemer enn det som er gitt i selve problemet. Som lærer bør man stille spørsmål som kan hjelpe elevene å identifisere viktige deler av problemet, eller tenke på liknende problemer som kan belyse hovedproblemet. Når man har så har kommet frem til en ide eller strategi om hvordan problemet kan løses gjennom utregninger, tegning av skisser etc., har man lagt en plan. Punkt 3 er å gjennomføre planen. Dette punktet avhenger av om vi har en ide til løsning som fører frem. Her er det viktig å gjennomføre planen slik man planla i punkt 2 (Pólya, 1957). Det siste punktet som handler om å se tilbake, er et punkt «uerfarne» problemløsere ofte dropper, i begeistring av å ha fått en løsning. Det er viktig at vi øver opp denne evnen til å se tilbake og vurdere løsningen, uavhengig om vi har fått rett løsning eller ikke. Hvis løsningen viser seg å stemme, gir dette punket en mulighet til å tenke gjennom hva vi har lært av dette problemet. Hvis løsningen viser seg å ikke stemme, må man gjennomgå punkt 2, for å forsøke å identifisere hvor i planen det gikk galt. Dette er en viktig del i arbeidet med problemløsning (Solvang, 1992).

For å lykkes med problemløsning er det som viktig med gode strategier for å komme i mål. Som Albert Einstein visstnok skal ha uttrykt: «If I were given an hour in which to do a problem upon which my life depended, I would spend 40 minutes studying it, 15 minutes reviewing it and 5 minutes solving it.». Arbeid med problemløsing gir elevene muligheter til å utvikle en helhetlig matematisk kompetanse. Elever med høy problemløsningskompetanse klarer å velge riktige

strategier, og klarer å bevege seg mellom ulike strategier. I tillegg får læreren mulighet til å vurdere elevenes kompetanse og få innsikt i hvordan de tenker. Elever som får eksplisitt undervisning i sentrale matematiske problemløsingsstrategier blir bedre problemløsere enn elevene som får tradisjonell undervisning. Torkildsen (2017) påpeker at arbeid med problemløsningsaktiviteter bidrar til økt forståelse og dybdelæring hos elevene (Torkildsen, 2017). Betydningen av problemløsing for elevenes læring og forståelse i matematikk fremheves også av forskere: «Problemløsing bør være arenaen der alle tråder av matematisk kunnskap konvergerer. Det bør gi muligheter for elevene til å veve sammen kunnskapstrådene og for lærere til å vurdere elevenes arbeid (performance) med alle trådene» (Kilpatrick, Swafford & Findell, sitert i Stedøy & Torkildsen, 2018, s. 1).

Problemløsning og programmering

Algoritmisk tenkning handler i stor grad om å løse problemer, og vi kan finne flere fellestrekk mellom problemløsning og algoritmisk tenkning der vi bruker programmering for å løse et problem. Programmering utfordrer elevene med ulike problemstillinger og det er sjelden en løsning eller fasit på hvordan utfordringen skal løses. Det legges til rette for at elevene må bruke relevante strategier for å løse utfordringene de møter i arbeidet med programmeringen (Sanne m.fl., 2016). En jobber problemløsende når man programmerer fordi man først må finne en løsning på problemet, for å så reflektere hvordan man skal gjøre dette om til en kode som datamaskinen forstår og klarer å gjennomføre (Papert, 1980). Når man arbeider med programmeringsoppgaver bruker man en lik prosess som Pólyas fire hovedpunkter ved problemløsning, se tabell 2.

Pólyas 4 hovedpunkter:	Programmeringssteg:
1. Forstå problemet	1. Forstå problemet
2. Lag en plan	2. Finn opp en algoritme som kan løse problemet
3. Gjennomfør planen	3. Skriv koden til algoritmen som et program
4. Se tilbake	4. Vurder programmet

Tabell 2: Sammenheng mellom Pólyas fire hovedpunkter og programmeringssteg

I artikkelen *teknologi og programmering for alle* (Sanne m.fl., 2016) brukes begrepet «bricolage» som er et fransk ord brukt av Seymour Papert. Bricolage handler om å arbeide kreativt med de verktøy og ressurser som til enhver tid er tilgjengelig, mye på samme måte som når barn konstruerer lek. På norsk kan vi kanskje snakke om «fiksing og triksing» for noe av det samme. Analytisk tilnærming til problemløsning og bricolage ses ofte som motsatser, men ofte veksler vi mellom å improvisere med verktøyene som er tilgjengelig og å være systematisk og analyserende. Programmering beskrives som en kreativ prosess der man bruker verktøy man har tilgjengelig for å løse en oppgave, samtidig som det stilles krav til systematikk, algoritmisk tenkning og analyse. Det kan derfor argumenteres med at programmering innebærer elementer av begge (Sanne m.fl., 2016). Som beskrevet i kapittel 2.6 har nyere forskning også vist at arbeid med programmering har effekt på problemløsningsferdighetene. Problemløsningsoppgaver, som også inkluderer programmering, kan forbedre produktivitet, fremme selvstendighet, øke mestringsevnen og forsterke et «growth mindset»[6] (Loksa & Ko, 2016). I programmering, som i problemløsning, er det å se tilbake essensielt, spesielt når du ikke finner rett løsning. I

[6] Growth mindset er en måte å tenke på som fremmer endring og utvikling av personlighet, intelligens, prestasjoner om mye annet. Stammer fra psykologen Carol Dweck. https://snl.no/growth_mindset

programmering kalles dette for «debugging» eller feilsøking. Når elevene programmerer, får de umiddelbar tilbakemelding på om det de har gjort er rett (de ser om det virker etter intensjonen eller ikke). De fleste verktøyene for programmering, inkludert programvare som brukes av profesjonelle programmerere, vil også gi indikasjoner på hva som kan være feil (Selvik m.fl., 2016). Redselen for å ha regnet feil, eller kommet frem til feil løsning kan være med på å begrense elevers læring, noe som kan ende med at de gir opp, istedenfor å ta inn over seg at feil er en naturlig del av læringsprosessen (Nostrati & Wæge, 2015). Det viser seg at arbeid med programmering og feilsøking kan få elever til å endre elevenes negative holdninger til feil. Dette påpekte Papert (1980) allerede på 1980-tallet:

> The process of debugging is a normal part of the process of understanding a program. The programmer is encouraged to study the bug, rather than forget the error. The Turtle context there is a good reason to study the bug. It will pay off. (Papert, 1980 s. 61)

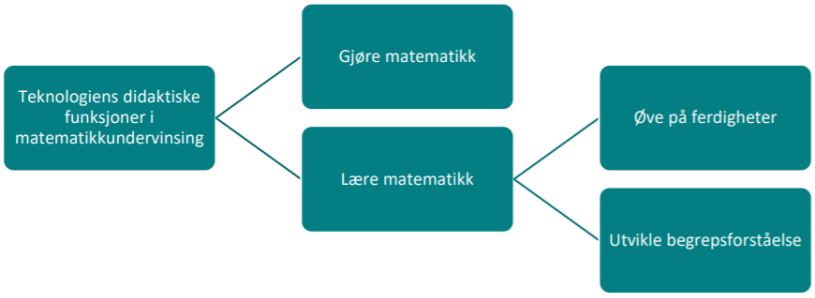
Siden elevene gjennom programmering hele tiden må drive feilsøking og erfare dette som en naturlig del av en læringsprosess, vil holdningen til å gjøre feil muligens endres.

3.4.3 Teknologiens didaktiske funksjon

Som beskrevet i kapittel 2 er hjelpemidler i matematikk ikke noe nytt fenomen og utvikling av ny teknologi har i dag mange bruksområder innen matematikk. I 2008 formulerte NTCM (National Council for Teachers of Mathematics in the United States) sin stilling til teknologi slik:

> Strategic use of technology in the teaching and learning of mathematics is the use of digital and physical tools by students and teachers in thoughtfully designed ways and at carefully determined times so that the capabilities of the technology enhance how students and educators learn, experience, communicate, and do mathematics. Technology must be used in this way in all classrooms to support all students' learning of mathematical concepts and procedures, including those that students eventually employ without the aid of technology. Strategic uses support effective teaching practices and are consistent with research in teaching and learning. (NTCM, 2008, s. 1).

NTCMs formulering anerkjenner at strategisk bruk av teknologi i undervisning og læring av matematikk kan bidra til hvordan elever lærer, erfarer, kommuniserer og gjør matematikk. NTCM understreker at de digitale og fysiske verktøyene skal brukes på en måte som støtter opp om studentenes læring av matematiske konsepter og prosedyrer. Drijvers (2015) skiller mellom tre didaktiske funksjoner teknologien kan ha i matematikkundervisningen, se figur 9 på neste side.



Figur 9: Teknologiens didaktiske funksjoner i matematikkundervisning. Oversatt fra Drijvers (2014), hentet fra Voll & Vinje (2018)

Bruk av teknologi for å effektivisere beregninger kalles for å *gjøre matematikk* med teknologi. I mange tilfeller vil bruk av kalkulatorer for å gjøre større beregninger, eller benytte CAS for å finne integraler være svært tidsbesparende i forhold til å gjøre beregningene for hånd. Programmering kan benyttes som kalkulator og ved å lage egne eller bruke ferdiglagede programmer kan integraler eller andre matematiske beregninger enkelt løses også med programmering. Effektivitet, nøyaktighet og bruk av gode strategier er viktige faktorer i arbeid med matematikk, og i mange situasjoner vil det være hensiktsmessig å benytte digital teknologi for å frigjøre tid og kapasitet til å arbeide med ulike problemer. For å bruke teknologi på denne måten trenger ikke eleven å forstå teknologien, men må kunne bruke den og vite i hvilke situasjoner det er nyttig å bruke det. Dette kan ses i sammenheng med prosedyrekunnskap som beskrevet i kapittel 3.4.1. Teknologi kan bidra til læring av matematikk gjennom å *øve på ferdigheter* eller *utvikle begrepsforståelse.* I dag finnes mange programmer og nettsteder der elevene kan øve på matematiske ferdigheter. Mange læreverk har i dag egne nettsteder der elevene kan får trening i grunnleggende regning, der programmene selv sjekker om eleven får rett svar eller ikke, eller kan komme med hint eller forklaringer til oppgavene. Bruk av digital teknologi på denne måten kan gi elevene mengdetrening og bidra til prosedyrekunnskap og automatisering av ferdigheter. Drijvers, Boon og Van Reeuwjik (2011) trekker frem teknologiens betydning for å utvikle matematiske konsepter og mentale modeller. Ved å bruke digitale verktøy kan læreren visualisere et konsept, eller presentere dette på en dynamisk måte som kan bidra til en dypere forståelse av konseptet. Digitale verktøy kan også brukes til å skape nysgjerrighet eller undring hos elever, og benyttes for å generalisere eller undersøke relasjoner mellom matematiske objekter (Drijvers m.fl., 2011). Bruk av digitale verktøy på denne måten kan bidra til økt begrepsmessig forståelse hos elevene, og hjelpe elevene til å bygge mentale strukturer som effektiviserer elevers arbeid med matematikk.

I dette kapittelet har jeg forsøkt å forstå hvorfor læreplanene endres, og også hvordan kompetansebegrepet er definert i fagfornyelsen. Dette for å forstå begrunnelsene i forhold til hvorfor programmering nå er på tur inn i læreplanene i flere fag. Dybdelæring er også et sentralt begrep i forbindelse med fagfornyelsen, og det er derfor viktig å forstå dette begrepet, også i matematisk sammenheng. Teorien som er belyst i dette kapitlet ønsker jeg å bruke for å drøfte argumentene som kommer frem som funn i kapittel fem, slik at jeg kan se argumentene i en faglig og historisk sammenheng.

4 Metode

I dette kapittelet vil jeg presentere forskningsmetodisk tilnærming og forskningsdesign ble brukt for å besvare mine forskningsspørsmålene. Det er viktig at målene med en studie står sentralt i forhold til hvilket forskningsdesign som brukes, og målene kan deles opp i personlige, intellektuelle og praktiske mål (Krumsvik, 2014). Jeg jobber som matematikklærer i den videregående skole, derfor vil den nye læreplanen som er ferdig høsten 2019 ha direkte og stor innvirkning på min arbeidshverdag. Mine personlige mål handler om at jeg fra høsten 2020 skal undervise i elever i programmering, og ved hjelp av denne studien ønsker jeg derfor å belyse hva programmering kan tilføre matematikkfaget av både fordeler og ulemper, og undersøke hva som må til for å lykkes med implementeringen av programmering i matematikkfaget. Jeg har alltid vært interessert bruken av digitale verktøy i undervisningen, og ønsket derfor å kunne bruke denne oppgaven til å utforske hvilke nye muligheter og utfordringer programmering kan gi meg som lærer i undervisningssammenheng. De intellektuelle målene er å undersøke om teori knyttet til matematikkompetanse kan underbygge argumentene for eller imot innføringen av programmering slik at disse argumentene kan begrunnes faglig i forhold til forskning om læring og undervisning i matematikk. De praktiske målene handler om å belyse et tema som er svært sentralt og omdiskutert i norsk skole og blant lærere i dag, og i tillegg kunne bidra til å at andre lærere eller skoleinteressenter kan se hvilke argumenter som kan brukes i debatten om innføringen av programmering, da spesielt med hensyn på matematikkfaget.

4.1 Vitenskapelig tilnærming

For å besvare forskningsspørsmålene mine er erfaringene og opplevelsene til informantene sentrale, da jeg ønsker å beskrive og belyse ulike argumenter som brukes i debatten rundt programmering. Forskningsspørsmålene er avgjørende for valg av forskningsmetoder som skal velges, og hvilken metodisk tilnærming forskeren bør benytte. Det er ofte vanlig å skille mellom kvalitativ og kvantitative forskningsmetoder, og ifølge Alan Bryman (2016) er dette skillet et nyttig middel for å klassifisere ulike metoder for forskning. For min masteroppgave ble det naturlig å velge en kvalitativ forskningsmetode, og begrunnelsen for dette vil jeg utdype i dette delkapittelet.

I kvalitativ forskning er man opptatt av ord istedenfor tall og forskeren har ofte en *induktiv* tilnærming på forholdet mellom teori og forskning. Med induktiv tilnærming menes at teorien genereres utfra datamaterialet og ikke omvendt som er tilfelle ved en deduktiv tilnærming. Ved induktiv tilnærming antar eller utvikler forskeren noen generelle sammenhenger ut fra datamaterialet (Tjora, 2017). I min oppgave vil det være naturlig å velge en induktiv tilnærming til emnet programmering, fordi jeg ved hjelp av innsamlet datamateriale ønsker å se etter generelle sammenhenger/argumenter i debatten. Ifølge Malterud (2011) kan kvalitativ forskningsmetodikk bidra til å presentere mangfold og nyanser i det feltet som induktivt tilnærmes og personintervjuer vil kunne bidra til å belyse feltet gjennom egne opplevelser og erfaringer. Bryman (2016) påpeker at selv om forskningen i utgangspunktet beskrives som induktiv, vil prosessen i seg selv ofte ha elementer fra deduktiv tilnærming. Dette forklares med at en forskningsprosess ofte er iterativ siden forskeren beveger seg frem og tilbake mellom data og teori, noe som også var tilfellet i arbeidet med denne oppgaven.

Innen kvalitativ forskning er perspektivet på kunnskapens natur og hvordan man oppnår kunnskap *interpretivistisk*. Forskere som velger et interpretivistisk perspektiv er som regel ute

etter opplevelser og oppfatninger fra deltakerne i forskningen, og ønsker å forstå deltakernes verden gjennom å få innsikt i deres bakgrunn, opplevelser, tro og meninger (Thanh & Thanh, 2015). Interpretivister søker vanligvis å tolke og forstå, og har en tro på at virkeligheten er sosialt konstruert, og at vi tolker og lærer av hverandre. Robson (2011) trekker frem at metodene som oftest benyttes for å få innsikt i deltakernes verden, er intervju og observasjoner. Deltakerne, sammen med forskeren, er med på å konstruere «virkeligheten» omkring forskningsområdet, og det påpekes at forskerens subjektivitet på den måten integreres i forskingen. Bryman (2016) viser også til at kvalitativ forskning har et syn på at den sosiale virkeligheten kontinuerlig formes av, og gjennom, de involverte aktørenes handlinger og språk. Slik vil verden, slik den oppfattes av aktørene, være i konstant bevegelse. Dette perspektivet på virkeligheten og verden kalles for *konstruksjonistisk* der forskeren erkjenner at det ikke finnes én «sannhet», men at sannhet og virkelighet skapes av interaksjonen mellom mennesker og samfunnet de lever i, og den gjensidige påvirkningen mellom disse (Berger & Luckmann, 1967).

For min oppgave er det informantenes opplevelser, erfaringer og meninger omkring programmering som vil utgjøre «virkeligheten» eller «sannheten» omkring emnet. Denne virkeligheten kan tolkes og forandres, ettersom ny kunnskap om emnet oppstår eller mennesker får nye erfaringer og opplevelser. I min forskning er jeg er nødt til å tolke informantenes utsagn, og gjennom informantene forsøke å konstruere en beskrivelse av virkeligheten slik jeg, sammen med mine informanter, opplever og erfarer programmeringens fordeler og baksider. Min forskning vil derfor kunne sies å ha et interpretivistisk syn på hvordan kunnskap om verden skapes, og et konstruksjonistisk syn på verden og hvordan den endres utfra ulike perspektiver. På bakgrunn av dette vil det være naturlig å velge en kvalitativ forskningsmetode for arbeidet med denne oppgaven.

4.1.1 Forskningsdesign og forskningsstrategi

For å velge riktig forskningsdesign er det viktig å ha tenkt gjennom hva målet med forskningen er, hvilke teori og forskningsspørsmål som velges, hvilke metoder man skal bruke og hvordan man skal samle inn data (Robson, 2011). Robson (2011) skiller mellom bestemt design og fleksibelt design, der fleksibelt design ofte er å foretrekke i kvalitativ forskning fordi man ofte starter med en enkelt ide eller et problem. Fleksible design utvikler seg underveis, og det er vanlig å bruke datatriangulering, dvs. bruke ulike metoder for å samle inn data. I denne oppgaven ble det naturlig å bruke et fleksibelt forskningsdesign siden den vitenskapelige tilnærmingen til forskningen min er kvalitativ som beskrevet i forrige delkapittel, og jeg har brukt datatriangulering gjennom å benytte både dokumentstudie og intervju som metoder for å skaffe datamateriale til oppgaven.

Innen fleksible design er det tre ulike forskningsstrategier som har vært vanlige å benytte seg av: Casestudy, etnografiske studier og grounded theory (Robson, 2011). De siste tiårene er det observert en vekst i antall kvalitative studier som ikke har posisjonert seg innen disse vanlige kvalitative forskningsstrategiene, og der metoden ikke har et ledende filosofisk perspektiv (Caelli, Ray & Mill, 2003). Denne tilnærmingen kalles generisk kvalitativ metode (GKM), og Merriam (sitert i Caelli m.fl., 2003) er av den oppfatning at generiske kvalitative forskningsstudier er de studier som kjennetegner egenskapene til kvalitativ forskning, men fremfor å fokusere på kultur slik som etnografiske studier, eller oppbygging av teori slik som grounded theory, så er målet med GMK at «forskerne bare prøver å oppdage og forstå et fenomen, prosess, eller perspektivene og verdensbildene til de involvert». Gjennom GKM er man

som forsker interessert i selve erfaringene, holdningene, meningene og oppfatningene personer har omkring et tema, problemstilling eller opplevelse, istedenfor *hvordan* dette oppleves eller føles (Percy, Kostere & Kostere, 2015). Denne metoden passer, ifølge Caelli m.fl. (2003), for uerfarne forskere som f.eks. mastergradsstudenter som meg. De begrunner dette med at en masterstudent sjelden har tid til å utvikle en grundig forståelse av kvalitative metodiske tilnærminger, men likevel ønsker å undersøke kvalitative forskningsspørsmål. Denne metoden passer også for mange forskere som har forskningsspørsmål som ikke passer inn i de tradisjonelle kvalitative forskningsstrategiene, f.eks. de studiene som omhandler folks holdninger og meninger om et emne. Mine forskningsspørsmål havner i denne kategorien, og jeg kommer derfor til å benytte meg av GKM som strategi for å besvare mine forskningsspørsmål.

4.2 Metoder for datainnsamling

Jeg har i denne oppgaven brukt to ulike metoder for datainnsamling: dokumentstudie og intervjuer. Innenfor fleksible design er det vanlig å kombinere metode for å kunne øke validiteten i studien (Robson, 2011). Det er intervjuer med fagpersoner som har erfaring med programmering eller vært med i utarbeidelsen av kjerneelementer eller læreplanarbeid innenfor matematikkfaget som er vektlagt mest i denne oppgaven som empiri. For å få forkunnskaper om innføringen av programmering i matematikkfaget, har det også vært nødvendig å studere relevant litteratur og ulike dokumenter som bl.a. stortingsmeldinger og høringssvar.

4.2.1 Dokumentstudie

Bruk av dokumenter som datamateriale er sentralt i de fleste forskningsdokumenter og blir ofte benyttet som bakgrunnsdata i tillegg til data fra f.eks. intervjuer eller observasjoner (Tjora, 2017). Tjora (2017) understreker at dokumenter kan bidra til å styrke forskerens historiske følsomhet, og at gjennom dokumenter kan forskeren danne seg et tidsbilde, noe som har betydning for hvordan nåtidige fenomener kan forståes i sammenheng med den historiske utviklingen. Et slikt tidsbilde har jeg forsøkt å danne meg gjennom kapittel 2.4. Det er vanlig å ha et dokumentstudie i forkant, for å kunne bruke litteraturen til å danne grunnlaget for videre forskning (Robson, 2011). Siden jeg ikke hadde noe erfaring med programmering fra før, så har tidligere forskning på programmering i skolen, i tillegg til utredninger og stortingsmeldinger vært viktige for å danne et bilde av hvorfor akkurat programmering ble omdiskutert i høringsrundene for både kjerneelementer og læreplanutkast. For å finne frem til aktuelle dokumenter har jeg hovedsakelig brukt internett. I dag er det lett å finne det meste av offentlige dokumenter, og jeg har hovedsakelig brukt utdanningsdirektoratet sine internettsider for å finne frem til høringer, høringssvar og innspill, samt utredninger og andre dokumenter. Siden flere av utredningene og stortingsmeldingene er svært omfangsrike, har jeg brukt søkeverktøy og søkt etter ord som «programmering» eller «algoritmisk tenkning» i dokumentene. For å finne forskningsartikler eller annen litteratur om programmering har jeg brukt bl.a. Google Scholar da den gir meg tilgang til publiserte artikler, bøker, rapporter og andre dokumenter. Jeg har vært nøye med å se om dokumentene jeg har funnet er publisert i kjente tidsskrift for å stadfeste at dette er gode kilder som jeg kan bruke. På denne måten har jeg funnet nye og gamle forskningsartikler om programmering, fra både Norge og utlandet som jeg har brukt som litteratur i denne oppgaven.

Siden hovedmålet med oppgaven er å se hvilke muligheter og utfordringer programmering kan gi for matematikkfaget har det vært nødvendig å belyse litteratur og nyere teori innenfor

matematikkdidaktikk. Robson (2011) påpeker at å systematisk identifisere, finne og analysere dokumenter som inneholder informasjon som er relevant er en viktig del av en litteraturstudie, og at litteraturen som velges bør ha betydning for studiens design eller tolkningen av studiens tema. En gjennomgang av høringssvar fra matematiske fagmiljø og oppsummeringer laget av kjerneelementgruppa var med på å finne en del av argumentene som ble brukt i debatten rundt programmering. Dokumentstudien i sin helhet har vært viktig for å utarbeide intervjuguiden og temaene som jeg ønsket å snakke om med informantene.

4.2.2 Intervju

Den mest utbredte datagenereringsmetoden innenfor kvalitativ forskning er ulike former for intervju, særlig populære er såkalte *semistrukturerte intervjuer* eller *dybdeintervjuer*[7] (Tjora, 2017, s. 113). Tjora (2017) kaller semistrukturerte intervjuer for dybdeintervjuer der målet er å skape en situasjon for en relativt fri samtale som kretser rundt noen spesifikke temaer som forskeren har bestemt på forhånd. Meningen er å få informanten til å reflektere over egne erfaringer og meninger knyttet til de utvalgte temaene. Intervjuer kombineres ofte med andre metoder, og jeg har basert intervjuene mine på dokumentstudiet som ble gjort i forkant. Utfra dokumentstudiet utarbeidet jeg en intervjuguide (Vedlegg A) for lettere å kunne strukturere intervjuet og være sikker på at jeg kom gjennom alle de temaene jeg ønsket i løpet av intervjuet. Med intervju som metode ønsket jeg å få en forståelse av informantenes tanker, opplevelser og meninger om bruken programmering som verktøy i undervisning, spesielt i matematikkfaget. Ved å velge dybdeintervju hadde jeg mulighet til å være fleksibel og kunne respondere på det som ble sagt under intervjuet, selv om samtalen var forankret i utvalgte temaer i forkant. Tjora (2017) understreker at dybdeintervjuer utelukkende kan gi kunnskaper om forhold knyttet til informantens subjektivitet, og at kvaliteten på intervjuet avhenger av tillit mellom forsker og informant noe jeg kommer tilbake til i kapittel 4.2.4.

4.2.3 Valg av informanter

Gjennom strategisk utvalg bør man i kvalitative intervju velge informanter som av ulike grunner vil kunne uttale seg på en reflektert måte rundt temaet (Tjora, 2017). Kvale og Brinkmann (2014) fremhever at antall informanter må bestemmes ut fra hvordan man best får svar på forskningsspørsmålene. På grunn av oppgavens størrelse og omfang hadde jeg i utgangspunktet tenkt å ha med 4-6 informanter, dette begrunnes med at å ta med flere informanter ville blitt vanskelig å gjennomføre, da alle intervjuene skal både gjennomføres, transkriberes og analyseres, noe som i utgangspunktet er en tidkrevende prosess.. I tillegg til intervjuene hadde jeg også et dokumentstudie som krevde tid.

Målet med å finne informanter var at de skulle kunne gi mye og god informasjon omkring temaet jeg ønsker å belyse, og at de skal kunne gi troverdighet til oppgaven. Innenfor GKM er det mindre viktig å gå «i dybden», men det er ønskelig å få et bredt spekter av meninger, erfaringer og refleksjoner, og derfor brukes det ofte et større utvalg av informanter eller kilder. Likevel vil et lite, velvalgt utvalg av informanter i mange tilfeller være nok, så fremst disse informantene er representative (Percy m.fl., 2015). Derfor ønsket jeg at informantene som skulle intervjues hadde god erfaring med matematikkfaget og/eller programmering, og at de kunne uttale seg om programmering, matematikk og fagfornyelsen.

[7] I denne oppgaven velger jeg å bruke betegnelsen dybdeintervjuer

Den første informanten jeg fant som aktuell for min oppgave, var Tom Louis Lindstrøm var lederen for kjerneelementgruppa i matematikk. Han hadde jeg lest om, og sett foredrag av, vedrørende de nye kjerneelementene til fagfornyelsen. Lindstrøm er professor ved Det matematisk-naturvitenskapelige fakultet ved Universitetet i Oslo (UiO). Han underviser i matematikk på universitetsnivå og har skrevet flere bøker og artikler innen matematikk.

I min videre søken etter informanter kom jeg raskt over UiO sine nettsider, som bl.a. tilbyr kurset ProFag, som er en forkortelse for programmering for fagenes skyld, og er et kurs i programmering tilpasset lærere fra 8.–13. trinn. En av kursholderne der er Cathrine Wahlstrøm Tellefsen som er førstelektor ved det Det matematisk-naturvitenskapelige fakultet ved UiO. Hun har vært lærebokforfatter i fysikk og naturfag i mange år og har lang undervisningserfaring fra videregående skole. Wahlstrøm er for tiden utdanningsleder for lektorprogrammet i realfag og en del av ledergruppen for CCSE (Center for Computing Science Education) ved Universitetet i Oslo.

Den neste informanten kom jeg over på jobb. Siden programmering er «i vinden» hadde vi nylig fått en ny lærebok på skolen jeg er ansatt ved, for å oppdatere oss på programmering. Boka er skrevet av Andreas Drolsum Haraldsrud og er tiltenkt det nye programfaget programmering og modellering. Haraldsrud har også vært med på å utarbeide kompetansemålene til dette programfaget. Han har erfaring fra videregående skole innen kjemi, fysikk, matematikk og nå er han universitetslektor ved det Det matematisk-naturvitenskapelige fakultet ved Universitetet i Oslo. Der samarbeider han med Cathrine Tellefsen på ProFag-kursene og er en av representantene for CCSE.

Jeg ønsket også å intervjue en person som har mye erfaring med programmering og kan se dette i sammenheng med matematikk. Ved å benytte med av mulighetene internett byr på kom jeg over hjemmesiden til Børre Steinar Stenseth, som er en pensjonert førsteamanuensis i informatikk ved Høgskolen i Østfold. Han har over 50 års erfaring med programmering, og har vært med på utviklingen av bruk av informasjonsteknologi i undervisningen ved å produsere bl.a. læremidler og bøker. De siste årene har han samarbeidet med fagpersoner i matematikk og skrevet artikler om bruk av programmering i matematikkfaget.

Jeg ønsket også å intervjue en fagperson som hadde erfaring med fagfornyelsen og som deltok i utarbeidelsen av de nye læreplanene i matematikkfaget, og ble tipset om Tor Espen Kristensen. Han var også med i Sanneutvalget som skrev rapporten *Teknologi og programmering for alle*, og har vært en del av kjerneelementgruppa i matematikk. Nå er han med i læreplangruppa for matematikk, i tillegg er han også en ressursperson for matematikksenteret og LAMIS (landslaget for matematikk i skolen). Han jobber ved Stord videregående skole og underviser i fagene matematikk og fysikk.

Forespørsel til alle informantene ble sendt på mail med informasjon om oppgaven og forskningsspørsmålene (Vedlegg B). Alle fem var svært positive til å delta i undersøkelsen min, og sa ja til å delta.

4.2.4 Gjennomføring av intervjuene

Etter gjennomgang av intervjuguiden (Vedlegg A) med min veileder og etter å ha fått godkjent søknaden min hos NSD (Vedlegg D) kunne jeg avtale tidspunkt for videomøtene. Jeg hadde

dessverre ikke mulighet å reise for å intervjue informantene personlig, og av den grunn måtte alle intervjuene gjennomføres over video. Alle intervjuene ble holdt individuelt, og jeg valgte å bruke min egen videomøteplattform som jeg har gjennom min arbeidsgiver, da informantene kun trenger å trykke på en link og koble seg på, noe som fungerte utmerket. Informantene fikk selv bestemme tidspunkt for møtet, og selvfølgelig hvor de ville gjennomføre intervjuet da alt de behøvde var en PC med lyd og kamera. Ifølge Tjora (2017) er det en fordel å overlate valget av sted for intervjuene til informantene selv. Alle informantene ble gjort kjent med at det ble gjort lydopptak under videomøtet, og for å sikre at jeg fikk brukbart opptak brukte jeg to ulike lydopptakere. Det ble satt av en time til hvert intervju og alle intervjuene varte mellom 40 til 60 minutter. Alt fungerte godt og alle intervjuene ble gjennomført uten store utfordringer, sett også fra den tekniske siden Intervjuene startet med en orientering om oppgavens formål. Alle informantene hadde fått informasjonsskrivet på forhånd, og hadde få spørsmål angående opplegget. Etter innledningen startet hoveddelen av intervjuet der intervjuguiden utgjorde den røde tråden. Etter gjennomførte intervjuer skulle det empiriske materialet omgjøres til tekstformat gjennom transkribering. I kapittel 4.3.1 forklarer jeg denne prosessen mer detaljert.

4.3 Analyse

Innenfor GKM er det vanlig å benytte tematisk analyse (TA) som en prosess for å analysere kvalitativ data (Percy m.fl., 2015). TA blir av Braun og Clarke (2006) sett på som en av de mest brukte analysemetodene innen kvalitativ forskning og metoden legger vekt på å understreke, utforske og registrere mønstre i datamaterialet. Metoden betraktes som brukervennlig og fleksibel, og blir ofte benyttet av uerfarne forskere. Braun og Clarke (2006) deler hovedsakelig opp TA i to ulike tilnærminger, den induktive og den teoretiske analyse. Gjennom en induktiv analyse blir temaene som indentifiseres gjennom analysen drevet av datamaterialet i seg selv, og forskerens forforståelse og kunnskap skal i minst mulig grad påvirke kodingen. I teoretisk analyse drives kodingen av forhåndsdefinerte temaer som forskeren har identifisert, og kodingen drives av et eller flere faste forskningsspørsmål rettet mot et tema (Braun & Clarke, 2006). Denne oppgaven dreier seg mot en induktiv tilnærming selv om jeg gjennom dokumentstudien dannet meg en formening på forhånd om noen av temaene eller argumentene som ville dukke opp. Dokumentstudiet dannet grunnlaget for intervjuguiden, der spørsmål om bl.a. usikkerheten rundt begrepet algoritmisk tenkning og diskusjonen om at programmering vil føre til stofftrengsel er med i guiden. Oppgaven er likevel induktiv i den forstand at jeg ikke brukte forhåndsdefinerte temaer som grunnlag for analysen, men lot datamaterialet være det som driver kodingen og kategoriseringen.

4.3.1 Analyseprosessen

I dette delkapittelet vil jeg vise til fremgangsmåter jeg har benyttet meg av under analyseprosessen og hvilke beslutninger som ble tatt underveis. Braun og Clarke (2006) har utviklet en fremgangsmåte til TA som kan brukes uavhengig av tilnærming. Denne fremgangsmåten har Johannessen (2018) laget en forenklet versjon av med fire steg. Versjonen er fleksibel og mer «studentvennlig» enn Braun og Clarkes versjon, og er et godt alternativ for nybegynnere innen forskning (Johannessen, 2018). Versjonen fungerer som et rammeverk i analyseprosessen, og jeg har forsøkt å følge denne fremgangsmåten Her blir det viktig å påpeke at selv om de fire trinnene ser ut som en lineær prosess, er den i praksis iterativ og forskeren beveger seg mellom de ulike stegene i løpet av hele analyseprosessen.

Steg 1: Forberedelse

Det første steget i TA er å gjøre seg kjent med datamaterialet. Denne prosessen starter umiddelbart etter innhenting av datamaterialet, og innebærer at forskeren må skifte fokus fra datagenerering til analyse. Dette steget inkluderer å høre på lydopptakene og selve transkriberingsarbeidet (Braun, Clarke, Hayfield & Terry, 2019). Alle intervjuene ble tatt opp på lydbånd og jeg hørte gjennom alle intervjuene før jeg startet transkriberingsarbeidet. Intervjuene ble fullstendig transkribert i etterkant, dette anbefales av Tjora (2017) for ikke å miste noe informasjon. Det anbefales også å være mer detaljert enn man tror er nødvending. Gjennom transkripsjonen av intervju er målet å gjøre de muntlige fortellingene mer egnet for analyse (Kvale & Brinkmann, 2015). Jeg holdt et relativt høyt detaljnivå på transkriberingen, noe jeg valgte å gjøre for å ha mest mulig datamateriale til analyseprosessen. Ifølge Tjora (2017) er det viktigste tapet fra selve intervjuet tapet av visuelle ledetråder og informasjon om stemningen i løpet av intervjuet. For å unngå å miste mye informasjon under «oversettelsen» av intervjuet til det skriftlige datamaterialet er det en stor fordel at intervjueren selv transkriberer opptakene, en anbefaling jeg valgte å følge.

Steg 2: Koding

Etter å ha brukt tid til å bli kjent med datamaterialet og på transkribering av intervjuene, startet det viktige arbeidet med den første kodingen av datamaterialet. Dette steget handler om å sette ord på, eller fremheve viktige poeng i dataene. Braun m.fl. (2019) understreker at denne delen av analyseprosessen krever at forskeren er fokusert og går systematisk og grundig gjennom datamaterialet. Koding kan utføres manuelt, ofte med hjelp av fargekoder, eller ved bruk av dataprogrammer. Jeg valgte å benytte Nvivo som analyseverktøy for å utføre kodingsprosessen. I Nvivo knytter man relevante sitater til «noder», som fungere på samme måte som vanlig koding, man navngir hver enkelt «node» med en passende kode som er relatert til sitatet. Hvis noen sitater handler om det samme, benyttes samme kode. Etter den første kodingsprosessen satt jeg igjen med 77 ulike noder. Tabell 3 viser eksempel på noden «algoritmisk tenkning – uklart begrep» og sitater knyttet til dette.

Node	Sitat
algoritmisk tenking - uklart begrep	Alle som skriver om dette skriver jo forskjellige ting, noe drar det jo helt ut og andre går i alle mulige detaljer og bruker det til hva som
	Jeg føler at tilhengere av AT trekker det begrepet veldig lang, jeg synes at veldig mange trekker det så langt at det egentlig blir snakk om matematisk tenking, eller matematisk problemløsning.
	Det er liksom mer enn bare den AT, men AT er et fint begrep siden det er kjent, og det kjenner vi fra matematikk, så de handler kanskje litt om å bruke en retorikk som gjør at vi tør å gå inn i det.
	Men hvis du egentlig skal ta inn over deg det som skjer så handler det egentlig om computational literacy, og det er mer enn AT.
	Synes egentlig ikke at algoritmisk tenkning er et godt begrep, skulle egentlig ønske at vi brukte computational literacy, som er mitt favorittuttrykk om dagen.
	det er et veldig generelt begrep, har ikke så mye med algortimer å gjøre egentlig..
	Det burde forklares litt dypere hva som legges i begrepet, og det burde påpekes hvor essensielt det er innenfor programmering. Gjerne få inn definisjonen av en algoritme. Ellers kan begrepet lett misforstås, det er fare for at flere (lærere) tenker på «oppskriftstenkning» når de leser dette ordet.

Tabell 3: Noder og sitater

Steg 3: Kategorisering

Etter å ha laget koder må dataene sorteres og man begynner arbeidet med å lage overordnede kategorier (Johannessen, 2018). Kategoriene bygges, lages og gis mening gjennom å knytte sammen data, forskerens erfaringer og subjektivitet og forskningsspørsmålene (Braun m.fl., 2019). I starten laget jeg litt generelle kategorier:

- Fremtidens kompetansebehov
- Utvikling av læreplaner

- Tilgang på digital teknologi
- Stofftrengsel
- Programmering som eget fag eller ikke?
- Digital kompetanse
- Utvikling av digital teknologi
- Algoritmisk tenkning (AT) – uklart begrep
- Hva menes med algoritmisk tenkning?
- Problemløsning, AT og programmering
- Programmering er tidkrevende
- Programmering og dybdelæring i matematikk
- Digitale verktøy
- Feilretting
- Motivasjon
- Samarbeid
- Utholdenhet
- Lærerkompetanse
- Eksamen

Etter en mer grundig gjennomgang av kategoriene fant jeg ut at noen av disse omhandler det samme og kan samles under ett. Etter gjennomgangen endte jeg opp med disse kategoriene for forskningsspørsmål 1: *Hvilke argumenter kan brukes for eller imot innføringen av programmering i matematikkfaget?*

Tema	Undertema
Fremtidens kompetansebehov	Fremtidens kompetansebehov Utviklingen av digital teknologi Tilgang til digital teknologi Utvikling av læreplaner
Stofftrengsel	Programmering som eget fag eller ikke Programmering fører til stofftrengsel i matematikk Programmering er tidkrevende
Dybdelæring	Programmering kan bidra til økt dybdelæring Programmering gir andre innfallsvinkler til emner Programmering kan øke matematikkompetansen hos elever Programmering kan hindre dybdelæring
Algoritmisk tenkning	AT – uklart begrep Hvordan defineres algoritmisk tenkning Problemløsning, AT og programmering
Digitale verktøy	For mange digitale verktøy? Feilretting som prosess
Sosiale og emosjonelle kompetanser	Motivasjon Samarbeid Utholdenhet

Tabell 4: Kategorier for forskningsspørsmål 1

Disse kategoriene ble dannet for forskningsspørsmål 2: *Hvilke forutsetninger er viktige for å lykkes med implementeringen av programmering i matematikk?*

Tema	Undertema
Lærerkompetanse	Lærerkompetansen for lav Læreren må se relevansen til sitt fag Etter- og videreutdanning
Eksamen	Programmering må testes på eksamen Hvordan teste dette på eksamen

Tabell 5: Kategorier for forskningsspørsmål 2

Steg 4: Rapportering

I dette steget skal forskeren rapportere kategoriene som ble laget i steg 3, og skrive om disse for å få svar på forskningsspørsmålene (Johannessen, 2018). Hensikten med rapportering er å gjøre det forståelig og oversiktlig for leseren, og jeg har valgt å dele opp rapporteringsdelen i to. I kapittel 5 presenterer jeg de resultater jeg har funnet som hører til de ulike kategoriene. I kapittel 6 diskuteres funnene fra kapittel 5 i lys av relevant teori.

Oppsummering av analyseprosessen

Analyseprosessen som beskrevet over har ikke vært en lineær prosess, selv om det presenteres gjennom fire steg. Analyseprosessen har for meg foregått gjennom hele skriveprosessen, fra start til slutt, og gjennom prosessen har jeg har flere ganger beveget meg mellom stegene. I arbeidet med denne oppgaven har jeg intervjuet fem informanter og gjort en omfattende dokumentstudie i forkant for å blant annet å forberede intervjuguide og få et overblikk av debatten. Fremgangsmåter brukt av meg til analysen av datamaterialet opplevde jeg til å ha fungert bra, og at jeg ved å bruke rammeverket til Johannessen (2018) har klart å få frem gode kategorier.

4.4 Etiske hensyn

Det er viktig å være klar over de etiske hensynene man må ta i enhver forskning, spesielt når mennesker er deltakere. Enhver forsker er forpliktet til å følge forskningsetisk lov[8] som har som formål at all forskning skjer i henhold til anerkjente forskningsetiske normer. I mitt arbeid har jeg gjort etiske refleksjoner rundt planleggingen av intervjuene. Som første steg måtte jeg sende meldeskjema inn til NSD (Norsk senter for forskningsdata), se vedlegg C. NSD er personvernombudet for forskning og siden jeg skal samle inn data med personopplysninger så er jeg pliktig til å melde fra om studiet. Godkjenning fra NSD til å gjennomføre forskningsprosjektet og kvittering fra NSD ligger som vedlegg D.

I arbeidet med denne oppgaven er det gjennom arbeidet med dybdeintervjuene at jeg må være svært bevisst hvilke hensyn jeg må ta. Tjora (2017) understreker at i forbindelse med gjennomføringen av intervju er forskningsetikken først og fremst knyttet til kravet om at informanten ikke skal komme til skade, dette gjelder spesielt når det forskes på områder som kan oppleves ubehagelige eller vanskelige. Det er alltid intervjuerens ansvar å begrense følelsesmessige problemer som følge av intervjuet (Tjora, 2017). Siden mine informanter ikke er

[8] For selve loven, se: https://lovdata.no/dokument/NL/lov/2017-04-28-23

anonyme, har det vært spesielt viktig å informere om rettighetene ved å delta i denne prosjektet. Jeg ønsket at informantene skulle være kjente for å øke validiteten i oppgaven, noe jeg begrunner med informantenes faglige tyngde og samlede bredde innenfor emnet jeg forsker på. De fikk alle tilsendt informasjon på mail i forbindelse med forespørselen om deltakelse (Vedlegg B), i tillegg informerte jeg om dette i starten av intervjuene. Informantene har hatt mulighet til å trekke seg fra prosjektet når som helst, og alt av opplysninger som lydopptak og tekstfiler vil da slettes. Informantene ble tilsendt en sitatsjekk der de ble kjent med de sitatene jeg kom til å bruke i oppgaven, dette for å gi informantene mulighet til å fjerne det som eventuelt hver enkel av dem ønsket å fjerne. Slik mener jeg at jeg har ivaretatt informantene på en god måte gjennom arbeidet med denne oppgaven.

4.5 Oppgavens kvalitet

Forskning handler alltid om å skape troverdige resultater, det vil si godt begrunnede tolkninger gjort ved systematiske analyser av empirisk data, som igjen er generert ved en nøye gjennomtenkt design (Tjora, 2017, s. 253). Ifølge Tjora (2017) benyttes ofte *pålitelighet*, *gyldighet* og *generaliserbarhet* som indikatorer og kvaliteten knyttet til forskningen. Han viser også til at begrepene troverdighet, bekreftbarhet og overførbarhet har vært brukt en del innenfor kvalitativ forskning, men mener at disse begrepene er lite hensiktsmessige og at de første indikatorbegrepene fungerer utmerket godt også innen kvalitativ forskning.

Pålitelighet handler om kvaliteten på datamaterialet og på det arbeidet forskeren har gjort. Det er viktig at forskeren dokumenterer hele prosessen. Tjora (2017) påpeker at det skal være en klar sammenheng mellom empiri, analyse og resultater i en undersøkelse. Metodekapitlet er viktig for å øke påliteligheten for denne oppgaven, siden jeg i dette kapitelet redegjør for alle de valgene jeg har tatt underveis og forklarer i hvilke metoder jeg har brukt både til innsamling av data og til analyseprosessen. Jeg har videre brukt direkte sitater i kapittel 5 for å gjøre informantenes «stemme» synlig, noe som ifølge Tjora (2017) er med på å øke påliteligheten. Jeg har også begrunnet mine valg av informanter, og forklart hvordan jeg gikk frem i utvelgelsen av disse.

Gyldighet handler om en logisk sammenheng mellom prosjektets utforming og funn, og de spørsmålene man søker å få svar på (Tjora, 2017). Det trekkes paralleller mellom gyldighet og validitet, og begge disse går inn på om metodene som er brukt i forskningen er egnet til å svare på det man skal undersøke. Siden tilnærmingen innen kvalitativ forskning somregel er fortolkende, kan dette være relativt komplisert (Tjora, 2017). I kvalitative forskningsintervjuer er forskerens kompetanse og egnethet avgjørende for data som skapes (Drageset & Ellingsen, 2010). Drageset og Ellingsen (2010) skriver videre om gyldighet i kvalitativ forskning slik:

> Forskerens kompetanse refererer til den håndverksmessige kvaliteten, det vil si hvordan man har utført intervjuet, gjort nødvendige notater underveis, hvordan man har transkribert og dokumentert, begrunnet og redegjort for hva man har gjort. Uten dette er det ingen gyldighet i kvalitativ forskning.

Ved å redegjøre for de valg jeg har tatt når det kommer til f.eks. valg av datagenereringsmetoder og valg av rammeverk for analyse inviterer jeg leseren til å ta stilling til forskningens gyldighet. Blant annet kan datatriangulering bidra til økt validitet, noe jeg har benyttet med av i denne oppgaven. Ved å gjennomføre en informantvalidering gjennom å utføre en sitatsjekk, kan jeg

være også sikker på at sitatene fra informantene er kvalitetssikret og at de kan stå inne for de utsagnene jeg bruker i oppgaven.

Generaliserbarhet handler om at funnene har gyldighet utover utvalget og konteksten til forskningsprosjektet, og er relevant og anvendbart i andre situasjoner (Drageset & Ellingsen, 2010). Tjora (2017) skiller mellom naturalistisk, moderat og konseptuell generalisering. Naturalistisk generalisering er ikke aktuelt for min oppgave og jeg velger derfor å ikke utdype dette her. Moderat generalisering kan ses på i mer strukturell forstand der forskeren beskriver i hvilke situasjoner resultatene vil kunne være gyldige. Dette handler blant annet om at funnene knyttes opp til settingen i forskningsprosjektet og at generaliseringen muligens bør begrenses til bestemte tidsperioder. For min oppgave vil valget av informanter kunne være med på å øke generaliserbarheten. Ved å gjøre et strategisk utvalg og velge informanter som jeg mener har god og relevant informasjon vil funnene kunne ha gyldighet utover forskningsprosjektet. Debatten rundt programmering var stor i starten av dette prosjektet, med mange avisartikler og høringsinnspill. Nå som kjerneelementene og læreplanen er fastsatte er debatten stilnet av, og om noen år vil lærere og andre i skolen ha fått testet ut programmering i matematikkfaget og da er det ikke sikkert at de argumentene som blir belyst i denne oppgaven vil være gyldige lengere. Konseptuell generalisering handler om å framstille funn i form av konsepter eller teori. Dette forskningsprosjektet hadde ikke som formål å utarbeide nye konsepter eller teori, men jeg ønsket å belyse et omdiskutert område innenfor skoleverket. Ved å bruke dokumentstudie og dybdeintervju har jeg kommet frem til en del overordnede kategorier der funnene omhandler både argumenter som kan brukes for eller imot programmeringen, og hvilke forutsetninger som er viktige for å lykkes med implementeringen.

En faktor som også er med på å avgjøre kvaliteten til en oppgave er knyttet til begrepet transparens eller gjennomsiktighet. Mens pålitelighet og gyldighet reflekterer hvor godt valgene i et forskningsprosjekt er begrunnet, handler transparens om hvor godt disse valgene er formidlet i rapporteringen av forskningen (Tjora, 2017). I kapittel 4 forsøker jeg å begrunne alle de metodiske valgene jeg har gjort og i kapittel 5 forsøker jeg å presentere funnene jeg har kommet frem til på den detaljert og oversiktlig måte. Slik ønsker jeg å gi leseren et best mulig innblikk i forskningen slik at de selv kan ta stilling til kvaliteten i oppgaven.

4.6 Metodediskusjon

I forrige delkapittelet forsøkte jeg å belyse oppgavens kvalitet, gjennom begrepene pålitelighet, gyldighet og overførbarhet. Dette forskningsprosjektet baserer seg på generisk kvalitativ metode som i seg selv er en metode som ikke er en av de etablerte kvalitative forskningsmetodene, og har av den grunn fått en del kritikk (Caelli m. fl., 2003). Denne metoden er likevel mye brukt, spesielt av nybegynnere innen forskning. Ved å bruke denne metoden kunne jeg fokusere mer på selve forskningsprosessen og ikke bruke mye tid på å sette meg inn i de mange forskningsfilosofiske tilnærmingene. Denne metoden passet også godt til forskningsspørsmålene mine og ved å benytte meg av tematisk analyse i analyseprosessen kom jeg frem til funn som kunne besvare disse.

Funnene baserer seg på dybdeintervju av fem informanter. Med tanke på studiens pålitelighet bør forskeren legge vekt på at det ikke stilles ledende spørsmål, som kan påvirke informantenes svar. Jeg har ikke gjennomført dybdeintervjuer tidligere, noe som kan påvirke kvaliteten på intervjuet. Selv om jeg forsøkte å være bevisst ledende spørsmål, hørte jeg under

transkriberingsarbeidet at jeg i noen tilfeller stilte spørsmål som nok kan klassifiseres som ledende. Intervjuene ble gjennomført over video, og jeg gikk på den måten glipp av det personlige møtet med informantene. Jeg følte ikke at dette hadde stor påvirkning på kvaliteten på intervjuet, bortsett fra at det var vanskelig å se kroppsspråket til informantene. I min oppgave vil nok ikke kroppsspråket kunne gitt så mye mer informasjon enn språket i seg selv, da jeg var ute etter informantenes erfaringer og meninger, og ikke følelser.

Bruk av lydopptak og transkriberingen gjorde at dataene fra intervjuene ble nøyaktige og korrekte, i forhold til bruk av notater. I denne oppgaven har mesteparten av dataene som presenteres kommet fra de transkriberte intervjuene og er i den forstand objektive, men gjennom analyseprosessen og tolkningen vil sitatene kunne blitt subjektivisert av meg.

Begrepet *metning* brukes innen kvalitativ forskning og handler om hvor stort utvalget i en studie bør være for at funnene skal være gyldige (Tjora, 2017). Dette går ut på at når ikke nye funn fremkommer er analysen «mettet», og ingen ny informasjon vil oppstå. For min oppgave er det ikke gitt at studien nådde sitt metningspunkt, da oppgavens omfang gjorde det vanskelig å utvide utvalget mitt for å undersøke om jeg kunne kommet frem til flere argumenter. Likevel mener jeg at jeg ved å velge informanter med så stor faglig tyngde og bredde, klarer å få frem de argumentene som kan benyttes i debatten rundt programmering. Min bedømming er at jeg ikke hadde kommet frem til store endringer i de funnene jeg endte opp med i kapittel 5, ved å utvide utvalget med noen informanter.

5 Funn

I dette kapitlet vil jeg først vise til deler av dokumentstudiet der programmering ble funnet nevnt, og hvordan jeg gjennom disse dokumentene har kommet frem til beslutninger som kan ha ført til at programmering nå blir innført som del av læreplanen. Videre vil jeg gå igjennom hvilke argumenter som er blitt trukket frem for eller imot innføringen av programmering. Jeg vil hovedsakelig bruke informantenes utsagn, men også henvise til høringsinnspill og media. Siste del av dette kapitlet vil belyse hvilke forutsetninger informantene tror er viktige for at implementeringen av programmering skal lykkes i skolen.

5.1 Prosessen mot programmering som del av matematikkfaget

Arbeidet med å fornye læreplanen i norsk skole er en lang prosess. I matematikkfaget, og andre fag, har det blitt gjort mange ulike utredninger på flere områder, blant annet med fokus på det den digitale tilstanden i skolen. Offentlige utredninger (NOU) og stortingsmeldinger har lagt rammer for hvordan den nye læreplanen skal se ut, og det har vært utarbeidet flere utkast til både kjerneelementer og læreplaner. Til hvert av utkastene til både kjerneelementer og læreplan har alle vært oppfordret til å komme med høringsinnspill, og mange fagmiljø, skoler og andre instanser har tatt oppfordringen. I dette delkapittelet vil jeg vise relevante funn vedrørende innføringen av programmering som jeg har funnet i disse dokumentene.

5.1.1 Programmering i offentlige dokumenter

Hvis man følger ideen om programmering i skolen gjennom offentlige dokumenter nevnes programmering i liten grad:

Difiutvalgets rapport (NOU 2013: 2) *Hindre for digital verdiskaping* mener at manglende oppmerksomhet på grunnleggende programmerings- og utviklingsferdigheter gjør at vi utdanner barn og unge til å kunne bruke digitale verktøy, men at det fokuseres for lite på hvordan disse verktøyene fungerer og hvordan bruke disse verktøyene til å produsere og skape. De er bekymret for den digitale kompetansen hos elevene og i rapporten foreslås det at timetallet til matematikk økes, og at man i tillegg bør sørge for at noe grunnleggende programmering kan læres allerede i ungdomsskolen gjennom å tilby dette som valgfag. Utvalget mener det er for lenge å vente til at elevene kan velge formell undervisning i programmering først på videregående nivå. Myndighetene oppfordres i denne rapporten til å ta programmering inn i læreplanen, og henviser til utviklingen i andre land (NOU, 2013: 2).

Borgeutvalgets rapport *Matematikk i norsk skole anno 2014* (Borge m.fl., 2014) skulle gi en rapport som skulle gi beslutningstakere et kunnskapsgrunnlag om matematikktilbudet og som peker på muligheter og utfordringer når det gjelder matematikktilbudet i grunnopplæringen. Programmering nevnes ikke i denne rapporten.

I Ludvigsenutvalget rapport (NOU 2015: 8) *Elevers læring i fremtiden skole*, nevnes ikke programmering i det hele tatt. Denne rapporten legger store føringer for stortingsmelding nr. 28 (2015-2016), som er en del av grunnlaget for fagfornyelsen.

Lysneutvalgets rapport (NOU 2015: 13) viser til Difiutvalgets rapport og støtter forslag om valgfag i programmering i ungdomsskolen, og at det i realfagsstrategien *Tett på realfag* skal lanseres et prøveprosjekt med tilbud om valgfag i programmering ved 20 klasser fra skoleåret

2016/2017. Denne rapporten begrunner opplæring i programmering i skolen med at programmering vil kunne fungere som et støtteverktøy i andre fag, og samtidig bidra til utvikling av analytisk tenkning og problemløsning, og stimulere til kreativitet og gruppearbeid (NOU, 2015: 13).

I Stortingsmelding nr. 28 (2015-2016) nevnes programmering to ganger. En gang i rapporten henviser de til Lysneutvalgets ønske om at tilbudet i programmering bygges ut, og at det starter opp prøveprosjekt med valgfag i programmering på ungdomsskolene.

Sanneutvalget startet sitt arbeid i begynnelsen av 2016 og hadde som mandat å foreta en faggjennomgang av teknologi i grunnopplæringen. Gruppa skulle levere en rapport som gir beslutningstakere et kunnskapsgrunnlag om hvilken kompetanse fremtidens elever skal ha innenfor teknologi og teknologirelaterte emner. I dette ligger digitale ferdigheter og programmering (Sanne m.fl., 2016). Arbeidsgruppa skulle bl.a.

- Belyse begrepet teknologi, inkludert programmering
- Sammenligne tilbudet av teknologirelaterte fag, inkludert programmering, i grunnopplæringen i Norge med tilbudet i noen relevante land (fellesfagene)
- Beskrive og vurdere hva digital og teknologisk kompetanse innebærer i det enkelte realfag. I dette inngår vurderinger av om programmering bør inngå som en del av opplæringen i realfagene (Sanne m.fl., 2016, s. 3).

Rapportens konklusjon er svært tydelig når det kommer til programmering, og mener teknologi og programmering trenger dedikerte timer hvis det skal bli vellykket og at det bør opprettes et nytt fag til dette:

> Arbeidsgruppa foreslår at det opprettes et nytt obligatorisk fag i grunnskolen. Faget skal omfatte teknologi og programmering. Det skal være et praktisk fag hvor eleven får mulighet til å tilegne seg grunnleggende, teknologisk kompetanse (Sanne m.fl., 2016, s. 76).

I 2017 lanserte Kunnskapsdepartementet sin digitaliseringsstrategi *Framtid, fag og digitalisering*. Her kommer det frem at i fagfornyelsen skal det vurderes hvordan teknologi, programmering og algoritmisk tenkemåte kan inngå i bestemte læreplaner for fag, særlig i matematikk og naturfag. Det er et mål at alle elever skal få kjennskap til hvordan teknologi og ulike programmer fungerer og spiller sammen gjennom opplæringen (Kunnskapsdepartementet, 2017. s. 18). Strategien viser til at erfaringer med programmering som valgfag skal evalueres i 2019 og at videre anbefalinger skal oppsummeres. Alle skoler som ønsker å tilby programmering som valgfag fra høsten 2019 skal få mulighet til dette. Forsøk med programfaget programmering og modellering i videregående skal også videreføres (Kunnskapsdepartementet, 2017).

Stoltenbergutvalgets rapport (NOU 2019: 3) påpeker at elevmassen som velger valgfaget programmering for det meste er gutter. I skoleåret 2017–2018 ble valgfaget programmering innført, med 210 skoler som tilbyr faget. På disse skolene velger i overkant av 10 prosent av elevene programmering, og 93 prosent av elevene er gutter (NOU 2019: 3, s. 206). Ellers nevner heller ikke denne rapporten programmering utover dette.

For å oppsummere er det Difiutvalget, Lysneutvalget og digitaliseringsstrategien som legger føringer for at programmering skal inn i læreplanene, da litt på ulike premisser. Både Difiutvalget og Lysneutvalget legger opp til programmering som valgfag på ungdomsskolen, mens digitaliseringsstrategien legger opp til at programmering skal inngå i bestemte læreplaner, særlig i matematikk og naturfag, men da som vilkår at timetallet utvides. Dette er i strid med det Sanneutvalget konkluderte med året før, men det finnes ikke noen faglige begrunnelser i dokumentet for digitaliseringsstrategiens tiltak om å innføre programmering som del av læreplanen. I Innstilling nr. 19S (2016-2017) til Stortinget fra Kirke-, utdannings- og forskningskomiteen er det på et tidspunkt vedtatt i regjeringen at gjeldende fag- og timefordeling skal ligge til grunn i fagfornyelsen (Kirke-, utdannings- og forskningskomiteen, 2016)). På bakgrunn av dette vedtaket kan ikke Sanneutvalgets forslag til eget fag gjennomføres, men det har ikke kommet frem noen dokumenter som belyser hvorfor eller hvor disse beslutningene ble tatt, men man kan anta at det ligger politiske argumenter eller interesser bak. Å se på de politiske argumenter for denne beslutningen ligger utenfor denne oppgavens formål og vil derfor ikke belyst.

5.1.2 Kjerneelementer i matematikkfaget

Etter vedtak fra politisk hold blir programmering tatt inn i læreplanene ved å innføres som del av andre fag. Dette er slik det er gjort i Sverige og en del andre land. Etter denne bestemmelsen var det videre arbeidet med fagfornyelsen etter stortingsmelding nr. 28 (2015-2016), utarbeidelse av kjerneelementene i fagene. Disse ble fastsatt i 26. juni 2018, men arbeidet med dette begynte allerede våren 2017, og det var tre store høringsrunder før endelige kjerneelementer ble fastsatt. Kjerneelementgruppa hadde en måned fra første samling til den første skissen ble lansert den 6. september 2017. I denne skissen påpekes det at:

> Det er også bestemt at programmering skal inn i matematikkfaget uten at timetallet økes. Dette har gitt kjerneelementgruppen spesielle utfordringer. Selv om det er enkeltpunkter i dagens læreplan som kan fjernes, har kjerneelementgruppen vanskelig for å se at det er større emneområder som kan tas ut av læreplanen i matematikk fellesfag. Det betyr at faglig konsentrasjon primært må oppnås gjennom å vektlegge arbeidsformer som fremmer relasjonell forståelse, og ved å omorganisere, omprioritere og effektivisere innholdet i dagens plan. Vi tror dette kan gjøres dels ved å unngå den stadige repetisjonen (spiralprinsippet) som preger mye av dagens skolematematikk, og ved å tone ned noen av de temaene som har fått unødvendig mye plass tidlig i grunnskolen (Utdanningsdirektoratet, 2017).

Algoritmisk tenkemåte blir i denne skissen foreslått som en generell kompetanse i faget, og gjennom algoritmisk tenkemåte skal grunnlaget for programmering dannes (Kjerneelementgruppen for matematikk, 2017b). Som tidligere nevnt kom det inn over 100 høringsinnspill på dette utkastet, og det var en gjennomgående sterk motstand mot ideen om å innføre programmering som del av matematikkfaget i mange av høringsinnspillene (Kjerneelementgruppa for matematikk, 2017b). Det ble likevel understreket at mange er for programmering som eget fag, men at det ikke er rom for programmering innenfor dagens rammer. Det vanligste argumentet i høringsinnspillene er at innføringen vil føre til ytterligere stofftrengsel og derfor vil motvirke istedenfor å stimulere til dybdelæring (Kjerneelementgruppa for matematikk, 2017a). Disse argumentene vil jeg komme tilbake til senere i analysen. Den andre skissen til kjerneelementer kom 23. oktober 2017, også i denne skissen legger man algoritmisk tenkning til grunnlag for programmering, men nå er algoritmisk

tenkning tatt inn under kjerneelementet problemløsning og utforsking, og står ikke som eget kjerneelement. De fem grunnleggende ferdighetene blir også skissert som relevante, og under digitale ferdigheter foreslås det at ulike programmeringsspråk skal fungere som hjelpemiddel for flere matematikktemaer. Denne skissen fikk 120 innspill og også her viser oppsummeringen at mange er imot at programmering blir en del av matematikkfaget, selv om denne innspillsrunden ikke gir et entydig bilde av dette. Argumentene fra forrige runde finner vi også nå, i tillegg er mange bekymret over manglende lærerkompetanse innen programmering. Det viser seg også at det er en del uklarheter rundt begrepet algoritmisk tenkning, siden ordet algoritme frembringer ulike assosiasjoner i ulike miljøer (Kjerneelementgruppa for matematikk, 2017c).

Den tredje og siste skissen til kjerneelementer kom i mars 2018. Fortsatt er programmering foreslått som del av de digitale ferdighetene, på samme måte som i forrige skisse. Det som er ulikt er at programmering ikke nevnes spesifikt i sammenheng med algoritmisk tenkning under kjerneelementet utforskning og problemløsning. I denne skissen finner vi programmering under «sentrale begreper, metoder, tenkemåter, kunnskapsområder og uttrykksformer i kjerneelementet». På videregående nivå blir det foreslått at elevene skal utforske og løse problemer ved hjelp av programmering. I kjerneelementet *Modellering og anvendelser* foreslås det at elevene skal bruke programmering til å utforske matematiske modeller. Oppsummeringen av høringsinnspillene viser fortsatt at mange er kritiske til at programmering og algoritmisk tenkning er en del av faget. Dette forslaget får også mye støtte for programmering, da med forbehold om at det integreres på en god måte og at denne delen av matematikkfaget ikke får et oppskriftsfokus. Det fremheves spesielt at programmering er fremtidsrettet (Utdanningsdirektoratet, 2018b).

Det ble etter tre høringsrunder fastsatt seks kjerneelementer, der de fem første kjerneelementene beskriver arbeidsmåter, metoder og tenkemåter i matematikk. Det sjette kjerneelementet beskriver de sentrale kunnskapsområdene i matematikk. Elevene skal møte det sjette kjerneelementet gjennom de fem første kjerneelementene. Disse seks er:

- Utforsking og problemløsing
- Modellering og anvendelser
- Resonnering og argumentasjon
- Representasjon og kommunikasjon
- Abstraksjon og generalisering
- Matematiske kunnskapsområder

I forhold til programmering er det spesielt de to første som er viktig da algoritmisk tenkning nevnes i det første kjerneelementet, og modellering knyttes sammen til teknologi. Disse kjerneelementene i sin helhet er fastsatt slik:

> **Utforsking og problemløsing** – Utforsking handler om at elevene leter etter mønstre og finner sammenhenger. Elevene skal legge mer vekt på strategiene og framgangsmåtene enn på løsningene. Problemløsing handler om at elevene utvikler en løsningsmetode på et problem de ikke kjenner fra før. Algoritmisk tenking er viktig i prosessen med å utvikle strategier og fremgangsmåter og innebærer å kunne bryte ned et problem i delproblem som kan løses systematisk (Kunnskapsdepartementet, 2018b, s. 15).

> **Modellering og anvendelser** – Elevene skal ha innsikt i hvordan matematikk brukes i dagligliv, samfunnsliv, vitenskap og teknologi. Det innebærer å ta en problemstilling fra virkeligheten, omformulere den til en matematisk modell og tolke modellen i lys av den opprinnelige situasjonen. Elevene bør få innsikt i hvordan modeller kan anvendes i nye situasjoner. Kritisk tenkning er viktig å utvikle i slike sammenhenger (Kunnskapsdepartementet, 2018b, s. 15).

Når jeg studerer utviklingen fra første skisse til det som ble de endelige fastsatte kjerneelementer det tydelig at programmeringsfokuset er tonet ned, og i det endelige utkastet er begrepet programmering tatt helt bort. Programmering som begrep finner man nå under digitale ferdigheter i det endelige forslaget til læreplanen i matematikk, og i noen av kompetansemålene.

5.2 Argumenter for eller imot programmering

Jeg vil i dette delkapitlet fokusere på hvilke tanker og meninger mine informanter har omkring innføringen av programmering, og hvilke argumenter de kommer med for eller imot innføringen av programmering i matematikkfaget. I tillegg vil relevante innspill som ulike matematiske fagmiljøer sendte inn under ulike høringsrunder bli tatt med. Det er hovedsakelig seks hovedområder som kom frem etter kodingsprosessen av intervjuene og høringsinnspillene.

Disse er:

- Fremtidens kompetansebehov
- Stofftrengsel
- Dybdelæring
- Algoritmisk tenkning
- Digitale verktøy
- Sosiale og emosjonelle kompetanser

5.2.1 Fremtidens kompetansebehov

På 1980-tallet var det som nevnt i kapittel 2.2. gjort mange forsøk med programmering, men dette ble ikke videreført. Informantene har kommet med tanker om hvorfor programmering er tilbake i skoledebatten igjen. Det at programmering nå skulle komme inn som en del av fagene lå ikke i kortene ifølge Tom Lindstrøm, som var leder for kjerneelementgruppa i matematikk. Ifølge Lindstrøm fikk kjerneelementgruppa beskjed på første samling om at Utdanningsdirektoratet ville ha et forslag til kjerneelementer der programmering lå inne. Denne beskjeden fikk de kun muntlig og uten noen begrunnelse eller noe skriftlig dokumentasjon, Han sier også at Utdanningsdirektoratet var veldig unnvikende på hvem som egentlig hadde bestemt dette, noen sa dette var skjedd på politisk nivå, men hvem vet han ikke. Han trekker frem utviklingen i andre land som en av årsakene.

> Vi burde kanskje sett dette komme, hvis man ser på læreplanutviklingen i andre land som har tatt dette inn i planene på litt ulike måter.

Lindstrøm er likevel forundret over innføringen av programmering og begrunner dette med at selv om det har vært mye fokus på digital kompetanse, så har det vært liten satsning på informatikkfaget i seg selv. Han trekker frem at ved Universitetet i Oslo har de jobbet med realfaglig programmering i mange år. Realfagstudentene ved UiO lærer programmering første skoleår fordi studentene skal ha studier der de må kunne programmere, eller de kommer ut i

jobber der de enten må forholde seg til programmering, eller må kunne modifisere eller lage sine egne programmer.

> … men dette synes jeg er litt annerledes. Vi gir dette til våre realfagstudenter som vi vet har behov for dette. Det å si at hele befolkningen skal lære og programmere fra 2. -3. klasse er en helt annen grunnlagsproblematikk. Utviklingen de siste 25-30 år har jo gått mot at brukergrensesnittene blir enklere og enklere, og du trenger jo egentlig å kunne mindre og mindre for å kunne bruke programvare, og at vi nå liksom skal helt tilbake for å se på det grunnleggende synes jeg er litt rart.

I dag er det blitt enklere og enklere å ta i bruk digital teknologi, dette er noe Andreas Haraldsrud trekker frem. Haraldsrud jobber selv som lærer og har god erfaring i å bruke programmering i undervisningen.

> Jeg tror at med den brukervennlighet som er i dag, så kom brukerperspektivet veldig frem og så glemte man skaperperspektivet og det å forstå datamaskinen, istedenfor å bruke den.

Både Haraldsrud, Tom Espen Kristensen og Børre Stenseth trekker frem IT-utviklingen generelt, både i skolen og samfunnet som en av grunnene til at programmeringsbølgen er tilbake. Stenseth, som også var aktiv med i debatten på 1980-tallet, sier bl.a.:

> Det var ganske naturlig at det ikke ble noe fart i det på 80-tallet, utstyret var dyrt og det var lite tilgjengelig og relativt få som drev med det. I dag med internett og alt som digitaliseres det gjør jo at aktualiteten er mye større nå enn da.

At forventningene til programmering er annerledes i dag er noe Kristensen trekker frem.

> Jeg tenker at det er en annen forventning i samfunnet i dag til programmeringskompetanse, enn det var på 80-tallet. Den gangen var det begrenset med hardware, i dag er vi omgitt av teknologi på en helt annen måte. I tillegg er det gjort enklere nå, med blokkprogrammering osv. Det er lavere terskel nå en da, som kan være en av årsakene til at programmering er tilbake.

Cathrine Tellefsen tror at det på 1980-tallet rett og slett var for vanskelig, skummelt og nytt for å klare det nasjonale løftet i forhold til programmering. Datamaskinen va relativt ny da, men siden den gang har vi kommet langt med den digitale kompetansen og i dag er datamaskinen et viktig verktøy.

> Datamaskinen bør brukes for det den er verdt. Den kan gjøre en del ting mye bedre enn oss, så da burde vi bruke den til det. Datamaskinen gjør at fag forandrer seg, sånn at matematikk er ikke det samme som for tjue år siden, man kan gjøre helt andre ting, f.eks. med programmering nå.

Både Stenseth og Haraldsrud trekker frem dette med fremtidens kompetanse. Haraldsrud påpeker at programmering brukes mye ved universiteter, høgskoler og i jobbsammenheng, og det blir brukt som et integrert verktøy i mange sammenhenger. Stenseth trekker frem litt ulike perspektiver:

> … det ene er jo det yrkesmessige og kunnskapsmessige og det andre er jo å forstå litt av verden rundt seg.

Tellefsen mener at det er på høy tid å få programmering med datamaskin inn i skolen, hun mener at skolene er modne for dette og at det egentlig burde skjedd for lenge siden. Norge er som nevnt et av de siste landene i Europa som innfører dette i læreplanene i skolen, og det kommer i forbindelse med fagfornyelsen av læreplanen. Det trekkes frem at brukergrensesnittet er blitt mye lavere enn tidligere, og at forventningene til bruken og kompetansen i forhold til programmering er endret seg. I dag brukes programmering i mye større grad enn før, og derfor ses dette på som en kompetanse for fremtiden av flere av informantene.

5.2.2 Stofftrengsel

Informantene var i utgangspunktet enige om at eget fag hadde vært mer effektivt dersom det er programmeringsferdighetene til elevene og befolkningen som er viktigst i fremtiden. Uansett vil innføringen føre til stofftrengsel på en eller annen måte, men informantene er enige i at eget fag nok hadde vært det beste. Kristensen satt selv i Sanneutvalget og sier han skulle ønske at det ble et eget fag:

> Det har noe med at hvis det skal bli noe «kjøtt på beinet» her så må det brukes tid på dette. Det å lære seg å programmere er ikke gjort i en håndvending for mange, så man må få tid til å jobbe med prosjekter eller problem som krever litt av elevene. Når det blir emner som skal integreres i ulike fag så blir det fort at det smuldrer litt opp og kanskje det blir litt på noen prosjekt, men det gjennomsyrer ikke noen ting. Så et eget fag vil sikre at det ble en struktur og arbeid med dette, men slik ble det ikke

Lindstrøm refererer også til Sanneutvalgets rapport:

> Det finnes jo kun én skikkelig utredning på dette her, Sanneutvalget, og de fraråder jo eksplisitt å ta programmering i matematikkfaget.

Også Haraldsrud mener det beste ville vært å ha hatt programmering som eget fag, men ser likevel nytteverdien i å få det inn i matematikkfaget, og ser på dette som den nest beste løsningen. Han påpeker at det er mye matematikk man kan gjøre med programmering som man ikke kan gjøre uten.

Hvis det skille blitt et eget fag ville det ført til at andre fag mister timer, så Tellefsen mener det er en fornuftig begynnelse at programmering kom inn i matematikken istedenfor. Hun utdyper likevel hvorfor det kanskje burde vært et eget fag og sier:

> Det er jo allerede et eget fag, her har vi jo institutt for informatikk som er eget fagområde, og noe handler jo om realfaglig programmering, men det er jo veldig mye mer enn det. Og jeg tenker jo at fremtidens befolkning trenger mer enn realfaglig programmering slik det ligger an til nå.

Hvis det hadde blitt et eget fag ville det ha vært relativt krevende for det er såpass manglende kompetanse på alle nivåer, dette er noe både Tellefsen og Stenseth understreker. Tellefsen tenker at man må begynne i det små, og knytte det opp til noe kjent og trygt i matematikken, der matematikken er i sentrum, noe også Stenseth trekker frem:

> Nå har vi mulighet til å få det inn litt sånn fra sidelinja, og i så store doser at man greier og håndtere dette. Og det er nok mye lurere.

En konsekvens av at programmering blir en del av et eksisterende fag, vil være at annet fagstoff da må vike for å gi plass. Stofftrengsel en viktig årsak til at så mange var imot programmering i matematikkfaget ifølge oppsummeringen av høringsinnspillene. Det er klart at det må bli endringer i læreplanen for å få plass til programmerin, og ifølge Lindstrøm måtte jo innføring av noe nytt i et fag som allerede er fullt, føre til enda dårligere tid enn det er i dag, eller så må noe at de klassiske matematikktemaene ut. Kristensen er enig i at dette kan føre til stofftrengsel, men tilføyer at det kommer an på måten man underviser på. Han henviser til M87 da problemløsning var et av hovedområde:

> På 1980-tallet var problemløsning eget hovedområde, og dette førte til at det kom som et eget kapittel i mange lærebøker. Hvis vi nå får et eget kapittel i boka som heter programmering, slik som problemløsning på 80-tallet, så betyr det at man får stofftrengsel. Men hvis man bruker programmeringen i læring av de ulike temaene, så kan det fungere.

Han er likevel tydelig på at på et tidspunkt må det brukes en del tid på opplæring i selve programmeringen og at det nok i starten av arbeidet med den nye læreplanen høsten 2020, vil ha de største utfordringene da disse elevene ikke har lært noe programmering i grunnskolen, bortsett fra dem som valgte programmering som valgfag. Etter hvert vil dette bli bedre, og da vil det bli enklere å bruke tid på utforskning og problemløsning, istedenfor opplæring. Dette er også noe Haraldsrud trekker frem. Han mener likevel at i opplæringen er det mulig å bruke programmeringen til å lære matematikk, og selv om han her snakker om ungdomsskolen, vil nok dette også være overførbart til videregående.

> På ungdomskolen må en jo ha en viss opplæring i dette, og da er det ikke sikkert man kommer rett i matematikken, men man kan alltid bruke matematiske eksempler for å illustrere programmeringstekniske ting, sånn som når man skal lære om funksjoner i programmering så kan man jo bruke funksjoner fra matematikken, og dermed få med seg litt algebra på veien, og utforske ting på en annen måte.

Stenseth skjønner at mange vil oppleve at det blir stofftrengsel, og forstår frustrasjonen. Han trekker frem at programmering ikke kan bli en detalj som man gjør ett par timer i halvåret, det må ta tid hvis det skal komme noen vei, og da vil det måtte føre til at det tar tid fra annet, eller at noe må fjernes.

> Noen av argumentene som føres mot dette er nok reelle, for dette må gå på bekostning av noe. Så man må veie gevinsten imot tapet på en måte. I et langsiktig perspektiv tror jeg man vil vinne på dette.

Han legger også til at hvis man skal arbeide med mer krevende oppgaver i matematikk, så krever dette mer enn en helt generell opplæring. Han forteller videre at han har jobbet tett med en matematikkdidaktiker en periode i flere prosjekter i ungdomsskolen og at de ofte blir usikker på om de bruker matematikk til å lære programmering eller motsatt.

> … vi finner hverandre veldig når det gjelder å sette opp problemløsning og koble de mot ting, og når vi gjør eksperimenter med elevene, så er vi usikker ofte blir usikker på om de bruker matematikk til å lære programmering eller motsatt. Så da er vi enige om at det egentlig ikke spiller noen rolle, man kan lykkes fra den ene eller den andre siden.

At innføringen av programmering vil kunne føre til stofftrengsel er noe alle er enige i, men at det bare er negativt er ikke en selvfølge. Det ser ut som de fleste informantene mener at elevene og lærerne også har noe å hente fra innføringen. Dette avhenger av hvordan man underviser, og det er viktig hvilke undervisningsmetoder læreren benytter. Fagmiljøet frykt for at klassiske matematikkemner måtte vike har vist seg å være reell. Hvis man ser på det siste utkastet til den nye læreplanen har emnene på videregående blitt redusert ganske kraftig. Kristensen sitter selv i læreplangruppa for matematikk og sier at han forstår hvorfor en del av fagmiljøene er imot.

> Så er det at programmering tar tid. Som i R1 så ble geometrien tatt ut, og det er jo fordi vi måtte rydde plass. Jeg sliter litt med det selv om, for jeg liker godt geometridelen, men programmering tar tid.

Han legger til at han tror at man kan få til mye bra i undervisningen med programmering. Men man må begynne litt enkelt, og så jobbe seg opp igjennom etter hvert og elevene blir bedre. Han påpeker at det er viktig å knytte programmering til tingene vi jobber med, å se mulighetene dette kan gi.

At differensiallikninger ser ut til å bli borte fra læreplanen er noe flere av informantene har kommentert. Haraldsrud sier at det jo er først når programmering kommer inn i faget at differensiallikninger blir interessant. Mange lærebøker i R2 sparer dette emnet til slutt og veldig mange lærere følger det. Han påpeker at da er det jo uansett for sent til at de kan bruke det i fysikk eller kjemi fordi da er man forlangt ute i faget. Både han og Tellefsen trekker at i dagens læreplan, og i mange lærebøker er eksemplene innenfor temaet differensiallikninger ofte spesialtilfeller som er lite virkelighetsnære. I arbeidet med differensiallikninger i fysikken vil det ofte dukke opp problemer som ikke lar seg løse analytisk, men som enkelt lar seg løse ved hjelp av numeriske fremgangsmåter og programmering. Tellefsen bemerker bl.a.:

> … de lærer 1. ordens differensiallikninger, og løse å løse disse for noen små sære spesialtilfeller, som ikke nødvendig vis kan brukes på en reell problemstilling. Men lærer du f.eks. Eulers metode, så kan du løse alle differensiallikninger, så det er sånn at jeg tenker at vi ikke nødvendigvis fortrenger noe stoff, men vi fjerner noe sånn «pugge-regler» type stoff.

Det er enighet blant informantene om at programmering vil føre til stofftrengsel. Å lære programmering tar tid, og denne tiden vil kunne gå på bekostning av temaer i matematikkfaget. Flere av matematikkfagene i den videregående skole har også fått redusert temaene i læreplanen for å få plass til programmering. Det trekkes likevel frem at selv om programmering fører til stofftrengsel, kan innføringen også føre til bla. endrede undervisningsmetoder og en annen innfallsvinkel til en del av pensum i matematikkfaget.

5.2.3 Dybdelæring

Om programmering kan være med på å fremme eller hemme dybdelæring har fagmiljø og andre instanser ikke klart å bli helt enige om. Dahl, Ranestad og Hole (2017), alle ansatt i det matematiske fagmiljøet ved UiO, skrev kronikken *Programmering rammer dybdelæring i matematikk* som ble publisert i Aftenposten. De påpeker at programmering kan være en viktig del av høyere utdanning, spesielt innen realfag, men anbefaler ikke å innføre programmering som en del av matematikken i grunnskolen, annet enn som et valgfag for spesielt interesserte. De trekker frem at å ta inn et helt nytt emne vil dybdelæringen i matematikk rammes, og det fryktes at matematikksvake elever vil rammes, da disse elevene trenger god tid til å bygge opp

forståelsen sin. Det vises til at matematikklærere og matematikkdidaktisk forskning lenge har etterspurt en reduksjon av antall emner som skal dekkes av skolematematikken, og innføring av et helt nytt emne er da å gå i motsatt retning (Dahl m.fl., 2017). Også noen av høringsinnspillene som ble sendt inn under høringsrundene av kjerneelementene uttrykte samme bekymring som kronikkforfatterne. Lindstrøm mener at om programmering vil fremme eller hemme dybdelæring handler om hvilket aspekt man har på dybdelæring, og trekker frem at det kan være dybdelæring på ulike felt. Han viser til at det er mulig å se på differensiallikninger som en del av dybdelæringen i R2 og geometrien som en del av dybdelæringen på R1, og begge disse emnene er nå tatt ut av læreplanen.

> Differensiallikninger gir deg dybdekunnskaper innenfor derivasjon og integrasjon, og geometrien gir evner til å tenke geometrisk, men har også med bevis og begrunnelser og gjøre, Det å utforske en situasjon der det ikke er noen åpenbare løsninger i utgangspunktet. Hvis de har blitt trengt ut på grunn av programmering, så taper vi dybdelæring.

Han legger til at programmering også kan ha med seg sin egen dybdelæring hvis innføringen av programmering i matematikkfaget kan få elever til å se på matematisk eksperimentering og matematisk tankegang og utforskning på en annen måte, men han understreker at det er vanskelig avgjøre nå, før planene er klare. Tellefsen trekker frem at med programmering får en man annen representasjonsform som innfallsvinkel til matematikken, som kan gjøre at det må jobbe mer i dybden og at man kan ha mye å vinne på dette i matematikkfaget. Selv om hun tror programmering vil føre til at noe av matematikken må ut, er hun tydelig på at dette vil kunne føre til mer dybdelæring:

> Jeg tror det blir mer dybdelæring fordi vi får en annen innfallsvinkel til stoffet. Vi vet at hvis du skal forstå en ting ordentlig godt, så er det bra med ulike representasjonsformer.

I skolen bruker man mye tid på å lære metoder for å finne den antideriverte, men likevel finnes det mange integraler man ikke kan løse ved å bruke disse metodene. Tellefsen tenker hvis poenget er å skjønne hva integral er og kunne finne et areal under grafen, da er det jo et lite program som kan løse dette, så kan man heller fokusere på relevante problemstillinger.

> Ja, jeg tenker at det kommer til å fortrenge noe stoff ja, men det kommer til å føre til dybdelæring og på noen steder så kan vi komme til å blir kvitt noe av den puggingen. Så ja, vi tar på denne måten ut pugging av integrasjonsmetoder, men jeg tror ikke det blir borte noe matematikkforståelse for det.

Poenget er jo å gjøre matematikk med programmering understreker Haraldsrud. Han mener at hvis man ikke har noen utdannelse innen dette og heller ikke vet hva mulighetene er, så ser man jo ikke hvordan det kan styrke matematikkfaget og bidra til dybdelæring. Haraldsrud trekker videre frem mulighetene innen derivasjon, integrasjon og differensiallikninger og ved å bruke numeriske metoder og programmering.

> Du kan gjøre ting slik som Euler og Newton gjorde for flere hundre år siden, men de gadd jo ikke holde på med det fordi det var jo slitsomt, de hadde jo ikke PC. De måte sitte der å gjøre det 50-60 ganger for hånd, samme operasjonen. Nå kan vi gjøre hundretusener av den operasjonen på under ett sekund.

Når man skal lage et program må elevene angi hvilke instrukser som skal gjennomgås i programmet helt eksplisitt. Han sier at elevene må være ganske sikker på matematikken for å kunne få til å lage programmer, og mener derfor at elevene lærer en del matematikk ved å holde på med programmering. Når man starter med den numeriske matematikken så vil elevene få et helt nytt perspektiv på ting som f.eks. derivasjon, likningsløsning og integrasjon.

> Det å derivere numerisk så jobber man hele tiden med definisjonen til den deriverte, noe som fører til at man faktisk må forstå hva derivasjon egentlig er. Så det åpner en del nye muligheter, den numeriske matematikken.

At det er lett å se for seg anvendelser innen programfagsmatematikken er også noe Lindstrøm trekker frem, og da spesielt ved numeriske metoder. Likevel tenker han at dette kanskje ikke er det mest interessante man kan gjøre med programmering.

> … en av ulempene er at det er den samme løkka man bruker, med litt forskjellig innpakking og litt forskjellige parameterverdier, og litt forskjellige funksjoner, men programmet og det programmeringstekniske, er faktisk det samme. Så hvor mye lærer man egentlig av det?

Hvis det kun blir prosedyrer elevene skal lære seg og ferdige programmer som de kan laste ned og bruke til eksamen, mener ikke Kristensen at programmering er noe god idé. Men han trekker også frem det det motsatte:

> Men hvis de selv får tid til å lage programmene, så vil dette vil kunne fungere bra i forhold til å forstå å se ting sammenheng, som jo ligger bak begrepet dybdelæring. Så det er igjen helt avhenting av hvordan man jobber med dette.

Stenseth ser også at programmering åpner for å kunne gjøre nye ting i matematikk f.eks. av numerisk art, som man ikke kan gjøre uten. Han mener disse tingene kan være lurt å se på, og at på sikt kan bruken av programmering være med på å endre pensum i matematikkfaget.

Om innføringen av programmering vil kunne fremme eller hemme dybdelæring er noe som har vært omdiskutert, og informantene har noe delte i synspunkter på dette. På den ene siden kan programmering fremme dybdelæring i matematikkundervisningen fordi elevene nå skal lage side egne programmer, istedenfor å bruke ferdiglagede programmer som f.eks. CAS. For å få dette til må elevene forstå matematikken bak i større grad. På den andre siden gjør innføringen av programmering at læreplanen må reduseres, bl.a. på bekostning av geometri i R1 og differensiallikninger i R2, noe som kan bidra til dårligere dybdelæring i faget.

5.2.4 Algoritmisk tenkning

Algoritmisk tenkning er et omdiskutert begrep i høringsuttalelsene, og det trekkes bl.a. frem at begrepet kan misoppfattes og at det ikke er like dekkende som det engelske «computational thinking». Matematikkseksjonen ved Lærerutdanningen ved Universitetet i Sørøst-Norge skriver i sitt høringsinnspill til det første læreplanutkastet at begrepet bør forklares dypere og at det bør komme tydeligere frem hvorfor dette er essensielt for programmering. De mener at begrepet kan misforståes av flere (bl.a. lærere) og at det kan oppfattes som «oppskriftstekning» (Hofman, 2018). Matematikkseksjonene ved Institutt for lærerutdanningen ved Norges Teknisk-naturvitenskapelige universitet (NTNU) stiller spørsmålstegn ved å ta inn «utvikle algoritmisk tenkning» i kjerneelementene, og skriver i sitt høringsinnspill at de ikke ser en naturlig

sammenheng mellom algoritmisk tenkning og problemløsning. De trekker frem at å legge algoritmisk tenking inn i kjerneelementene kan få en negativ effekt ved å føre til mer pugg og algoritmedrill, men er enige i at det å bryte ned et problem ned i mindre deler hører hjemme under problemløsning (Sikko, 2017). Både Lindstrøm og Tellefsen synes ikke at den norske oversettelsen algoritmisk tenkning er like tydelig som det engelske CT, og flere av informantene føler at begrepet AT blir strukket veldig lang. Lindstøm utdyper dette slik:

> Jeg føler at tilhengere av AT trekker det begrepet veldig lang, jeg synes at veldig mange trekker det så langt at det egentlig blir snakk om matematisk tenking, eller matematisk problemløsning. Det er ikke nødvendigvis noe algoritmer, i hvert fall ikke noe programmer igjen i det hele tatt. Og det synes jeg er litt sånn politisk snikangrep nærmest, altså at man utvidet begrepet så langt at ingen kan være imot det. Og etterpå sier man «ja dere er enig, og dette er viktig, og da må vi begynne å programmere». Jeg synes AT er blitt så vidt at det nesten ikke er meningsfullt lengre.

Tellefsen sitt ønske er å bruke begrepet computational literacy, som hun mener er mye mer dekkende enn AT. Hun begrunner dette med at literacy er et helt kompetansefelt, og at hun da tenker at dette innebærer kompetanser og ferdigheter i å bruke datamaskinen, og skjønne hva den kan hjelpe oss med og hva den ikke kan hjelpe med, og hvordan bruke PC til å løse problemstillinger er en måte å tenke på. Hun tenker at man valgte AT som begrep er et retorisk grep, siden dette er litt kjent fra matematikken, og som gjør at det er lettere å tørre å gå inn i dette.

> Algoritmisk tenkning er noe du liksom ikke kan opponere mot, selvfølgelig trenger vi algoritmisk tenking, så hvis vi kaller det det, er det liksom litt trygt på en måte for det har vi holdt på med i matematikken før. Men hvis du egentlig skal ta inn over deg det som skjer så handler det egentlig om computational literacy, og det er mer enn algoritmisk tenkning.

Stenseth påpeker også at dette er et begrep som oppfattes ulikt:

> Alle som skriver om dette skriver jo forskjellige ting, noe drar det jo helt ut og andre går i alle mulige detaljer og bruker det til hva som helst. Men hvis vi skal se på essensen i det, så er det å finne et slags mønster og løse problemer på en systematisk måte og for meg så holder det som definisjon.

Også Kristensen er enig i at den norske oversettelsen ikke er så god, noe han begrunner dette med at mange henger seg opp i ordet algoritme. I skolematematikken er dette begrepet ofte forbundet med å følge regler og metoder, og han tror det er en av årsakene til at ordet algoritmisk tenkning blir kritisert:

> …men da blir kritikken på litt feil grunnlag, at det tenkes at det handler om å lære seg standardalgoritmer. Men det er jo heller motsatt, det handler om å selv lage algoritmer som kan løse problemer.

At begrepet AT handler om å utvikle algoritmer selv, og ikke pugge dem, er noe Haraldsrud presiserer når han snakker om AT-begrepet. Han mener at AT er et veldig generelt begrep, og at det egentlig ikke handler så mye om algoritmer. Han tenker at det handler mer om å abstrahere problemstillinger å klare å dele et stort problem i små problemer, lage modeller og utvikle algoritmer. Dette er også noe Kristensen er enig i:

> Algoritmer er jo med i dette begrepet selvfølgelig, men det er ikke den mest sentrale delen egentlig. Men algoritmen for å løse et problem må en jo da ha etter at man har abstrahert og generalisert og tatt fra hverandre dette problemet og analysert det. Så det handler det om å analysere og reflektere rundt problemer på en litt mer drøftende måte.

Kristensen ser på algoritmisk tenkning som en kompetanse som bl.a. innebærer feilsøking, fikling, utholdenhet og kreativitet. Han trekker også frem hva han tenker det handler om:

> Det handler om å løse et problem, med å dele det opp i mindre problem, ikke nødvendigvis med en datamaskin, og at man kan abstrahere ting, dele opp i steg slik at ulike deler lettere lar seg løses. Det er en annen måte og tenke på som er verdt et eget begrep.

Hva begrepet *algoritmisk tenkning* egentlig betyr, er uklart ifølge flere av informantene. Det kan lett misforståes hvis man henger seg opp i ordet algoritme, som betyr noe helt annet. Det påpekes at det engelske ordet «computational thinking» er et begrep som er mer dekkende enn det norske AT-begrepet. Informantene har noe ulike oppfatninger av begrepet, selv om alle presiserer at det handler i all hovedsak å løse problemer ved å dele opp problemet, og kunne abstrahere og reflekterer rundt problemet, med eller uten datamaskin.

5.2.5 Digitale verktøy

Det faktum at vi nå får tre (fire hvis man skiller mellom Geogebra og CAS) digitale verktøy i skolen kan skape utfordringer for elever og lærere og Lindstrøm understreker dette:

> En problemstilling til, er jo trengsel, vi risikerer jo å få veldig mye i skolen. Regneark, Geogebra med CAS og programmering. Jeg er litt redd for at disse tingene kommer til å gå i veien for hverandre, at det blir veldig mye.

Haraldsrud er enig i at det kan bli for mange digitale verktøy, og at spesielt CAS og programmering ikke bør leve side om side. CAS kan være veldig instrumentell, og skaper lite forståelse for hva som egentlig skjer når man bruker verktøyknappene, eller kommandoer.

> Enten må man ha det perspektivet at man må forstå det man gjør, eller så man bruke en kalkulator som gjør ting for deg. Må velge enten eller.

Sammenhengen mellom forståelse for matematikk og CAS er også påpekes av Tellefsen:

> … det blir stadig mer teknologi i matematikk, Geogebra ikke minst, og så ser man at man lærer jo ikke noe mer matematikk med dette, i alle fall ikke av CAS, det man bare trykker på en tast.

Lindstrøm ser også fordel med å lage et program selv, istedenfor å bruke CAS hvis forståelse skal være et sentralt mål. Han sammenlikner CAS med en «sort boks» der elevene putter inn noe data og trykker på knapp, så gjør maskinen ett eller annet, så kommer noe ut. Ved å bruke programmering må elevene vite hva som er inni «boksen»:

> Ved programmering så må du på en måte lage den sorte bokse selv. Du må vite hva som er inni der, og det kan du lære en del matematikk av, du må bruke den matematikken du kan for å vite hva du skal putte inn i boksen, og bruke det du kan av programmering til å lage dette. Sånn sett kommer man på en måte nærmere matematikken gjennom programmering, enn ved f.eks. å bruke Geogebra eller CAS.

Kristensen tenker at det å jobbe med CAS kan være programmering på en måte, men at det trenger ikke være det, alt avhenger av måten man underviser på. Han tenker at mange er kritiske til CAS siden det ofte sammenliknes med å kun trykke på knapper uten forståelse, men legger til at programmering kan det bli veldig instrumentelt det også. Han understreker at alt kommer an på hvordan man legger dette opp undervisningen og at det gjelder å finne de gode eksemplene og de gode oppgavene. Han mener derimot at elevene vil takle å ha flere verktøy å forholde seg til.

> Jeg tror ikke det blir for mange verktøy for elevene å forholde seg til, men de må få tid til å jobbe med det. Men hvis det blir jobbet overfladisk med disse verktøyene så blir det dårlig, man må jobbe grundig med dette. Jeg tror ikke at elevene vil få problemer i å forholde seg til flere verktøy, det er ikke min erfaring i alle fall.

Kristensen påpeker at elevene må få tid til å tilegne seg den kompetansen som behøves, og at lærerens valg av undervisningsmetoder vil være avgjørende for hvordan resultatet blir. Når elever bruker digitale verktøy er det ikke til å komme utenom av at man av og til taster inn feil, slik at det som forventes som resultat ikke stemmer. Dette er vanlig i Geogebra, når man f.eks. skal tegne grafer. Det er fort gjort å bruke komma istedenfor punktum i desimaler, noe som Geogebra ikke skjønner. På kalkulator er det fort gjort å glemme parenteser, noe som gjør at svaret kan bli feil. Når man programmerer er feilretting, også kalt debugging, en naturlig del av prosessen, og ofte går mye av tiden med til prøving og feiling for å lage et program som fungerer slik man ønsker. I arbeid med matematikkoppgaver er det heller motsatt, og elevene vegrer seg ofte for å svare feil eller gjøre feil. De er ofte ikke gode i å finne, eller lære av de feilene de gjør. Stenseth ser på feilrettingsdelen av programmering som noe av det viktigste:

> … noe av det aller viktigste er hypoteseprøvingen, og prøving og feiling. Du prøver på en systematisk måte. Og det syns jo jeg er et sånn undervurdert tema med programmering, og det er jo at programmering er 99 % feilretting. Mens mange som underviser i programmering, de skal ha det rett med en gang, og forbereder seg veldig godt for å presentere den beste løsningen, og så får ikke elevene til det og det blir en nedtur for mange. Jeg har programmert i 50 år, og jeg vet jo at jeg bruker over 90 % av tida til å rette feil.

Han trekker frem viktigheten av at elevene aksepterer å gjøre feil, og at de må trenes i å ha en systematisk måte å lage en hypotese for å rette den feilen på. Både Lindstrøm og Tellefsen påpeker forskjellen mellom holdningen til feil i matematikk og i programmering og Tellefsen sier bl.a.:

> …. og så er det en ting til så er det når du programmerer så gjør du feil, og det er helt umulig å ikke gjøre feil. Og dessverre så er det en sånn skolekultur på at «hvis jeg gjør feil så er jeg dum», men med programmering så blir dette borte for det er så vanlig å gjøre feil. Det er alltid en eller annen syntax error, eller logisk feil. Det blir plutselig spennende å gjøre feil for da kan man lære noe nytt. Det er jo egentlig sånn vi vil at læring skal skje, men det er ikke så mye av det i norsk matematikkundervisning.

At man kan lærer mye av feil er noe understrekes av Lindstrøm, med det argumentet at den tradisjonelle matematikkundervisningen ikke har klart å få dette godt nok frem. Han sier likevel

at det kan både være vanskeligere og mindre motiverende å finne feilen i et matematikkstykke, kontra når man programmerer.

> Ved å løse en matematikkoppgave rett på 4. forsøk, er jo bare å få rett fasitsvar. Så det er nok en mer fristende setting og kan være mer motiverende å debugge sitt eget program, enn å sitte og finne feil i matematematikk.

Akkurat dette at elevene gir fortere opp når de får feil i matematikk, enn når det får feilmeldinger når de programmerer, er noe Haraldsrud også opplever når han underviser:

> Jeg synes ofte i matematikk at de lettere gir mer opp, de får ikke noe feilmelding, det bare stemmer ikke med fasiten, og det er nok lett å bare legge dette vekk da. I programmering får man feilmeldinger som man kan bli gode til å tolke, og du kan utvikle gode debuggingsstrategier etter hvert så jeg synes at alle blir velig gode i dette.

For å få et program til å fungere riktig, må alle kodene være rett, og elevene må forstå hva som forårsaker feilen og kunne rette opp i programmet, og ofte får man feilmeldinger som kan hjelpe med dette. I matematikkfaget er feilene ikke like opplagte, og det kan være vanskelig å finne, noe som ikke gjør det like motiverende å finne feilen, eller rette opp i denne. Programmering kan på denne måten også påvirke de emosjonelle kompetansene, bl.a. motivasjon og utholdenhet, noe som trekkes frem av flere av informantene, og dette belyses i neste delkapittel.

5.2.6 Sosiale og emosjonelle kompetanser

Når det kommer til programmering er det særlig motivasjon, utholdenhet og samarbeid informantene trekker frem i forbindelse med læring av programmering og de emosjonelle og sosiale kompetansene. At programmering gir økt motivasjon er allerede nevnt av Haraldsrud. Han erfarer at elevene hans ofte kan sitte uten pause i flere timer og programmere:

> De sitter så fordypet, og så er det jo så gøy når de får til. De får jo en fysisk output, ikke bare tall, men et program som gjør noe for deg. Jeg synes elevene har god utholdenhet og blir gode på dette.

Akkurat dette at elevene lager et program, og ikke bare får et fasitsvar, påpekes også Lindstrøm og han mener dette er en faktor som kan bidra til økt motivasjon hos elevene:

> Det er klart at det å finne feil i sitt eget program, er nok mer motiverende enn å finne feil i et regnestykke. Belønningen er så mye større, det du sitter igjen med er et program som fungerer.

At programmering i tillegg kan bidra til økt utholdenhet er noe Stenseth trekker frem og han påpeker også at elevene trenger tid til å få gjøre feil, og tid til å rette opp i disse ved hjelp av god veiledning. Det å rette opp i, og endre programmer, krever algoritmisk tenkning og hypoteseprøving ifølge Stenseth, og han mener det er viktig at elevene lager hypoteser om hva som er feil, før de begynner med feilretting. Han beskriver et opplegg han var med på å gjennomføre i ungdomsskolen, der eleven skulle endre et program for å få et hjul til å rulle riktig. Elevene skulle endre på programmet uten veiledning, og det viste seg da at mange ikke fikk dette til.

> Dette lærte jeg mye av, i forhold til hvordan jobbe med problemløsning og hvilken support de burde ha når eleven gjorde det. Og det ligger mye der, det er så lett å gjøre slike intuitive endringer som ikke er reflekterte, og da kommer man ofte ingen vei.

Han understreker viktigheten av riktig veiledning og support, og at hvis dette fungerer vil det gå bra, og at man da kan studere mer morsomme problemstillinger i f.eks. matematikk:

> Så kan man gå inn på litt mer morsomme ting, numeriske ting, finne gylne snitt, studere fibonaccirekka, se på russisk multiplikasjon, det er mange ting du kan illustrere gjennom programmering og da forstår du mer av hva som skjer.

Tellefsen bruker seg selv som eksempel for å forklare hvilke erfaringer hun har gjort i forhold til dette med feil og motivasjon:

> Det er påfallende at jeg som egentlig er «flink pike» og vil gjøre alt riktig, når jeg programmerer så er jeg liksom fri fra det. Det er alltid noe man har glemt, kommafeil eller noe, det er lov å prøve og feile og utforske, og er en så morsom måte og jobbe på. I tillegg får man jo tilbakemelding med en gang, så er ganske ålreit.

Hun understreker at hun ser på programmering som en utforskende måte å jobbe på, og at man gjennom programmering kan se på ulike innfallsvinkler, diskutere, prøve ut nye måter å gjøre ting på, og at det ofte kan være flere måter å løse oppgaver på som ofte er svært matematiske. På denne måten tenker hun at elevene kan være mer utforskende i forhold til matematikkfaget gjennom innføringen av programmering. Kristensen er tydelig på at programmering kan gi bedre utholdenhet, men at det ikke er noe automatikk i dette:

> Så spørs det da når eleven skal programmere, vil de gi opp med en gang de ikke får til, eller vil de kunne stå i det og ha den utholdenheten. Og da er spørsmålet vil programmering være en sånn katalysator for utforskning og problemløsning, eller vil det bare være en ny greie som ikke blir bra likevel. Og det er et vanskelig spørsmål som tiden vil vise.

Tellefsen sier at programmering gir mulighet til diskusjon om ulike innfallsvinkler, og nye måter å gjøre ting på, og dette krever samarbeid.

> Så tror jeg at når man jobber med programmering så er en viktig bit det sosiale aspektet, for det er ikke ett fasitsvar, så man jobber sammen. Alle tror programmering er sånn «nerdegreie», men det er ikke sant. Det er en sosial måte å jobbe på, og derfor kanskje man holder ut mer fordi man sitter ikke der i et vakuum med det ene problemet og gir opp, man sitter og diskutere og tester sammen. Jeg tror ikke nødvendigvis at elevene blir mer utholdende i seg selv, men jeg tror at programmeringens natur gjør at det blir lettere å holde ut fordi de stadig får tilbakemeldinger.

Tellefsen viser i tillegg til at det engelske begrepet computational literacy er tredelt, og at den ene av disse tre delene er den sosiale biten og at denne delen er helt sentral for å lære å programmere. Også Kristensen mener at samarbeid er viktig og viser til at i beskrivelsen av AT er samarbeid en vesentlig del. Stenseth trekker frem en metode som heter parprogrammering der en skriver kodene og den andre kommenterer, og så bytter man roller etter en stund. Denne metoden testet han og en kollega ut i samme opplegget som ble beskrevet tidligere der elevene skulle endre et program:

> Denne metoden brukte vi og det var interessant, det ga veldig entusiasme og iver, men de greide likevel ikke å knekke den koden og ta med seg problemet over og bruke matematikk til å formulere, istedenfor å kode. Men med noen små vink ville de nok klart dette.

Tellefsen trekker også frem det at programmering kan hjelpe de elevene som ikke er så veldig analytiske:

> Jeg tror en del av de elevene som strever litt med den veldig analytiske tilnærming til matematikken, hvor du bare må se hva du skal bruke av regler, f.eks. integrasjonsmetoder, de som ikke ser dette, de kan få en «ny vår» og det kan være hyggelig å være læreren deres fordi de må ikke være så analytiske for å lære matematikk, det tenker jeg er gøy som lærer.

Hun mener det er synd at mange elever som trenger en mer visuell tilnærming ekskluderes fra matematikken, fordi de kan jo være kjempegode i matematikk, men disse elevene må bare ha det vist på en annen måte.

Flere av informantene trekker frem at programmering er i seg selv er en metode som fordrer til samarbeid, gjennom f.eks. parprogrammering. Samarbeid i seg selv kan bidra til økt utholdenhet siden elevene jobber sammen, og det nevnes at siden sluttprodukter blir et program som fungerer, kan dette bidra til større motivasjon og utholdenhet. Ved å bruke programmering får elevene en litt annen innfallsvinkel på matematiske problemer noe som også kan engasjere og gi mer motivasjon i faget.

5.3 Forutsetninger for å lykkes

For å lykkes med implementering av programmering som en naturlig del av matematikkfaget i fremtiden, er den flere suksessfaktorer som poengteres av informantene. Lærerkompetanse er det som anses som absolutt viktigst, og at lærere bl.a. får tid og mulighet til etter- og videreutdanning innen programmering. Lærerne må også erfare at programmering er relevant i forhold til deres fag. I tillegg så trekkes eksamen inn som en viktig faktor om programmering vil bli implementert som en del av undervisningen, da eksamen ofte lager føringen for hva som blir vektlagt i undervisningen.

5.3.1 Lærerkompetanse og relevans

Alle er tydelig på at lærerkompetansen er den største faktoren for om dette kommer til å lykkes eller ikke. Stenseth har selv over 50 års erfaring innen programmering og understreker at læreren må kunne ganske mye for å kunne presentere og beskrive et problem slik at det er forståelig.

> Den store minusfaktoren er lærerkompetanse, og det kommer til å vare lenge tror jeg. For du må forstå ganske mye selv. Det å sette elevene i gang med å programmere og gå å se rundt hvorfor de ikke kan kompilerere[9] det er ikke noe poeng.

Haraldsrud påpeker at skolen kommer til å slite med lærerkompetanse, spesielt de første årene. Han mener man må regne med at det vil ta noen år med en innkjøringsfase.

[9] Kompilere betyr å se at programmet er formelt riktig

> Det er klart det er noen elever som vil rammes av uinspirerte og lite utdanna lærere. Så vi må ta på alvor dette med kursing og gode etterutdanninger, det må man tilby.

At kursene som tilbys må vise gode eksempler på oppgaver som passer ulike trinn tror han er viktig, slik at lærerne kan ta med seg oppgaver direkte inn i klasserommet som de allerede har jobbet med. Han kaller dette for undervisningsnær kursing, som med litt didaktikk, kan hjelpe lærerne til å se hvordan dette kan fungere i klasserommet, og hvordan man kan treffe og motivere ulike elevtyper. At alt må gjøre på fagets premisser er også svært viktig ifølge han:

> Ved UiO har vi alltid hatt dette på fagets premisser, og så man må være opptatt av å drive fag med programmering, og se på dette som er verktøy. Akkurat som man ser på matematikk som et verktøy i fysikk.

Både Tellefsen og Lindstrøm sier seg også enige i at programmering må undervises på fagets premisser. Lindstrøm understreker at det er viktig at dette ikke blir en isolert del, adskilt fra resten av faget.

> Må få opp lærerkompetansen. Det å få frem gode oppgaver og prosjekter, som ikke bare er morsomme, men som knytter programmering til matematikkfaget.

At den faglige vinklingen er spesielt viktig for lærerkompetansen er også noe Tellefsen presiserer og hun påpeker at vi være veldig tydelig på at det skal handle om fag:

> … for det er det som er viktig for oss lærere at vi kan knytte det opp mot faget. Så jeg er jo ikke så veldig glad i sånn «moderne sløyd», altså teknikk for teknikkens skyld, for da blir faget borte, og jeg tror norske lærere har en faglig stolthet og det er den jeg tror vi må bruke for å få dette til på en god måte.

Hun tror det kommer til å bli viktig at mange får videreutdanningskurs, men at det også legges vekt på interne kurs i skolene. Alt dette må jobbes med systematisk og jevnlig, og hun mener at det bør lages gode nettressurser og forum der lærerne kan få hjelp og støtte. Siden samarbeid er en viktig del av opplæringen i programmering, stiller hun seg skeptisk til at det tilbys nettbaserte MOOC-er uten faglig forankring. Kristensen mener at dette med lærerkompetanse er tosidig. Det ene er å lære å programmere, og det andre er å vite hvordan man skal bruke dette i undervisningen, og hvilke oppgaver som er gode og interessante. Han understreker at den didaktiske kompetansen er viktig, i tillegg til kompetanse i programmering. Dette adresserer han til å være et ledelsesansvar:

> Så er det jo også dette at det legges til rette for utviklingsarbeid fra skoleledelsen fremover, det vil være en viktig faktor. Lærerkollegiet må få mulighet til å utvikle seg i felleskap og dele erfaringer.

Kristensen er kritisk til måter mye av etterutdanningen i programmering er lagt opp til. Mange av studiene er nettbaserte, generelle MOOC-er, som ikke er knyttet til noen fag. Kristensen utdyper dette:

> Det tror jeg ikke er særlig smart. Skal lærere bli gode å undervise i programmering så må det jobbes med å forankre det i faglige problemstillinger. Og det skal være vanskelig. Oppgavene skal

> være slik at du må jobbe med de, feilsøke, fikle og herje litt, og hvis vi bare har dette på nett kan dette bli vanskelig.

Et aspekt Lindstrøm og Tellefsen også drar frem er frykten for det ukjente. Å øke lærerkompetansen og vise lærere hva dette egentlig handler om, og hvilken nytteverdi programmering kan ha for matematikkfaget er viktig å få frem hvis dette skal bli vellykket. Tellefsen tror at ved å ha fokus på det faglige, og se på programmering som et nytt verktøy, vil gjøre dette mindre skummelt.

> Vi er ute og skal gjøre ting vi ikke kan, og kommer til å møte elever som kan mer enn oss. Så den der læreren som den allvitende, den er jo en myte, læreren er jo ikke det, men noe liker kanskje å se på seg selv som trygg og sterk i klasserommet, og kan det meste. Disse vil bli utfordret, og da er det viktig at man er tydelig på at man kan det matematikkfaglige. Jeg tror at de som ikke er så glad i endring synes dette er litt skummelt. Det eneste vi kan gjøre for å få dette til på en god måte er å spille på det faglige. Være tydelig på at det er det matematikkfaglige som er viktig, og det kan norske lærere. Så må de lære seg en nytt verktøy, og det kan være litt vanskelig, noen tar det fort og noen tar det sakte, og av og til er elevene flinkere enn læreren.

Som vi ser er lærerkompetanse den faktoren som blir ansett som avgjørende i forhold til om implementeringen av programmering vil lykkes. Lærerne som skal undervise i dette, må få tilrettelagt for kurs og/eller etterutdanning. Det må også legges til rette for at lærere får mulighet til å både lage og dele gode opplegg. For lærerne vil programmering for faget skyld være det som er viktigst, og ikke opplæringen i programmering i seg selv.

5.3.2 Eksamen

Eksamen som styrende element for det som skjer i klasserommet er noe som flere av informantene anser som avgjørende for i hvilken grad lærere vil ta dette inn over seg eller ikke. Tellefsen tenker at dette bør komme på eksamen etterhvert, men at det bør innføres på samme måte som Geogebra. Det bør ta noen år slik at lærere får nok tid til å sette seg inn i dette og få erfaring før dette kan vurderes på en eventuell sentralgitt eksamen. Kristensen viser også til at erfaringsmessig ble det en stor endring i forhold til Geogebrabruken blant elevene, da det i 2015 kom på eksamen, og de signaler som blir gitt i forhold til eksamen er avgjørende:

> Jeg tror at eksamen har veldig mye å si for de valg som blir gjort av lærere og lærebokforfattere, sånn at hvis det blir sendt signaler om at programmering blir en del av sentralgitt eksamen. Så vil nok lærebøkene ta dette inn, og lærere finner ut at de må lære seg dette. Det er i alle fall vår erfaring fra våren 2015, da kravene til digitale verktøy ble innskjerpet, så skjedde det en stor endring, og plutselig brukte nesten alle elevene Geogebra.

Han legger til at hvordan eksamen blir lagt opp etter fagfornyelsen, ikke er avgjort enda, så det vil bli interessant å se hva det ender med. Lindstrøm tenker at programmeringsbiten kunne vært lagt opp som en del av en muntlig/praktisk eksamen i matematikk, og ikke som en del av den skriftlige.

> Vi har jo den muntlige eksamen som ingen riktig vet hva man skal gjøre med, og kanskje kunne man bruke denne delvis til å teste programmeringsferdigheter. Kanskje en mer vektstedpreget situasjon der elevene arbeider en hel dag der sensor går rundt og ser og spørrer, litt som den praktisk-muntlige eksamen som noen fag har. Tenker at dette kan være lab-delen av faget, gå inn å vise at man kan gjøre ett eller annet.

Haraldsrud mener også at måten eksamen legges opp blir viktig. I dag gis det føringer for hvilke verktøy man skal bruke gjennom ordlyden i oppgaveteksten til eksamen, noe han mener er dårlig bruk av digitale verktøy.

> Man må velge løsningsstrategi selv, det er jo hovedpoenget, man må lære seg å forstå når det er bra å bruke det ene kontra det andre.

Både Stenseth og Haraldsrud mener at det er viktig å forholde seg til få programmeringsspråk, helst ett, hvis ikke kan det bli vanskelig for elevene. Haraldsrud mener programmeringsspråket Python kan være et godt valg, og begrunner det med at det har såpass enkel syntaks, slik at elevene kan fokusere på faget og ikke trenger å være redd for masse syntaksfeil. Lindstrøm er også bekymret for at syntaksfeil kan ha innvirkning på eksamen.

> For at dette skal kunne komme på skriftlig eksamen så må nok læreplanmålene være enda mer spesifikke enn de er i dag, så må man bli enig om formatet. Programmering er jo litt sånn sjansespill, du kan jo sitte i 2 timer for å finne at du har gal type parentes, og jeg er litt redd for at det skal ha innvirkning på eksamen. At hvis du stryker på et helt rett program, men du har feil parentes. Viktig å bedømme programmenes oppbygning, og ikke vekte syntaksfeil i særlig grad.

Eksamen vil ha innvirkning på hvor mye tid lærere vil bruke på programmering er tydelig, dette viste også erfaring med implementeringen av Geogebra. Det argumenteres for at vurdering av programmeringsferdighetene kan gjøres på flere ulike måter, i tillegg til at man ikke må ha fokus på syntaksfeil, men at det er selve oppbygging av programmene som er viktig. Det er en fordel å bli enige om et programmeringsspråk som skal benyttes i skolen, men man frykter at det kan bli vanskelig da dette ble et dilemma også på 1980-tallet. Eksamen, i tillegg til lærerkompetanse, vil derfor være avgjørende faktorer for om man vil lykkes med implementeringen av programmering.

6 Diskusjon

Jeg vil i dette kapitlet av oppgaven drøfte mine resultater. Med utgangspunkt i forskningsspørsmålene:

1. *Hvilke argumenter kan brukes for eller imot innføring av programmering i matematikkfaget?*

2. *Hvilke forutsetninger er viktige for å lykkes med implementeringen av programmering?*

vil drøftingen ta utgangspunkt i mine resultater fra dokumentstudiet og intervjuene med informantene, og vil belyses med relevant teori. I delkapittel 6.1 vil jeg drøfte de seks områdene jeg gjennom kodingsprosessen kom frem til som argumenter rundt debatten om innføringen av programmering i matematikkfaget. I delkapittel 6.2 vil jeg drøfte hvilke forutsetninger som informantene mente er viktige for at implementeringen av programmering i matematikkfaget skal lykkes.

6.1 Argumenter for eller imot innføringen av programmering i skolen.

Etter å ha studert offentlige utredninger, stortingsmeldinger og andre relevante dokumenter er det påfallende lite som skulle tilsi at programmering skulle komme inn som en del av matematikkfaget. Det er tydelig at dette ikke er etterspurt av det matematiske fagmiljøet når en ser på oppsummeringene av innspillene til kjerneelementene. Det har i stedet vært fokus på redusering av matematikkpensum, da dette anses til å være for stort i dagens læreplan. Programmering er så vidt nevnt i de offentlige dokumentene som legger føringer for fagfornyelsen og for Stortingsmelding nr. 28 (2015-2016). Koding og programmering er blitt svært populært, og i Norge, som mange andre land, er det de siste årene dukket opp mange såkalte kodeklubber, der barn og unge kan lære programmering. På internett er det også mange ulike nettressurser og spill der programmeringsferdigheter kan læres. Utfor Norges grenser er det tydelig at programmering er i vinden og det er blitt innført i mange læreplaner i Europa, også i Skandinavia. Ifølge Erfjord og Haara (2018) gir den teknologiske utviklingen nye muligheter i matematikkundervisningen, og den gir konsekvenser man på relativt kort sikt ikke kunne forutse. Papirbasert lærebok og bruk av arbeidsplaner er fortsatt styrende for elevers arbeid med matematikk i skolen, til tross for alle mulighetene som den teknologiske utviklingen gir. Det å være åpen for teknologiske endringer i fag er viktig, og innføringen av programmering i matematikkfaget er eksempel på dette (Erfjord & Haara, 2018).

Programmering anses som en viktig kompetanse for fremtiden av både næringsliv og andre aktører, og dette er også noe informantene ser på som et viktig argument for innføringen av programmering i læreplanen også i Norge. Det hevdes igjen at programmering forbedrer kognitive ferdigheter som f.eks. kreativitet og problemløsning. I dag har nyere forsking (se kapittel 2.6) kommet frem til at dataprogrammering gir overføringsverdier til andre områder, også kognitive, slik som Seymour Papert hevdet allerede på 1970-tallet gjennom sin satsing på LOGO-programmering.

6.1.1 Fremtidens kompetansebehov

Informantene påpeker alle endringene som er i samfunnet, i skolen og i fagene, er en følge av blant annet utvikling av ny digital teknologi. Ifølge rapporten P*rogrammering i skolen* er det

fremtidige arbeidslivet avhengig av teknologisk kompetente arbeidstakere (Selvik m.fl., 2016). Flere av informantene trekker frem den digitale teknologiutviklingen og fremtidens behov for kompetanse, både i studie- og i yrkessammenheng, som et viktig argument for hvorfor programmering nå bør innføres som del av læreplanen. Dette er i tråd med det f.eks. Sanneutvalget presiserer i sin rapport der de viser til at programmering i skolen ofte begrunnes med at det er nødvendig kompetanse for å lære, arbeide og leve i dagens og morgendagens samfunn. Elevene skal ikke bare fungere i samfunnet, men også sette preg på, utvikle og gi retning til framtidens samfunn. I denne sammenhengen er det helt sentralt at elevene får innsikt i og erfaring med grunnleggende teknologiske prinsipper og digital teknologi, inkludert programmering. (Sanne m.fl., 2016). Forståelse for og kompetanse i programmering er viktig for å kunne skape eller produsere noe digitalt, og kunnskap om hvordan digital teknologi fungerer er vesentlig i et digitalt samfunn.

Da programmering først ble forsøkt innført i skolesammenheng på 1980-tallet var ikke tilgangen på PC og programvare slik den er i dag, og i tillegg var det svært dyrt. I dag har alle elever i den videregående skole PC-er, og også i ungdomsskolen er tilgangen til PC god. Programmerings-språkene er også blitt adskille enklere og utviklingen av blokkprogrammering gjør at også barn i barneskolen lettere kan lære seg programmering. Det gjør at innføring av programmering kan settes i gang tidligere, siden teknologien og programvaren allerede er på plass. Dale, Engelsen og Karseth (2011) trekker frem at økonomien i dagens samfunn er avhengig av at mennesker og institusjoner produserer kunnskaper, særlig innenfor naturvitenskap, teknolog, forskning og utvikling. De begrunner dette med at det industrielle samfunnet var konsentrert rundt produksjon av ting, mens arbeidskraften i dag i økende grad er konsentrert rundt tjenesteytelser, idéproduksjon og kommunikasjon. Programmering i skolen kan være med på å inspirere elever til å bli interesserte i teknologiske fag, slik at dagens skoleelever i fremtiden kan være med på å utvikle ny teknologi som er viktig for samfunnet (Kaufmann, Stenseth & Holone, 2018). Erfjord og Haara (2018) mener likevel at det ikke alltid er slik at skolens utvikling må gå i takt med samfunnsutviklingen. Av og til kan det ha vært gode grunner for skolen å opponere mot utviklingen, eller finne et alternativ, men slike alternativer er ikke så lett å se for seg i forbindelse med den teknologiske utviklingen som samfunnet gjennomgår nå. Utviklingen er så rask og omfattende i skolen, og i andre samfunnsinstitusjoner, at man i skolen heller bør innrette seg etter denne utviklingen, og bidra til at elevene i størst mulig grad er digitalt kompetente for de behovene som fremtidens samfunn trenger (Erfjord & Haara, 2018).

Så selv om programmering ikke er nevnt i stor grad i offentlige dokumenter, så kan det se ut som at både samfunnet og næringslivet, tydelig har gitt signaler om at satsing på teknologi og programmering er viktig. Andre land har allerede innført programmering, og Norge er sist av de skandinaviske landene til å få dette inn i læreplanene. Dermed kan det argumenteres for at innføringen av programmering i Norge er i tråd med både samfunnsutviklingen og utviklingen av digital teknologi generelt.

6.1.2 Stofftrengsel

Det er klart at innføringen av programmering i læreplanene på en eller annen måte må føre til stofftrengsel i skolen uavhengig av om det innføres som eget fag eller som del av læreplan til et eksisterende fag er, så fremt ikke timetallet i skolen økes. Regjeringen bestemte at det ikke var ønskelig med nytt fag i skolen. Opprettelse av et nytt fag ville fått konsekvenser for andre fag, siden timer til programmeringsfaget måtte tas fra eksisterende fag- og timefordeling slik at

andre fag ville blitt skadelidende. Det er bestemt at matematikkfaget skal ha hovedansvar for opplæringen av programmering, slik at det også kan brukes i andre fag som naturfag, kunst og håndverk og musikk. Programmering som del av eksisterende fag, er en løsning som både Sverige og Finland allerede har valgt i revideringen av sine læreplaner. På denne måten kan elevene og lærerne begynne i det små, og samtidig knytte programmering til noe kjent i matematikken, der matematikken er i fokus. Senter for IKT støtter valget om innføringen av programmering som del av matematikkfaget, og begrunner dette med at det er mest realistisk i dag. Men de understreker likevel at på lengre sikt bør programmering, innføres som eget obligatorisk fag i grunnskolen, noe som også flere av informantene trekker frem. Dette er i tråd med det Sanneutvalget skriver i sin rapport. De påpeker at hvis elevene først lærer programmering i et eget fag, vil det være lettere å integrere det i de etablerte fagene (Sanne m.fl., 2016).

Programmering og teknologi som et eget fag vil kunne sikre at alle elever får en grunnleggende opplæring i programmering, noe som igjen vil gi mulighet til faglig fornyelse av de øvrige skolefagene (Selvik m.fl., 2016). Bakgrunnen til at Sanneutvalget konkluderte med at programmering og teknologi burde bli et eget fag var bl.a. det faktum at tidligere forskning på emnet *teknologi og design* i grunnskolen viste en nedprioritering av emnet i naturfaget. Dette forklares med at emnet ikke var lagt inn som eget emne eller fag, men skulle integreres gjennom arbeid med andre naturfaglige emner. Teknologi og programmering integrert i eksisterende fag vil ifølge rapporten risikere å bli systematisk nedprioritert på lik linje, både fordi lærerne ikke opplever å ha nok kompetanse, og fordi det ikke passer naturlig inn i fagets kultur (Sanne m.fl., 2016). Det er selvfølgelig en fare at programmering ender opp «stemoderlig» behandlet i matematikkfaget, slik som teknologi og design er blitt i naturfaget. Dermed er det viktig å være bevisst dette, spesielt siden lærerkompetansen innen programmering lav, og det kan av den grunn være vanskelig å se mulighetene som programmering kan gi matematikkfaget.

Kaufmann m.fl. (2018) argumenterer med at datamaskinens økende kraft og funksjoner, gjør at det er viktigere nå enn før å kunne tenke systematisk på algoritmer og hvordan datamaskinen kan bidra til å løse komplekse problemer. Algoritmisk tenkning er viktig både i programmering og matematikk, det argumenteres også med at programmering er med på å utvikle elevers logiske tankegang og evne til problemløsning, noe som også er betydningsfullt for matematikkompetansen til elevene. Det er derfor naturlig at matematikkfaget fikk hovedansvaret for opplæringen av programmering i skolen, og muligens er dette den beste løsningen, siden det ikke ble aktuelt å opprette et eget teknologi- og programmeringsfag. Dette er også noe flere av informantene poengterer, og de påpeker i tillegg at programmering kan gi nye muligheter i faget og på sikt vil faget ha mye å vinne på dette ved å knytte programmering opp mot matematikk, noe jeg vil komme tilbake til i neste delkapittel om dybdelæring. Siden innføringen av programmering endte opp i matematikkfaget, fikk nok flere av dem som argumenterte med at innføringen av programmering ville føre til stofftrengsel rett. Hvis man studerer det nye læreplanutkastet er det tydelig at antall kompetansemål er redusert i forhold til LK06, og at noen emner er tatt bort. Dette gjelder i alle matematikkfagene i videregående skole. Som nevnt tidligere er geometridelen i R1 tatt bort, og i R2 er differensiallikninger fjernet. I tillegg er området sannsynlighet tatt ut av 1P, 1T, R1. I S1 og S2 er det fortsatt sannsynlighet i læreplanen og nivået er blitt høyere, og det virker ikke som om læreplanen her er spesielt redusert. En av årsakene til fagfornyelsen og utarbeidelsen av nye læreplaner var å redusere mengden fagstoff. Programmering har nok derfor ikke alene skyld i at en del emner forsvinner

fra læreplanene. For å få til dybdelæring i matematikkfaget, må man ha nok tid, og da må noe fagstoff bli fjernet uavhengig om programmering innføres eller ikke i faget.

6.1.3 Dybdelæring

Om programmering vil fremme eller hemme dybdelæring er nok det som er mest omdiskutert, både i media, høringsinnspill og av informantene. Hvis man ser på læreplanutkastet er geometri og differensiallikninger nå fjernet fra læreplanen på R1 og R2. Ifølge Kristensen som sitter i læreplangruppa, en dette en konsekvens av at programmering er innført og det må frigjøres tid til dette. Disse emnene blir av bl.a. Lindstrøm sett på som en del av dybdelæringen i programfagene i matematikk, der elevene får se sammenhenger mellom emner som f.eks. integrasjon og derivasjon. Differensiallikninger kan bidra til større begrepsmessig forståelse for derivasjon og integrasjon, ved at elevene gjennom differensiallikninger klarer å se matematiske relasjoner mellom disse konseptene. Ved å fjerne differensiallikninger fra pensum, vil elevene ikke få mulighet til å utvikle disse relasjonene på samme måte som tidligere. På den andre siden vil innføringen av programmering gi mulighet til å undervise i f.eks. derivasjon på en annen måte enn tidligere. Ved å lage programmer som bruker definisjonen av den deriverte, i stedet for å fokusere på regler, kan elevene oppnå det Skemp (1976) kalte for relasjonell forståelse (kapittel 4.3.1) for konseptet derivasjon. For å lage et program som fungerer må elevene kunne matematikken som ligger bak programmet, og på denne måten få en dypere innsikt om hvorfor derivasjonsreglene fungerer, noe Haraldsrud selv hadde erfart. Gjennom å bruke for eksempel numeriske metoder kan elevene lage programmer som gir dem mulighet til å løse funksjoner som ikke lar seg derivere med de vanlige derivasjonsreglene som læres i videregående skole. Slik kan programmering gi lærerne mulighet til å gi elevene virkelighetsnære problemstillinger som ikke kunne løses tidligere.

Selv om prosedyrekunnskap om derivasjon og integrasjon er viktig, slik at enkle funksjoner lett kan løses for hånd ved å bruke regler, eventuelt ved bruk av CAS, så mener flere av informantene at elevene, gjennom å lage programmer, kan utvikle en dypere forståelse i matematikkfaget. Elevers ferdigheter i å lage programmer selv trekkes frem som sentralt for at programmering skal føre til dybdelæring, og ikke få ferdige programmer, eller oppskrifter, som de kan bruke på eksamen eller i undervisningen. Det å lære å lage egne programmer krever kunnskap om programmering, noe som vil ta tid å lære seg, men når elevene har utviklet denne kompetansen vil det gi elevene muligheter til å løse mer komplekse problemstillinger, ikke bare i matematikkfaget. På denne måten vil elevene opparbeide seg fagspesifikk kompetanse i matematikk gjennom programmeringsarbeid, samtidig som de kan få bedre ferdigheter i programmering.

Dybdelæring handler også om det å anvende kunnskap. Anvendelse (eller strategisk tankegang) knyttes til problemløsningskompetanse og det er det sammenheng mellom algoritmisk tenkning og Pólyas fire hovedpunkter innen problemløsning. Programmering er mer allsidig enn man tror, og anvendes i bl.a. spill, finans, lyd, bilde, film og robotikk (Haraldsrud, 2018). Gjennom programmering er det mulig å lage simuleringer som viser hvordan et system oppfører seg ved hjelp av matematiske modeller, noe som er svært relevant spesielt innen naturvitenskapelige fag. Tid til å jobbe med ulike matematiske emner er unektelig en viktig faktor for å oppnå dybdelæring, og selv om det ser ut som at pensum er redusert, så vil arbeid med programmering ta tid, både å lære seg språket og selve prosessen med å lage programmene. Så om lærerne og elevene totalt sett har fått bedre tid i faget gjenstår å se.

6.1.4 Algoritmisk tenkning

For å klare å lage gode nok koder og algoritmer for å få en datamaskin til å utføre det man ønsker, kreves kompetanse i algoritmisk tankegang og problemløsning. Det påstås at programmering er med på å utvikle elevenes logiske tankegang og evne til problemløsning, noe flere forskningsartikler også har antydet (kapittel 2.6). Før elevene kan løse komplekse problemer ved hjelp av programmering må elevene få kunnskaper og ferdigheter i programmering og de må ha kompetanse i algoritmisk tenkning som problemløsningsstrategi. Etter dette kan elevene ifølge Van De Walle m.fl. (2014) lære gjennom problemløsning. Elevene må ta i bruk både programmerings- og matematiske ferdigheter og de må anvende løsningsstrategier for å løse aktiviteter knyttet til problemløsning. Gode aktiviteter gir elevene kompetanse i å utforske, og selve problemløsningsprosessen er viktig slik som Nygaard m.fl. (1999) påpeker. Mange programmeringsaktiviteter fordrer til en utforskende måte å jobbe på der elevene kan se på ulike innfallsvinkler, diskutere, feile og teste nye hypoteser. På denne måten vil hele prosessen være viktig, ikke bare løsningen på problemet. I arbeid med programmeringsaktiviteter benyttes ofte den samme firepunktsstrategien som Pólya presenterte i boka *How to solve it* allerede på 1950-tallet for å komme frem til en løsning. Gode programmeringsaktiviteter har også sjelden kun en løsning. Elevene må ofte prøve seg litt frem, være kreative og teste ulike koder og lage sine egne algoritmer. Dette kalte Papert for bricolage (kapittel 3.2.4). Det er likevel viktig at elevene jobber systematisk, og vurderer løsningene sine etter hvert i arbeidet, noe Pólya også påpekte i sin problemløsningsstrategi. Problemløsning og kreativ tenking er viktige ferdigheter for fremtiden, noe både NOU 2015: 8 og 21st Century skills påpeker. Gjennom programmering og debugging får elevene trening i algoritmisk tenkning, da slik tenkning er sentral for å lykkes med problemløsningsoppgaver. Det er også en av grunnene til at algoritmisk tenkning nå er tatt inn som en del av problemløsningsstrategiene i kjerneelementene i matematikkfaget.

Det kommer frem i både høringsinnspillene og av informantene at algoritmisk tenkning er et noe utydelig begrep, og kan misforstås. Både Lindstrøm, Tellefsen, Stenseth og Kristensen trekker frem at den norske oversettelsen ikke er helt heldig, noe som kan være en av årsakene til så mange diskusjoner om begrepet. Blant annet i høringsinnspillene er det flere som konsentrerer seg om begrepet «algoritme» og er frykter at dette begrepet kommer til å føre med seg mer pugging av formler og bruk av oppskrifter. Både Kristensen og Haraldsrud er tydelige på at det er det motsatte som er meningen med begrepet. Elevene skal bygge opp og utvikle egne algoritmer for å løse problemer, ikke bruke ferdige oppskrifter. Det er meningen at algoritmisk tenkning skal være en del av elevenes problemløsningsstrategi, og at elevene gjennom algoritmisk tenkning skal klare å løse større problemer, gjennom å dele disse opp i mindre deler, utvikle modeller og egne algoritmer for å finne løsninger. For at elevene skal få til dette kreves det at de kan resonnere abstrakt, representere og modellere problemer med matematiske symboler og algoritmer, og at de er presise i kodingsprosessen og klarer å se etter struktur og regularitet (Kaufmann m.fl., 2019).

6.1.5 Digitale verktøy

Gjennom LK06 ble det å beherske digitale verktøy innført gjennom digitale ferdigheter i bl.a. matematikk, og i kompetansemålene ble det spesifikt beskrevet hva elevene skulle beherske. I dag er det Geogebra, CAS og regneark som dominerer som digitale verktøy i matematikkundervisningen, i tillegg til kalkulator. Det påpekes av både Lindstrøm og Haraldsrud at innføringen av programmering vil kunne skape utfordringer for elever, og lærere,

ved at det blir for mange digitale verktøy for elevene å forholde seg til. Informantene er likevel litt uenige om det kan bli for mange digitale verktøy å mestre for elevene, og om man bør beholde CAS eller ikke. Flere av informantene trekker frem at programmering kan gi en annen vinkling til matematikken enn de dominerende digitale verktøy som Geogebra og CAS. Lindstrøm ser på bruken av CAS som en «sort boks» der elevene ikke kan se eller forstå hva som skjer. CAS gjør elevene stand til å gjøre matematiske beregninger, men vil i liten grad hjelpe elevene å forstå matematikk. I matematikkundervisningen er det å visualisere og dynamisk representere abstrakte konsepter til stor hjelp for å bygge opp forståelse hos elevene og digitale verktøy er godt egnet til dette (Egeberg, Hultin & Berge, 2016). Ved å lage egne programmer kan programmering derfor bidra med å skape en bedre forståelse for faget, slik som Drijvers m.fl. (2011) også påpeker. Da bruker man teknologien for å utvikle begrepsforståelse. Kristensen tror at elevene vil takle å forholde seg til flere verktøy, men at det er viktig at de får tid til å lære seg disse. *Monitor 2016* er den syvende undersøkelsen om bruk av digitale verktøy i skolen, lærernes og elevenes digitale kompetanse og skoleleders prioriteringer, og denne undersøkelsen vektlegger også bruken av IKT i matematikkfaget. Elevene som deltok i undersøkelsen opplyste at de hadde jevnt over god tro på egne digitale ferdigheter, men når det kom til bruk av bl.a. regneark så viste undersøkelsen av elevene var mer usikre på sine ferdighetene, noe undersøkelsen også testet til å stemme godt. Dette kan ha sammenheng med at det i denne undersøkelsen kommer frem at matematikkfaget er det faget der elevene bruker minst tid på digitale verktøy og tidsbruken i timene er hovedsakelig knyttet til bruk av lærebok og digitale resurser knyttet til læreboka (Egeberg m.fl., 2016).

Hvis elevene skal lære å bruke og vurdere digitale verktøy, må læreren sette av tid slik at elevene får erfaring med hvordan slike verktøy kan være nyttig, og bidra til å gjøre oppgaveløsning enklere og mer effektivt. Kristensen trekker frem valg av undervisningsmetoder som en annen viktig faktor, og da må mer varierte metoder tas i bruk i klasserommene, enn det som *Monitor 2016* undersøkelsen viser er normalen i norsk skole. I trådmodellen til Kilpatrick m.fl. (2001) er anvendelse en av kompetanseområdene. Elevene må opparbeide seg kompetanse i å kunne bruke de mest hensiktsmessige metodene å løse problemer på. Digitale verktøy kan og bør i mange tilfeller brukes for å effektivt løse oppgaver, som f.eks. i oppgaver der derivasjon og integrasjon skal brukes i videregående skole. Hvis elevene har tilgang til fire ulike digitale verktøy er det viktig at de lærer å bruke disse godt, for å kunne vurdere hvilket verktøy som er mest hensiktsmessig. Dette krever selvsagt at det brukes mye tid på innlæring, og det kreves god kompetanse i bruken av verktøyene, også hos læreren.

I arbeidet med programmering oppstår det ofte feil i koden man skriver, og det å oppdage og rette opp i feilen kalles for feilsøking eller debugging[10]. Feilsøking er noe alle informantene trekker frem når det kommer til programmering. Tellefsen sier at hun kan slippe seg mer «fri» når hun programmerer, og ikke er så opptatt at alt skal være riktig. Det å gjøre feil er uunngåelig, og gjennom arbeid med programmering blir man ikke opptatt av dette, noe som er mer vanlig i tradisjonell matematikkundervisning der feil kan bidra til dårlig selvtillit i faget for mange elever. En programmerer må kunne finne og tolke feil i koder. Å lære av feilene vil være en naturlig del av prosessen, og dette ses ikke på som negativt. At programmering kan endre elevenes holdning til å gjøre feil, var noe Papert påpekte allerede i boka *Mindstorm*:

[10] Store norske leksikon, s.v. «Feilsøking» 20.06.18 https://snl.no/feils%C3%B8king_-_programmering

> The process of debugging is a normal part of the process of understanding a program. The programmer is encouraged to study the bug rather than forget the error. And in the Turtle context there is a good reason to study the bug. It will pay off. (Papert, 1980, s. 61)

Hvordan elevers feil kan brukes til å lære avhenger av lærerens kompetanse, og en god matematikklærer klarer å utnytte elevers feil og misoppfatninger på en måte som bidrar til læring. Slurvefeil er en type feil som ikke alltid er like lett å oppdage, men som kan skape store problemer for løsningen. I programmering får man opp feilkoder, som gjør at feil kan oppdages på et tidligere tidspunkt, og det blir normalt å ikke få alt riktig med en gang. Gjennom debugging lærer elevene å se tilbake, og lete etter feil ved å dele opp problemet, og dette er også overførbart til problemløsning der disse egenskapene også er svært viktige.

6.1.6 Sosiale og emosjonelle kompetanser

Det kan se ut som at programmering kan gi både økt motivasjon og utholdenhet og dette er noe som trekkes frem av flere av informantene. Problemløsning med programmering kan gi andre innfallsvinkler, og man kan undersøke problemstillinger som ikke var mulig tidligere. Dette kan bidra til å skape entusiasme og motivasjon. Stenseth trekker frem under intervjuet en metode som ofte brukes i informatikk som kalles parprogrammering, der elevene arbeider i par og veksler på å skrive kode og se på. Han har selv erfart at denne metoden skaper entusiasme og motivasjon, samt fordrer til samarbeid. Elevene må resonnere sammen, og bruke det Lithner (2008) kalte for kreativ resonnering (kapittel 3.3.5) der elevene selv skaper noe sammen, og ikke er bundet til en bestemt fremgangsmåte.

I programmering er det ofte slik at eleven selv må finne ut av, og lage algoritmen, eller koden, som får programmet til å fungere. Dette avhenger likevel av hvordan man legger opp arbeidet med programmeringen. Samarbeid og diskusjon synes som viktige deler av arbeidet med programmering, og det understrekes at god veiledning og support er viktig for at elevene skal lykkes. Et fungerende program som sluttproduktet, kan være med på å bidra til å øke elevenes utholdenhet. Norske elever har dårligere utholdenhet sammenlignet med elever fra resten av landene som deltar i PISA-undersøkelsen, noe undersøkelsen fra 2014 viste (Ertesvåg, 2014). Hvis programmering kan bidra til økt motivasjon og utholdenhet hos elevene, og i tillegg lærer dem mer matematikk og økt kompetanse i å løse kognitivt utfordrende oppgaver gjennom programmeringsaktiviteter, er dette utdelt positivt. Programmering i skolen skal ha et sosialt aspekt, og elevene skal jobbe sammen for å finne løsninger, noe som kan bidra til at elevene ikke gir opp like lett. Forskning viser likevel at elever som samarbeider med problemløsnings-oppgaver, uten å bruke programmering eller liknende, konkluderer med at også disse elevene fikk økt motivasjon og utholdenhet. Det var ikke en signifikant forskjell mellom de elevene som arbeidet med problemløsningsaktiviter uten programmering, og de elevene som brukte programmering i arbeidet (kapittel 2.6). Dette kan tyde på at undervisningsmetodene i seg selv har større betydning for de sosiale- og emosjonelle kompetansene enn selve programmeringsaktiviteten i seg selv.

6.2 Forutsetninger for å lykkes

To faktorer som utmerket seg som helt vesentlige for å lykkes med implementeringen av programmering i matematikkfaget: Lærerens kompetanse, og eksamen. Spesielt lærerkompetanse utpekes som den viktigste faktoren for å lykkes.

6.2.1 Lærerkompetanse

Informantene enige om at lærerens kompetanse er den viktigste faktoren for å lykkes med implementeringen av programmering i matematikkfaget. Både lærerens faglige kompetanse og den didaktiske kompetansen er avgjørende for om læreren klarer å se mulighetene programmering og algoritmisk tenkning kan gi i matematikkundervisningen, og på en pedagogisk måte klare å presentere dette for elevene. På 1980-tallet ble det påstått at datamaskinen en dag kunne overta for læreren og gjøre slutt på dagen skole (Sanne m.fl., 2016), noe den amerikanske NCTM (National Council of Teachers of Mathematics) tilbakeviste:

> Technology cannot replace the mathematics teacher, nor can it be used as a replacement for basic understandings and intuitions. The teacher must make prudent decisions about when and how to use technology and should ensure that the technology is enhancing students' mathematical thinking (NTCM, sitert i Scandrett, 2008).

Drijvers (2015) tydeliggjør læreren, i tillegg til oppgavenes design og undervisningskontekst, som avgjørende for å lykkes med bruk av digital teknologi i undervisningen. Han understreker at bruken av digital teknologi aldri kan erstatte læreren, og at lærerens rolle er å fremheve fruktbare verktøyteknikker, vise effektiv bruk av teknologi i matematikkaktiviteter, kontra bruk av pen og papir. For å kunne gjøre dette er det nødvendig med profesjonell utvikling, som inkluderer utviklingen av lærerens teknologiske og pedagogiske kunnskap (Drijvers, 2015). Innen programmering er lærerkompetansen lav. Bruk av fagspesifikk digital teknologi i fag er avhengig av lærerens digitale kompetanse, og elever trenger digitalt kompetente lærere som rollemodeller (Sanne m.fl., 2016). Opplæring i programmering har ikke vært en obligatorisk del av lærerutdanningen tidligere, og etter- og videreutdanningskursene som har vært tilbudt de siste årene av utdanningsdirektoratet har ikke hatt programmering som tema frem til nå. I dag tilbyr flere universiteter kurs innen programmering, bl.a. NTNU, Høgskolen på Vestlandet, Universitetet i Stavanger og Universitetet i Oslo. Noen har kurs med studiepoeng, andre mindre kurs, eller MOOC som er åpne for alle. Bruken av MOOC stiller bl.a. Kristensen og Tellefsen seg kritiske til, spesielt hvis kurset ikke er knyttet til faget.

Tellefsen, Kristensen og Lindstrøm understreker at det er viktig at programmering knyttes opp mot fagets egenart, og ikke blir en adskilt del fra resten av faget. For å få dette til er videreutdanning og intern kursing viktig, både med fokus på didaktikk og faglig opplæring av programmering. Lærerne må samarbeide og dele erfaringer, og ha fokus på det faglige. Det trekkes frem at programmering bør ses på som et nytt og nyttig verktøy, og at programmeringen må skje på fagets premisser. Lærerne må føle at det er deres fag som er viktig, og ikke programmeringen i seg selv. Dette er i tråd med rammeverket Senter for IKT har laget for lærerens profesjonsfaglige digitale kompetanse (PfDk), som har et tosidig siktemål: Det ene handler om profesjonsutvikling, det andre om selve profesjonsutøvelsen. For å være i stand til å utvikle de grunnleggende ferdighetene og fagkunnskap hos elevene må lærere utvikle sin egen profesjonsfaglige digitale kompetanse i lærerutdanningen og videre gjennom profesjonell læring og utvikling i løpet av sin yrkeskarriere (Kelentric m.fl., 2017). For å få dette til er det viktig å «skynde seg sakte» som Tellefsen understreker. Dette kommer til å ta tid, og å tro at alle lærere er klare for dette allerede høsten 2020 er nok ikke rimelig, derfor er det nok viktig at man gjennom samarbeid finner de gode eksemplene og oppleggene og starte i det små. I tillegg må lærerne få nok tid til å lære seg dette, og at det legges opp til dette fra skoleledelsen.

6.2.2 Eksamen

Den andre faktoren for å lykkes med programmering er at det på sikt tas inn på eksamen på en eller annen måte. Kristensen viser til erfaringen med Geogebra, at først når Geogebra våren 2015 ble obligatorisk på sentralgitt eksamen, at bruken av dette verktøyet ble tatt alvorlig blant lærere. Dette på tross av faktumet at bruken av dynamisk programvare var allerede ført i læreplanen allerede i 2012. Eksamen er styrende for hva som undervises, og både lærere og rektorer mener at eksamen er viktig for elevenes læringsutbytte. Hva som vurderes, og hvordan det blir vurdert, påvirker opplæringen og dette omtales gjerne som washbackeffekten (Allerup, Kovac, Kvåle, Langfeldt & Skov, 2009). Det antas at eksamen i matematikk har sterk washbackeffekt, siden eksamen er viktig både for elever og lærere, og eksamen er derfor et viktig verktøy for å implementere læreplanen (Borge m.fl., 2014). At implementeringen av programmering er avhengig av en eller annen form for sentralgitt vurdering er derfor klart, noe også informantene trekker frem. Dersom en skriftlig eksamen makter å teste store deler av elevenes kompetanse, vil den påvirke undervisningen på en god måte (Borge m.fl., 2014).

Med tanke på at vi nå får fire digitale verktøy i faget, kan dette være vanskelig å gjennomføre på en fem timers eksamen. I tillegg viser Borge m.fl. (2014) til at en eksamen generelt gir et begrenset bilde av elevenes reelle læringsutbytte, da de er i en presset situasjon noe som kan sette begrensinger for deres kreativitet. Lindstrøm foreslår at den muntlige eksamen kan vare et sted å begynne i forhold til vurdering av programmeringsferdigheter. I fagfornyelsen er vurdering og eksamensform under revidering, og hvordan eksamen i matematikk blir etter innføringen av fagfornyelsen er fortsatt ikke sikkert. I dokumentet «Matematikk for alle, men alle trenger ikke å kunne alt» (Botten-Verboven m.fl., 2010) anbefalers det at det innføres standpunktkarakter i muntlig matematikk for å tvinge frem mer varierte arbeidsmetoder i faget. Innføringen av programmering kan også bidra til økt variasjon, og programmeringens egenart fremmer kommunikasjon og diskusjon, så at dette kan testes muntlig kan være et godt forslag.

Det vil være en fordel om det blir enighet om ett eller i alle fall få programmeringsspråk som brukes i opplæringen, men dette kan bli vanskelig, noe Stenseth trekker frem fra debatten som gikk på 1980-tallet. At det finnes mange ulike programmeringsspråk, og at det fort kan gå mye tid til debugging taler mot at programmering bør testes på en sentralgitt eksamen, da må i alle fall programmenes oppbygging og ikke syntaksfeil vektlegges. Hvordan eksamen kommer til å legges opp i den videregående skolen er enda ikke klart. Det er satt ned et eksamensutvalg som skal levere sin endelige rapport i 2020 som skal inneholde forslag om nye eksamensformer og videre retning for arbeidet med sluttvurdering i skolen.

Som diskusjonskapitlet viser er det mange gode argumenter for å innføre programmering i skolen, selv om programmering også fører med seg endel utfordringer for lærere og elever. Programmering er en kompetanse som anses som fremtidsrettet og kan bidra til dybdelæring, kompetanse i algoritmisk tenkning og kunne påvirke elevers sosiale og emosjonelle kompetanser. Det er felles enighet om innføringen vil medføre stofftrengsel, og for noen elever kan det bli for mange digitale verktøy å forholde seg til. Læreren og lærerens profesjonsfaglige digitale kompetanse trekkes frem som de viktigste forutsetningene for at implementeringen skal lykkes. I tillegg vil eksamen ha betydning for i hvor stor grad verktøyet vil tas i bruk i undervisningen, pga. eksamens sterke washbackeffekt.

7 Avslutning

7.1 Konklusjon

Det matematiske fagmiljøet ble «tatt på senga» da innføringen av programmering gjennom begrepet algoritmisk tenkning blir en realitet i faget matematikk. Begrepet algoritmisk tenkning har fått stort fokus, og er et begrepet det fortsatt råder uklarhet om. Både for meg og andre matematikklærere vil denne innføring skape konsekvenser, og jeg ville derfor finne ut hvilke argumenter som kan brukes både for og imot innføringen av programmering. Siden innføringen er et faktum og allerede fastsatt i kjerneelementer og læreplan, så ville jeg også fokusere på hvordan man på best mulig måte kan lykkes med implementeringen av programmering i matematikkfaget.

Jeg utarbeidet derfor to forskningsspørsmål:

1. *Hvilke argumenter kan vi finne for eller imot innføring av programmering i matematikkfaget?*

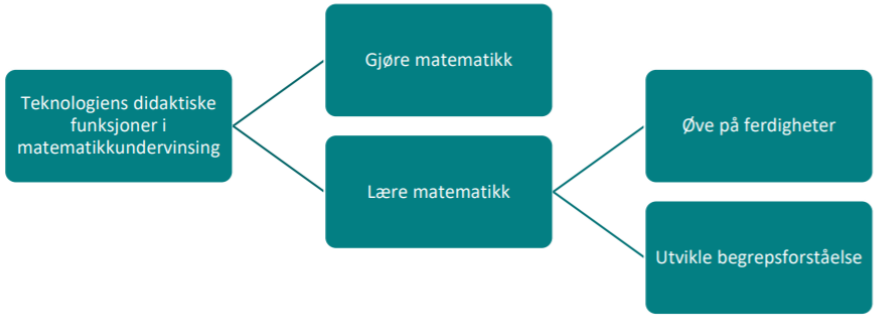
2. *Hvilke forutsetninger er viktige for å lykkes med implementeringen av programmering i matematikk?*

Resultatene fra dokumentstudiet og intervjuene viser at det er mange gode argumenter både for og imot innføringen av programmering i matematikkfaget. Innføringen av programmering kan bidra til økt motivasjon og utholdenhet i et fag som mange elever i dag sliter med og finner uengasjerende. Dagens tradisjonelle undervisningsform kan føre til instrumentell forståelse i et fag der relasjonell forståelse er sentralt for at man skal kunne oppnå kompetanse for fremtiden. Programmering gir elevene et innblikk i hvordan teknologien rundt dem er bygd opp, og selv om ikke alle skal blir programmerere, er algoritmisk tenkning en viktig problemløsningsstrategi som løftes eksplisitt frem av kjerneelementgruppa i matematikk. Matematikken er full av oppskrifter og presise framgangsmåter, for eksempel de grunnleggende aritmetiske algoritmene. Det å lære å mestre, og ikke minst forstå disse algoritmene trekker arbeidsgruppen frem som viktig. Viktigere er det at de gjenkjenner hva en slik presis oppskrift er og blir i stand til å utlede og designe egne oppskrifter som løser relevante problemer (Kjerneelementgruppen for matematikk, 2017b).

At programmering og teknologi er så viktig at det egentlig skulle blitt utarbeidet et nytt fag som fokuserte på dette har både utvalg og fagmiljøer understreket, men det ble avgjort på politisk hold at å etablere et nytt fag ikke var ønskelig. At programmeringsopplæringen derfor ble lagt til matematikkfaget er et godt alternativ, noe både informantene og Senter for IKT mener. Nyere forskning viser at programmering kan ha positiv effekt for elevenes læringsutbytte, og informantene trekker frem at programmering kan føre til dybdelæring. Gjennomført på en god måte kan derfor programmering gi en positiv fornyelse av matematikkfaget i skolen (Kjerneelementgruppen for matematikk, 2017b).

De fleste er enige i at programmering vil føre til stofftrengsel, siden både opplæringen og selve programmeringsaktiviteter tar tid. Når man ser på den nye læreplanen i matematikk er flere emner tatt ut av matematikkfagene i videregående skole, bl.a. plangeometri i R1 og differensiallikninger i R2. Det er likevel usikkert at det kun er innføringen av programmering

som er årsaken til disse endringene, da fagfornyelsens utgangspunkt blant annet hadde som mål å redusere omfanget av gjeldende læreplan for å gi elevene tid til å gå i mer i dybden. Selv om programmering fører til stofftrengsel, er innføringen ikke bare negativ, da læreren får et nytt kraftigfullt digitalt verktøy som kan føre til en annen vinkling av f.eks. forståelsen av derivasjon og integrasjon for å nevne noe. Datamaskiner med enorm regnekapasitet, har gjort at beregninger som tidligere var svært ressurskrevende eller umulig, i dag kan utføres raskt og effektivt. Teknologi har mange bruksområder innen matematikk, og teknologien har også endret fagområdet matematikk (Voll & Vinje, 2018). Digitale verktøy og teknologi har i dag fått en sentral rolle i arbeidet med å visualisere og utforske matematiske aspekter og sammenhenger, og som nevnt delkapittel 3.4.3 så trekkes det frem tre didaktiske funksjoner som teknologi kan ha knyttet til matematikkundervisningen, se figur 10.



Figur 10: Teknologiens didaktiske funksjoner i matematikkundervisning. Oversatt fra Drijvers (2014), hentet fra Voll & Vinje (2018)

Datamaskiner kan programmeres til å gjøre matematikk mer effektivt enn det noe menneskekapasitet tillater, og i mange situasjoner er det ikke nødvendig å forstå hvordan programmet fungerer for å benytte verktøyet på en hensiktsmessig måte. For å ha kompetanse til å lage egne programmer må derimot elevene både øve på ferdigheter i programmering og matematikk, samt utvikle begrepsforståelse slik at de kan løse kognitive utfordrende problemer effektivt ved hjelp av bl.a. bruk av digitale verktøy. Teknologi og programmering kan på denne måten bidra til å lære matematikk. Gjennom programmeringsaktiviteter vil elevene lære seg feilsøking, og det argumenteres med at elever kan utvikle en mer positiv holdning til det å gjøre feil, enn det man ofte observerer i «tradisjonelt» arbeid med matematikkoppgaver. Å gjøre feil kan gi en gylden mulighet til å lære og forstå matematikk hvis feilene brukes på riktig måte av både elever og lærere. At det kan bli utfordrende for elevene å lære enda ett nytt digitalt verktøy, i tillegg til regneark, Geogebra og CAS, er nok sant. Nok tid og gode undervisningsopplegg er avgjørende, slik at elevene både lærer å mestre alle verktøyene godt, og i tillegg klarer å bestemme hvilket verktøy som er mest hensiktsmessig å bruke i gjeldende situasjoner.

Nøkkelen til at implementeringen av programmering i matematikkfaget skal lykkes er læreren. Dette er både informantene, andre utvalg og arbeidsgrupper enige om. Kjerneelementgruppa i matematikk mener at en grunnleggende forutsetning for at dette skal lykkes er at alle matematikklærere i grunnskolen og videregående opplæring får etter- og/eller videreutdanning som gir faglig trygghet (Kjerneelementgruppa for matematikk, 2017b). For at dette skal være mulig må lærere få tid og mulighet til å ta kurs, og også dele erfaringer og samarbeide både i og på tvers av skoler. Hvordan dette skal organiseres eller finansieres er likevel ikke klart. Det er avgjørende at læreren selv får erfare hvilke muligheter programmering kan gi i faget, slik at

lærerne selv får tro på at dette er viktig for elevenes fremtidige kompetanse, og også relevant for matematikkfaget. Programmeringsaktiviteter skal i prinsippet være en utforskende arbeidsmåte for elevene, der samarbeid og kommunikasjon er viktig, og brukt på riktig måte kan det gi læreren mulighet til mer variasjon i undervisningen. Eksamen bringes frem som en utslagsgivende faktor som påvirker hvor viktig ferdigheter i programmering blir ansett av lærere og elever. Hvis programmering ikke anses som eksamensrelevant, vil nok implementeringen ikke lykkes i like stor grad som ønskelig, og det vil bli større variasjon i hvor stor grad verktøyet blir brukt. Det trekkes frem at programmering kan gi nye vurderingsmuligheter, spesielt innenfor muntlig matematikk, da diskusjon, kreativitet og problemløsning er sentralt.

Elevene skal møte framtidas arbeidsliv og fritid med kompetanse til å kunne håndtere teknologi og de skal oppleve å ha kontroll over sine teknologiske omgivelser (Sanne m.fl., 2016). Samfunnet i dag trenger at elever utvikler seg til skapere av teknologi, og ikke bare forbrukere, og kompetanse i programmering og algoritmisk tenkning er sentralt for å oppnå dette. Lærerne må selv utvikle tro på at algoritmisk tenkning og programmering kan bidra til å gi elevene den grunnleggende kompetansen som trengs for å mestre det samfunnet de skal være en del av i fremtiden. Det er likevel viktig å ikke glemme at bruken av digital teknologi og digitale verktøy i seg selv ikke bidrar til økt matematikkforståelse. Lærerne må bruke sin profesjonsfaglige digitale kompetanse til å bruke verktøyene i de tilfellene der det er hensiktsmessig i forhold til læringsutbyttet og fagets egenart, og i mange tilfeller vil papir, blyant og en god faglig diskusjon fortsatt være nok.

7.2 Veien videre

I arbeidet med denne oppgaven har jeg fått innblikk i hvordan programmeringsverktøyet som et nytt digitalt verktøy kan gi meg, og andre matematikklærere, mange nye muligheter i matematikkfaget. Det vil likevel komme en del utfordringer i kjølvannet av innføringen som ikke må neglisjeres, f.eks. hvilket programmeringsspråk skal velges, hvilke oppgavetyper bør brukes og hvordan programmeringsferdigheter skal vurderes. De nye læreplanene skal igangsettes høsten 2020, og mange lærere og skoler er enda ikke forberedt. Det hadde vært interessant å forske videre på hvordan matematikklærere nå forberedes på innføringen av programmering i matematikkfaget og på hvilke måter og med bruk av hvilke metoder og tilnærmingsstrategier til fagets egenart opplæringen gis til lærere. Et annet interessant område for forskning videre hadde også vært å undersøke på om utdanning eller kurs i programmering gjør at læreres holdninger til programmering endres.

Referanseliste

Allerup, P., Kovac, V., Kvåle, G., Langfeldt, G. & Skov, P. (2009). *Evaluering av det Nasjonale kvalitetsvurderingssystemet for grunnopplæringen.* Kristiansand: Agderforskning.

Alseth, B., Breiteig, T. & Brekke, G. (2003). *Endringer og utvikling ved R97 som bakgrunn for videre planlegging og justering - matematikkfaget som kasus*. Notodden: Telemarksforskning.

Alseth, B. & Røsseland, M. (2014). Undersøkelseslandskap i matematikk. I M. E. Frislid & H. Traavik (red.) *Lese, skrive, regne - Pedagogikk og fagdidaktikk i begynneropplæringen* (s. 109-131). Oslo: Universitetsforlaget.

Applied Educational Systems (2019). *What are the 4 C's of 21st Century Learning*. Hentet 20.10.19 fra https://www.aeseducation.com/career-readiness/what-are-the-4-cs-of-21st-century-skills

Balanskat, A. & Engelhardt, K. (2015). *Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe*. Brussel: European Schoolnet.

Berger, P. & Luckmann, T. (1967). *The social construction of reality: A Treatise in the Sosciology of Knowledge.* New York: Doubleday.

Biggs, N. L. (1990). *Discrete Mathematics*. Oxford: Oxford University Press.

Boaler, J. (2015). *The elephant in the classroom: Helping children learn and love maths*. London: Souvenir Press.

Bocconi, S., Chioccariello, A. & Earp, J. (2018). *The Nordic approach to introducing Computational Thinking and programming in compulsory education.* Rapport skrevet for Nordic@ BETT2018 Steering Group. doi: 10.17471/54007

Borge, I. C., Sanne, A., Nortvedt, G. A., Meistad, J. A., Skrindo, K., Ranestad, K. & Kristensen, T. E. (2014). *Matematikk i norsk skole anno 2014. Faggjennomgang av matematikkfagene – Rapport fra ekstern arbeidsgruppe oppnevnt av Utdanningsdirektoratet*. Hentet 20.juni, 2019, Utdanningsdirektoratet, https://www.udir.no/tall-og-forskning/finn-forskning/rapporter/Matematikk-i-norsk-skole-anno-2014/

Bostrøm, E., Bø, O., Langmyhr, D. & Rydland, T. (2008). *Informasjonsteknologifaget og skoleverket: en bakgrunn og handlingsplan.* (FoU-rapport nr. 7) Halden: Høgskolen i Østfold. Hentet fra https://hiof.brage.unit.no/hiof-xmlui/bitstream/handle/11250/148600/hefte7-08.pdf?sequence=1&isAllowed=y

Botten, G. (2016). *Matematikk med mening*. Bergen: Caspar forlag.

Botten-Verboven, C., Maugesten, M., Nilsen, G., Aigeltinger, R., Ødegaard, P., Bendiksen, V., ... & Tofteberg, G. N. (2010). *Matematikk for alle, men alle behøver ikke å kunne alt.* Hentet 24.mai, 2019, Utdanningsdirektoratet, https://www.udir.no/tall-og-forskning/finn-forskning/rapporter/Matematikk-for-alle/

Braun, V. & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, *3*(2), 77-101.

Braun, V., Clarke, V., Hayfield, N. & Terry, G. (2019). Thematic analysis. I P. Liamputtong (Red.), *Handbook of Research Methods in Health Social Sciences*, (s. 843-860). New York: Springer

Breiteig, T. & Venheim, R. (1998). *Matematikk for lærere 1*. Oslo: Universitetsforlaget.

Brekke, G. & Gjone, G. (2001). Matematikk. I S. Sjøberg (Red.), *Fagdebatikk. Fagdidaktisk innføring i sentrale skolefag*, (s. 215-265). Oslo: Gyldendal akademisk.

Bryman, A. (2016). *Social research methods*. Oxford: Oxford University Press.

Caelli, K., Ray, L. & Mill, J. (2003). Clear as mud: Toward greater clarity in generic qualitative research. *International Journal of Qualitative Methods, 2*(2), 1-23. doi: 10.1177%2F160940690300200201

Dahl, G., Ranestad, K. & Hole, A. (2017, 25. oktober). Programmering rammer dybdelæring i matematikk, *Aftenposten*. Hentet fra https://www.aftenposten.no/meninger/kronikk/i/E0zga/Programmering-rammer-dybdelaring-i-matematikk--Geir-Dahl_-Kristian-Ranestad-og-Arne-Hole¨

Dahlum, S. (2018). Algoritme. Store Norske Leksikon. Hentet 20.10.2019 fra https://snl.no/algoritme

Dale, E. L., Engelsen, B. U. & Karseth, B. (2011). *Kunnskapsløftets intensjoner, forutsetninger og operasjonaliseringer: en analyse av en læreplanreform.* Oslo: Universitetet i Oslo.

Drageset, S. & Ellingsen, S. (2010). Å skape data fra kvalitativt forskningsintervju. *Sykepleien forskning, 5*(4), 332-335.

Drijvers, P. (2015). Digital technology in mathematics education: Why it works (or doesn't). I S. Cho (Red.), *Selected Regular Lectures from the 12th International Congress on Mathematical Education*, (s. 135-151). Springer: Cham.

Drijvers, P., Boon, P. & Van Reeuwijk, M. (2011). Algebra and technology. I P. Drijvers (Red.), *Secondary Algebra Education* (s. 179-202). New York: Springer. doi: 10.1007/978-94-6091-334-1_8

Egeberg, G., Hultin, H. & Berge, O. (2016). *Monitor skole 2016 – Skolens digitale tilstand.* Senter for IKT I utdanningen. Hentet 03.06.19 fra https://www.udir.no/globalassets/filer/tall-og-forskning/rapporter/2016/monitor_2016_bm_-_2._utgave.pdf

Ekelund, N., Olander, C., Wirstedt, A., Roosquist, Å., Johansson, M., Aasa, M. & Jakobsen, M. (2018, 09. september). Viktigt med kritisk blick på programmering i skolan. *Lärarnas tidning*. Hentet fra https://lararnastidning.se/viktigt-med-kritisk-blick-pa-programmering-i-skolan/

Erfjord, I. & Haara, F. (2018). Digitale ressurser i matematikkundervisningen. I A. Norstein & F. Haara (Red.), *Matematikkundervisning i en digital verden* (s. 11-27). Oslo: Cappelen Damm Akademisk.

Ertesvåg, F. (2014, 12. mai). Ny PISA-rapport: Mange norske elever gir opp for lett. *Verdens gang*. Hentet fra https://www.vg.no/nyheter/innenriks/i/Ovq51/ny-pisa-rapport-mange-norske-elever-gir-opp-for-lett

Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 8-32. doi: https://doi.org/10.26803/ijlter.17.12.2

Gilje, Ø., Landfald, Ø. & Ludvigsen, S. (2018). Dybdelæring – historisk bakgrunn og teoretiske tilnærminger. *Bedre Skole -tidsskrift for lærere og skoleledere, 30*(4), 22-27.

Goldberg, M. F. (1991). Portrait of Seymour Papert. *Educational Leadership*, *48*(7), 68-70.

Haraldsrud, A. D. (2018). *Programmering og modellering: naturvitenskapelig programmering for videregående skole*. Sandvika: Realfagsforlaget.

Hiebert, J. & Lefevre, P. (1986). Conceptual and procedural knowledge in mathematics: An introductory analysis. I J. Hiebert (Red.) *Conceptual and procedural knowledge: The case of mathematics,* (s. 1-28). Hillsdale, NJ.: Erlbaum.

Hofman, A. (2018, 18. oktober). Høringsuttalelse til lærelanutkast for universitetet i Sørøst-Norge. [Høringssvar]. Hentet fra https://hoering.udir.no/Uttalelse/v2/17e24bde-9547-4526-98bd-6bcdb8ac3656?disableTutorialOverlay=True

Holvik, M. (2014, 26. desember). Denne maskinen knekte de umulige kodene, *Teknisk Ukeblad*. Hentet fra https://www.tu.no/artikler/denne-maskinen-knekte-de-umulige-kodene/225357

Johannessen, L. E. (2018). *Hvordan bruke teori?: nyttige verktøy i kvalitativ analyse.* Oslo: Universitetsforlaget.

Jåtten, E. (2006). *First Lego League og motivasjon for realfag*. Høgskolen i Stord. Masteroppgave.

Kaufmann, O. T., Stenseth, B. & Holone, H. (2018). Programmering i matematikkundervisningen. I A. Norstein & F. Haara (Red.), *Matematikkundervisning i en digital verden*, (s. 73-96) Oslo: Cappelen Damm Akademisk.

Kelentric, M., Helland, K. & Astorp, A. (2017). *Rammeverk for lærerens profesjonsfaglige kompetanse.* Senter for IKT i utdanningen. Hentet 05. juni fra: https://www.udir.no/contentassets/081d3aef2e4747b096387aba163691e4/pfdk-rammeverk-2018.pdf

Kilpatrick, J., Swafford, J. & Findell, B. (Red.)(2001). *Adding it up: Helping children learn mathematics*. National Research Council. DC: National Academy Press.

Kirke-, utdannings- og forskningskomiteen (2016). *Fag – Fordypning – Forståelse. En fornyelse av Kunnskapsløftet.* (Innstilling 19S. 2016-2017). Hentet fra https://www.stortinget.no/no/Saker-og-publikasjoner/Publikasjoner/Innstillinger/Stortinget/2016-2017/inns-201617-019s/?all=true

Kjerneelementgruppa for matematikk (2017a). *Matematikk*. Hentet fra https://udirbloggen.no/matematikk/

Kjerneelementgruppa for matematikk (2017b). *Kjerneelementer i matematikk, men hvorfor programmering*? Hentet fra http://udirbloggen.no/kjerneelementer-i-matematikk-men-hvorfor-programmering/¨

Kjerneelementgruppa for matematikk (2017c). *Oppsummering av matematikk – andre innspillsrunde.* Hentet fra https://udirbloggen.no/13194-2/

Kunnskapsdepartementet (2006). *Læreplanverket for kunnskapsløftet*. Oslo: Utdanningsdirektoratet.

Kunnskapsdepartementet (2016). *Fag – Fordypning – Forståelse. En fornyelse av Kunnskapsløftet*. Stortingsmelding nr. 28 (2015-2016). Hentet fra https://www.regjeringen.no/no/dokumenter/meld.-st.-28-20152016/id2483955/

Kunnskapsdepartementet (2017). *Framtid, fag og digitalisering. Digitaliseringsstrategi for grunnopplæringen 2017-2021*. Oslo: Utdanningsdirektoratet

Kunnskapsdepartementet (2018a, 26. juni). *Fornyer innholdet i skolen*. Pressemelding Nr: 132-18. Hentet fra https://www.regjeringen.no/no/aktuelt/fornyer-innholdet-i-skolen/id2606028/

Kunnskapsdepartementet (2018b, 26. juni). *Kjerneelementer i fag*. Hentet fra https://www.regjeringen.no/contentassets/3d659278ae55449f9d8373fff5de4f65/kjerneelementer-i-fag-for-utforming-av-lareplaner-for-fag-i-lk20-og-lk20s-fastsatt-av-kd.pdf

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

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

Landfald, Ø. & Gilje, Ø. (2018, 13. september). Dybdelæring. Universitetet i Oslo. Hentet fra https://www.uv.uio.no/forskning/satsinger/fiks/kunnskapsbase/dybdelering/

Lindbäck, S. & Glavin, P. (2015, 29. januar). Sosiale og emosjonelle kompetanser i fremtidens skole, *Bedre Skole*. Hentet fra https://utdanningsforskning.no/artikler/sosiale-og-emosjonelle-kompetanser-i-fremtidens-skole/

Lithner, J. (2008). A research framework for creative and imitative reasoning. *Educational Studies in Mathematics*, *67*(3), 255-276.

Loksa, D. & Ko, A. J. (2016). The role of self-regulation in programming problem solving process and success. *Proceedings of the 2016 ACM Conference on International Computing Education Research,* 83-91. ACM.

Ludvigsen, S. (2016, 20. oktober). Dybdelæring er en forutsetning for fremtidens skole. *Aftenposten*. Hentet fra https://www.aftenposten.no/meninger/debatt/i/oa33j/Dybdelaring-er-forutsetning-for-fremtidens-skole--Sten-Ludvigsen

Lynnenbakken, H. (2018, 11. oktober). Programmering er ikke det samme som koding. Titan.uio.no. Hentet fra https://www.digi.no/artikler/programmering-er-ikke-det-samme-som-koding-den-storste-jobben-er-a-finne-ut-hva-du-onsker-a-gjore-og-formulere-det-pa-en-presis-mate/448366

Malterud, K. (2011). *Kvalitative metoder i medisinsk forskning* (3. utgave). Oslo: Universitetsforlaget.

Marton, F. & Säljö, R. (1976). On qualitative differences in learning: Outcome and process. *British journal of educational psychology*, *46*(1), 4-11.

Nostrati, M. & Wæge, K. (2015). *Sentrale kjennetegn på god læring og undervisning i matematikk.* Matematikksenteret. Hentet fra https://www.matematikksenteret.no/nettbutikk/sentrale-kjennetegn-p%C3%A5-god-l%C3%A6ring-og-undervisning-i-matematikk

Nostrati, M. & Wæge, K. (2018). *Dybdelæring i matematikk*. Matematikksenteret. Hentet fra: http://realfagsloyper.no/sites/default/files/2018-04/MN%20KW%20dybdel%C3%A6ring%2015.04.18_0.pdf

NOU 2013: 2. (2013). *Hindre for digital verdiskaping*. Oslo: Norges offentlige publikasjoner. Hentet fra: https://www.regjeringen.no/no/dokumenter/nou-2013-2/id711002/

NOU 2014: 7. (2014). *Elevenes læring i fremtidens skole. Et kunnskapsgrunnlag*. Oslo: Norges offentlige publikasjoner. Hentet fra: https://www.regjeringen.no/no/dokumenter/NOU-2014-7/id766593/

NOU 2015: 8. (2015). *Fremtidens skole. Fornyelse av fag og kompetanser*. Oslo: Norges offentlige publikasjoner. Hentet fra: https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/

NOU 2015: 13. (2015). *Digital sårbarhet – Sikkert samfunn*. Oslo: Norges offentlige publikasjoner. Hentet fra: https://www.regjeringen.no/no/dokumenter/nou-2015-13/id2464370/

NOU 2019: 3. (2019). *Nye sjanser – bedre læring – Kjønnsforskjeller i skoleprestasjoner og utdanningsløp.* Oslo: Norges offentlige publikasjoner. Hentet fra https://www.regjeringen.no/no/dokumenter/nou-2019-3/id2627718/

NTCM (2008). Strategic use of Technology in Teaching and Learning Mathematics. Hentet fra https://www.nctm.org/Standards-and-Positions/Position-Statements/Strategic-Use-of-Technology-in-Teaching-and-Learning-Mathematics/

Nygaard, O., Hundeland, P. S. & Pettersen, P. (1999). *Aha*. *Matematikk og matematikkdidaktikk*. Kristiansand: Høyskoleforlaget

OECD (2015). *OECD Skills Studies: Skills for Social Progress: The Power of Social and Emotional Skills*. OECD Publishing. doi: 10.1787/9789264226159-en

Ottestad, G., Throndsen, I., Hatlevik, O. & Rothagi, A. (2014). *Digitale ferdigheter for alle?.* Senter for IKT i utdanningen. Hentet fra https://www.udir.no/globalassets/filer/tall-og-forskning/rapporter/2014/icils_rapport_rettet.pdf

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

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

Percy, W. H., Kostere, K. & Kostere, S. (2015). Generic qualitative research in psychology. *The Qualitative Report*, *20*(2), 6-85.

Pintrich, P. R. (2002). The Role of Metacognitive Knowledge in Learning, Teaching, and Assessing. *Theory Into Practice*, *41*(4), 219-225. doi: 10.1207/s15430421tip4104_3

Pólya, G. (1957). *How to* solve *it. A new aspect of mathematical method*. Princeton: Princeton University Press.

Robson, C. (2011). *Real world research* (Vol. 3). Chichester: Wiley.

Rossen, E. & Dvergsdal, H. (2019). Datamaskin - historikk. I Store norske leksikon. Hentet 29.04.19 fra https://snl.no/datamaskin_-_historikk

Rossing, N., Asphjell, A. & Aas, E. (2001). *Fra kuleramme til PC*. Vitensenteret. Hentet fra https://www.vitensenteret.com/sites/default/files/Datamaskinens%20historie.pdf

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

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E. & Voll, L. O. (2016). *Teknologi og programmering for alle - En faggjennomgang med forslag til endringer i grunnopplæringen.* Oslo: Utdanningsdirektoratet. Hentet fra https://www.udir.no/tall-og-forskning/finn-forskning/rapporter/teknologi-og-programmering-for-alle

Scandrett, H. (2008). Using Geoboards in Primary Mathematics: Going... Going... Gone?. *Australian Primary Mathematics Classroom*, *13*(2), 29-32.

Scherer, R., Siddiq, F. & Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of educational psychology*, 111(5), 764-792. doi: 10.1037/edu0000314

Schoenfeld, A. H. (2013). Reflections on problem solving theory and practice. *The Mathematics Enthusiast*, *10*(1), 9-34.

Selvik, Kristine m.fl. (2016). *Programmering i skolen*. Senter for IKT i utdanningen. Hentet fra https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Skemp, R. (1976). Relational understanding and instrumental understanding. *Mathematics Teaching, 77*(1), 20-26.

Skolverket (2016). *Kursplan för matematik.* Stockholm: Skolverket. *Hentet* fra: https://www.skolverket.se/undervisning/grundskolan/laroplan-och-kursplaner-for-grundskolan/laroplan-lgr11-for-grundskolan-samt-for-forskoleklassen-och-fritidshemmet?url=1530314731%2Fcompulsorycw%2Fjsp%2Fsubject.htm%3FsubjectCode%3DGRGRMAT01%26tos%3Dgr%26p%3Dp&sv.url=12.5dfee44715d35a5cdfa219f

Sikko, S. (2017, 8.november). Høringssvar 2. innspillsrunde av kjernelementer. [Høringssvar]. Hentet fra https://hoering.udir.no/Uttalelse/v2/fee87ed0-b7f3-49fc-a636-199a15beb19e?disableTutorialOverlay=True

Solvang, R. (1992). *Matematikkdidaktikk*. Oslo: NKI Forlaget.

Stedøy, I. & Torkildsen, S. (2018). *Hvorfor problemløsning?.* Matematikksenteret. Hentet fra https://www.matematikksenteret.no/sites/default/files/attachments/resources/Hvorfor%20probleml%C3%B8sing.pdf

Sylte, A. (2013). *Profesjonspedagogikk*, Oslo: Gyldendal akademisk.

Tellefsen, C. (2018). Realfaglig programmering. Hentet fra https://www.uv.uio.no/forskning/satsinger/fiks/kunnskapsbase/realfaglig-programmering/index.html

Tesch, N. (2019). Al-Khwarizmi. Encyclopedia Britannica. Hentet 20.10.2019 fra https://www.britannica.com/biography/al-Khwarizmi

Thanh, N. C. & Thanh, T. T. (2015). The interconnection between interpretivist paradigm and qualitative methods in education. *American Journal of Educational Science*, *1*(2), 24-27.

Tjora, A. (2017). *Kvalitative forskningsmetoder i praksis*. (3. utg). Oslo: Gyldendal akademiske.

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

Utdanningsdirektoratet (2017, 6. september). Første skisse til kjerneelementer i matematikk fellesfag. [Høringsutkast]. Hentet fra https://hoering.udir.no/Hoering/v2/151?notatId=222

Utdanningsdirektoratet (2018a, 26. november). Hva er fagfornyelsen. Hentet fra https://www.udir.no/laring-og-trivsel/lareplanverket/fagfornyelsen/nye-lareplaner-i-skolen/

Utdanningsdirektoratet (2018b, 4. juni). Matematikk –oppsummering av innspill. Hentet fra https://www.udir.no/laring-og-trivsel/lareplanverket/fagfornyelsen/kjerneelementer/matematikk--oppsummering-av-innspill/

Valenta, A. (2015). *Aspekter ved tallforståelse.* Matematikksenteret. Hentet fra https://www.matematikksenteret.no/sites/default/files/attachments/page/Valenta_Tallforsta%CC%8Aelse.pdf

Valenta, A. (2016). *Tallforståelse – anvendelse og engasjement*. Matematikksenteret. Hentet fra https://www.matematikksenteret.no/sites/default/files/attachments/page/Valenta_Tallforsta%CC%8Aelse.pdf

Van De Walle, J., Karp, K. & Bay-Williams, J. (2014). *Elementary & middle school mathematics. Teaching developmentally.* Essex: Pearson Education Limited.

Voll, L.O. & Vinje, B. (2018). Teknologi i realfagene. Hentet fra http://realfagsloyper.no/sites/default/files/2018-11/Teknologi%20i%20realfagene.pdf

Wæge, K. & Nosrati, M. (2018). *Motivasjon i matematikk*. Oslo: Universitetsforlaget.

Vedlegg

A: Intervjuguide

Denne intervjuguiden sees kun av intervjuer under intervjuet. Dette er et notat for å hjelper intervjueren med å huske spørsmål og struktur under selve intervjuet.

Før intervjuet
Presentere seg selv og målet med studien
- Masterstudie om "Programmering som del av læreplanen i matematikk" og målet er todelt:

1. Forsøke å belyse faglige argumenter for eller imot innføringen av programmering som del av matematikkfaget
2. Finne forutsetninger for å lykkes med implementering av programmering i matematikk

Informasjon om opptaket:
- Ønsker å ta opp intervjuet digitalt slik at vi kan bruke det senere
- Opptaket er kun tilgjengelig for meg, veileder og sensor
- Ønske om å ikke anonymisere, dersom jeg ikke får samtykket, blir det anonymisert.
- Opptaket slettes lokalt i oktober, men blir brent på CD for at sensor kan ha tilgang til dette under sensuren.

Intervjutype:
- Semi-strukturert intervju: Kommer til å stille spørsmål rundt valgte temaer og det kommer evt. oppfølgingsspørsmål basert på deltakernes svar.
Lurer du på noe?
HUSK: lytte, tillate stillhet, gi feedback, smile

Hjelpespørsmål: Kan du utdype det? Ville du sagt at…. Mener du da at…. Hva tenker du om det…

Spørsmål til deltakere:
Spørsmål 1:
1980-tallets forsøk og prosjekter innen programmering ble ingen suksess, men nå er fokuset på programmering større en noen gang. Hva mener du er grunnen til dette?

Spørsmål 2:
Hva mener du ligger i begrepet «algoritmisk tenkning»? Tenker du at dette begrepet kan tolkes på flere måter?

Spørsmål 3:
Gjennom algoritmisk tenkning blir programmering en del av kjerneelementene i matematikk. Programmering skal innføres i matematikkfaget, istedenfor som eget fag som var anbefalingen fra Sanneutvalgets rapport «teknologi og programmering for alle». Hva tenker du om dette?

Spørsmål 4:
Hvorfor mener du programmering ble så omdiskutert i høringsrundene av kjerneelementene?

Spørsmål 5:
Mener du matematikkompetansen til elevene kan dra nytte av å få programmering inn i faget? Eventuelt hvorfor?

Spørsmål 5:
Det har vært stilt spørsmål omkring det at programmering vil gå på bekostning av forståelse og dybdelæring i matematikk, blant annet pga. stofftrengsel. Hva tenker du om dette?

Spørsmål 6:
Hvilke nye muligheter og utfordringer tenker du matematikklærerne får som følge av innføringen av programmering i faget?

Spørsmål 7:
Hva tenker du må til for at lærerne skal se nytteverdien i å bruke til på programmering i matematikk?

Spørsmål 8.
Hvilke forutsetninger mener du må ligge til grunn for at implementeringen av programmering i matematikkfaget skal lykkes?

Annet:

B: Forespørsel om deltakelse i forskningsprosjekt

Bakgrunn og formål

Gjennom masterstudiet i matematikk ved Universitetet i Agder, skal jeg skrive en masteroppgave med arbeidstittel ”Programmering i matematikk – muligheter og utfordringer”. I oppgaven ønsker jeg å studere prosessen som har vært i forkant av innføringen av programmering som del av matematikkfaget og undersøke argumenter eller begrunnelser som taler for eller imot denne innføringen. Jeg ønsker også å finne hvilke forutsetninger som bør ligge til grunn for at skolene skal lykkes med implementeringen av programmering i matematikkfaget.

Jeg har derfor valgt å se på følgende forskningsspørsmål:

1. Hvilke faglige argumenter finner vi for eller imot innføringen av programmering i matematikkfaget?

2. Hvilke forutsetninger bør ligge til grunn for å lykkes med implementeringen av programmering i skolen?

Oppgaven skrives for Universitetet i Agder, Fakultet for teknologi og realfag.

Hva innebærer deltakelse i studien?

Siden jeg bor i Alta vil personlig møte nok blir vanskelig å få til, derfor er planen at intervjuene gjennomføres over Skype. Jeg vil gjerne bruke digitale verktøy for å ta opp lyd, dersom det er greit for deg som blir intervjuet. Intervjuet kommet til å være semi-strukturert med fokus på forskningsspørsmålene som er nevnt over. Spørsmålene jeg ønsker å stille er lagt ved som vedlegg i dette dokumentet.

Hva skjer med informasjonen om deg?

Alle personopplysninger vil bli behandlet konfidensielt. Det vil kun være meg, Renate Elise Jakobsen, og veilederen min, Niclas Larson, som vil ha tilgang til opptaket som blir samlet inn. I etterkant vil en sensor også ha tilgang til opptaket fra intervjuene. Dataene som blir brukt direkte i masteroppgaven vil være tilgjengelig for allmennheten etter masterprosjektet er ferdig og sensurert. Dataene som blir samlet inn kommer til å ligge lokalt på min personlige PC, som er passordbeskyttet og i tillegg brent over på CD for at sensoren skal ha tilgang. Etter

prosjektets slutt kommer opptakene og dataen som er samlet inn til å bli slettet fra min personlige PC, samt at CDen vil destrueres. Dersom deltakerne godkjenner det, vil jeg benytte deres fulle navn i masterprosjektet. Prosjektet skal etter planen avsluttes 31.12.2019.

Frivillig deltakelse

Det er frivillig å delta i studien, og du kan når som helst trekke ditt samtykke uten å oppgi noen grunn. Dersom du trekker deg, vil alle opplysninger om deg bli slettet. Dersom du har spørsmål til studien, kan du ta kontakt med *Renate Elise Jakobsen* på telefon (47 64 27 41) eller epost (Renate.elise.jakobsen@ffk.no).
Kontaktinformasjon til veileder *Niclas Larson*, epost: niclas.larson@uia.no

Studien er meldt til Personvernombudet for forskning, NSD – Norsk senter for forskningsdata AS.

Samtykke til deltakelse i studien

Hvis du er villig å delta i studien kan du gi ditt samtykke skriftlig via epost eller ved å skrive ut disse to sider, skanne dem og sende tilbake til Renate Elise Jakobsen.

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

Sted, dato, signatur

C: Meldeskjema til NSD

NSD NORSK SENTER FOR FORSKNINGSDATA

Meldeskjema 228371

Sist oppdatert

07.02.2019

Hvilke personopplysninger skal du behandle?

- Navn (også ved signatur/samtykke)
- Lydopptak av personer
- Bakgrunnsopplysninger som vil kunne identifisere en person
- Andre opplysninger som vil kunne identifisere en fysisk person

Type opplysninger

Du har svart ja til at du skal behandle bakgrunnsopplysninger, beskriv hvilke

Arbeidssted og stilling. Er relevant i forhold til valg av person som er valgt til intervju.

Du har svart ja til at du behandler andre opplysninger som vil kunne identifisere en person, beskriv hvilke

Navn, stilling og arbeidssted

Skal du behandle særlige kategorier personopplysninger eller personopplysninger om straffedommer eller lovovertredelser?

Nei

Prosjektinformasjon

Prosjekttittel

Programmering som en del av matematikkfaget - En studie av prosessen rundt implementeringen av programmering som del av de grunnleggende digitale ferdighetene i matematikk

Begrunn behovet for å behandle personopplysningene

Ønsker gjennom å intervjue personer som har deltatt i kjerneelementgruppa som har laget utkastene til læreplan å få innsikt i hvorfor programmering er innført i matematikkfaget. Ønsker også å interjvue andre relevante personer som gjennom media offentlig har uttalt seg i forhold til dette temaet. Til dette trenger jeg kun navn, stilling og eventuelt arbeidssted.

Ekstern finansiering

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Renate Elise Jakobsen, renate.elise.jakobsen@ffk.no, tlf: 47642741

Behandlingsansvar

Behandlingsansvarlig institusjon

Universitetet i Agder / Fakultet for teknologi og realfag / Institutt for matematiske fag

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Niclas Larson, niclas.larson@uia.no, tlf: 38142404

Skal behandlingsansvaret deles med andre institusjoner (felles behandlingsansvarlige)?

Nei

Utvalg 1

Beskriv utvalget

Personer som sitter i kjerneelementgruppa for matematikk, eventuelt andre skoleinteressenter eventuelt politikere

Rekruttering eller trekking av utvalget

Fagpersoner eller politikere som har vært sentrale i utarbeidelse av relevante dokumenter, eller personer som har vært å uttalt seg i media vil være relevante for undersøkelsen

Alder

18 - 100

Inngår det voksne (18 år +) i utvalget som ikke kan samtykke selv?

Nei

Personopplysninger for utvalg 1

- Navn (også ved signatur/samtykke)
- Lydopptak av personer
- Bakgrunnsopplysninger som vil kunne identifisere en person

Hvordan samler du inn data fra utvalg 1?

Personlig intervju

Grunnlag for å behandle alminnelige kategorier av personopplysninger

Samtykke (art. 6 nr. 1 bokstav a)

Informasjon for utvalg 1

Informerer du utvalget om behandlingen av opplysningene?

Ja

Hvordan?

Skriftlig informasjon (papir eller elektronisk)

Tredjepersoner

Skal du behandle personopplysninger om tredjepersoner?

Nei

Dokumentasjon

Hvordan dokumenteres samtykkene?

- Elektronisk (e-post, e-skjema, digital signatur)
- Manuelt (papir)

Hvordan kan samtykket trekkes tilbake?

Muntlig eller skriftlig via mail.

Hvordan kan de registrerte få innsyn, rettet eller slettet opplysninger om seg selv?

De vil få kopi av transkripsjoner og se utkast til masteroppgaven der de er sitert, før denne publiseres, for å godkjenne den. De vil når som helst kunne trekke seg fra dette og alt av dokumentasjon og lydfiler vil slettes.

Totalt antall registrerte i prosjektet

1-99

Tillatelser

Skal du innhente følgende godkjenninger eller tillatelser for prosjektet?

Behandling

Hvor behandles opplysningene?

- Mobile enheter tilhørende behandlingsansvarlig institusjon
- Maskinvare tilhørende behandlingsansvarlig institusjon

Hvem behandler/har tilgang til opplysningene?

- Prosjektansvarlig
- Student (studentprosjekt)
- Interne medarbeidere

Tilgjengeliggjøres opplysningene utenfor EU/EØS til en tredjestat eller internasjonal organisasjon?

Nei

Sikkerhet

Oppbevares personopplysningene atskilt fra øvrige data (kodenøkkel)?

Ja

Hvilke tekniske og fysiske tiltak sikrer personopplysningene?

- Opplysningene anonymiseres
- Adgangsbegrensning
- Andre sikkerhetstiltak

Hvilke

Brukernavn og og passord på PC

Varighet

Prosjektperiode

01.01.2019 - 31.12.2019

Skal data med personopplysninger oppbevares utover prosjektperioden?

Nei, alle data slettes innen prosjektslutt

Vil de registrerte kunne identifiseres (direkte eller indirekte) i oppgave/avhandling/øvrige publikasjoner fra prosjektet?

Ja

Begrunn

Fordi de er sentrale personer som er kompetanse innen området jeg forsker på. Derfor vil navn og arbeidsted/stilling være opplyst om i oppgaven siden deres meninger og tanker om området er sentralt for oppgaven. Deltakere kan når som helst før publisering trekke seg fra deltakelse.

Tilleggsopplysninger

D: Kvittering fra NSD

NSD NORSK SENTER FOR FORSKNINGSDATA

NSD sin vurdering

Prosjekttittel

Programmering som en del av matematikkfaget - En studie av prosessen rundt implementeringen av programmering som del av de grunnleggende digitale ferdighetene i matematikk

Referansenummer

228371

Registrert

07.02.2019 av Renate Elise Jakobsen - renatj16@student.uia.no

Behandlingsansvarlig institusjon

Universitetet i Agder / Fakultet for teknologi og realfag / Institutt for matematiske fag

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Niclas Larson, niclas.larson@uia.no, tlf: 38142404

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Renate Elise Jakobsen, renate.elise.jakobsen@ffk.no, tlf: 47642741

Prosjektperiode

01.01.2019 - 31.12.2019

Status

10.02.2019 - Vurdert med vilkår

Vurdering (1)

10.02.2019 - Vurdert med vilkår

FORENKLET VURDERING MED VILKÅR
Etter gjennomgang av opplysningene i meldeskjemaet med vedlegg, vurderer vi at prosjektet har lav personvernulempe fordi det ikke behandler særlige kategorier eller personopplysninger om straffedommer og lovovertredelser, eller inkluderer sårbare grupper. Prosjektet har rimelig varighet og er basert på samtykke. Vi gir derfor prosjektet en forenklet vurdering med vilkår.

Du har et selvstendig ansvar for å følge vilkårene og sette deg inn i veiledningen i denne vurderingen. Dersom du følger vilkårene og prosjektet gjennomføres i tråd med det som er dokumentert i meldeskjemaet vil behandlingen av personopplysninger være i samsvar med personvernlovgivningen.

VILKÅR
Vår vurdering forutsetter:
1. At du gjennomfører prosjektet i tråd med kravene til informert samtykke
2. At du ikke innhenter særlige kategorier eller personopplysninger om straffedommer og lovovertredelser
3. At du følger behandlingsansvarlig institusjon (institusjonen du studerer/forsker ved) sine retningslinjer for datasikkerhet
4. At du laster opp revidert(e) informasjonsskriv på utvalgssiden(e) i meldeskjemaet og trykker «bekreft innsending», slik at du og behandlingsansvarlig institusjon får korrekt dokumentasjon. NSD foretar ikke en ny vurdering av det reviderte informasjonsskrivet.

1. KRAV TIL INFORMERT SAMTYKKE
De registrerte skal få skriftlig og/eller muntlig informasjon om prosjektet og samtykke til deltakelse. Du må påse at informasjonen minst omfatter:
- Prosjektets formål og hva opplysningene skal brukes til
- Hvilken institusjon som er behandlingsansvarlig
- Hvilke opplysninger som innhentes og hvordan opplysningene innhentes
- At det er frivillig å delta og at man kan trekke seg så lenge studien pågår uten at man må oppgi grunn
- Når prosjektet skal avsluttes og hva som skal skje med personopplysningene da: sletting, anonymisering eller videre lagring
- At du/dere behandler opplysninger om den registrerte basert på deres samtykke
- Retten til å be om innsyn, retting, sletting, begrensning og dataportabilitet (kopi)
- Retten til å klage til Datatilsynet
- Kontaktopplysninger til prosjektleder (evt. student og veileder)
- Kontaktopplysninger til institusjonens personvernombud

På nettsidene våre finner du mer informasjon og en veiledende mal for informasjonsskriv:
http://www.nsd.uib.no/personvernombud/hjelp/informasjon_samtykke/informere_om.html

Det er ditt ansvar at informasjonen du gir i informasjonsskrivet samstemmer med dokumentasjonen i meldeskjemaet.

2. TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til 31.12.2019.

3. 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).

Dersom du benytter en databehandler i prosjektet må behandlingen oppfylle kravene til bruk av databehandler, jf. art 28 og 29.

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

NSD SIN VURDERING
NSDs vurdering av lovlig grunnlag, personvernprinsipper og de registrertes rettigheter følger under, men forutsetter at vilkårene nevnt over følges.

LOVLIG GRUNNLAG
Prosjektet vil innhente samtykke fra de registrerte til behandlingen av personopplysninger. Forutsatt at vilkår 1 og 4 følges er det NSD sin vurdering 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
Forutsatt at vilkår 1 til 4 følges vurderer NSD at den planlagte behandlingen av personopplysninger vil følge

prinsippene i personvernforordningen om:

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

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

Forutsatt at informasjonen oppfyller kravene i vilkår 1 vurderer NSD at informasjonen om behandlingen som de registrerte vil motta oppfyller lovens krav til form og innhold, jf. art. 12.1 og art. 13.

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

MELD ENDRINGER
Dersom den planlagte behandlingen av personopplysninger endrer seg, kan det være nødvendig å melde dette til NSD ved å oppdatere meldeskjemaet. På våre nettsider informerer vi om hvilke endringer som må meldes. Vent på svar før endringer gjennomføres.

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

Lykke til med prosjektet!

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