Programmeringsparadokset etter Sfard og Leron

Oline Sæverud Aksnes[1]
oline.s.aksnes@ntnu.no

Andreas Brandsæter[1,2]
andreas.brandsaeter@hivolda.no

Elise Indregård Petersen[1]
elise.petersen@ntnu.no

Hans Georg Schaathun[1⋆]
hasc@ntnu.no

[1] NTNU — Noregs Teknisk-Naturvitskaplege Universitet
Institutt for IKT og realfag
6025 Ålesund, Norge
[2] Høgskulen i Volda, Institutt for Realfag
6103 Volda, Norge

Samandrag Sfard og Leron (1996) observerte at studentar arbeider flittigare og oftare lukkast når dei programmerer enn dei gjer med analytisk matematikk, sjølv om programmeringsoppgåva løyser det same matematiske problemet i ein meir generell form. Her går me gjennom relevant litteratur og teori for å drøfta kvifor det kan vera slik. Dette reiser interessante spørsmål både om korleis me kan bruka programmering for å styrka matematikkforståinga og korleis ein best lærer programmering.

Nykelord: matematikkdidaktikk · informatikkdidaktikk · problemløysing · programmering · prosjekt

1 Innleiing

Sfard og Leron (1996) samanlikna to studentoppgåver i klasserommet. Den eine spurde om eit særtilfelle. Elevane fekk tre punkt i planet, og skulle finna sentrum og radius for sirkelen gjennom alle tre. Den andre var generell. Dei skulle skriva eit dataprogram som løyser det same problemet for tre vilkårlege punkt. Ein skulle kanskje tru at det generelle problemet er vanskelegast. Med tanke på logisk kompleksitet må det vera det. Ei generell løysing vil alltid implisera den spesielle løysinga. Paradoksalt nok viste det seg motsett i praksis. Elevane lukkast oftare med den generelle programmeringsoppgåva og miste oftare motet på den spesielle rekneoppgåva.

Dette programmeringsparadokset har berre fått beskjeden merksemd i litteraturen, trass i at observasjonen reiser fleire viktige spørsmål både om undervising i programmering og programmering som hjelpemiddel i matematikkundervisinga. Det spørsmålet som interesserer oss forfattarar mest, er om me kan bruka det pågangsmotet og dei praktiske ferdigheitene som elevane viser i programmering, til å styrka kompetansen i matematikk.

⋆ Korresponderande forfattar, hasc@ntnu.no

Artikkelen åt Sfard og Leron (1996) har eit trettitals siteringar, med eit par interessante teoretiske tolkingar av paradokset, men dei er relativt fragmenterte og lite systematiske. I denne artikkelen skal me primært granska det arbeidet som er gjort, og drøfta den teoretiske forståinga av paradokset meir systematisk. Vidare skisserer med eit empirisk forskingsdesign som kan utdjupa eksisterande innsikt, samt eit læringsdesign som utnyttar programmering for matematikkforståing[3] i forkurset til ingeniørutdanninga. Det er likevel den teoretiske granskinga som er målet for artikkelen, go den empiriske studien er framtidig arbeid.

2 Teoretiske tolkingar

Me har gått gjennom dei 34 siteringane av Sfard og Leron (1996) som vert rapporterte av Google Scholar, med nokre få unntak som var vanskelege å skaffa. Me skal ikkje presentera dei mange arbeida som berre nemnde Sfard og Leron i forbifarten, og berre fokusera på dei som tilbyr teoretiske tolkingar. Fleire av tolkingane viser tilbake til eldre, liknande arbeid som me òg vil ta med.

2.1 Problem mot prosjekt

Papert (1996) forklarer paradokset som ein inversjon frå problem til prosjekt. Han peikar på at den naturlege vegen til dei fleste formar for kunnskap er utforsking mot gradvis djupare forståing. Dette skjer ofte gjennom leik med umoden forståing og berre delvis forma konsept (jf. Luntley, 2018). Me ser det lett på barnehagenivå. Me forklarer ikkje borna korleis verda ser ut. Me tek dei heller med ut og *ser verda saman med dei*. Matematikkomgrep kjem inn når me saman tel ting eller samanliknar formar (sirklar, trekantar, osv.) rundt oss.

Skulen, ifylgje Papert (1996), inverterer denne naturlege vegen til kunnskap. Fyrst *forklarer* læraren kva omgrepa tyder og korleis dei skal brukast. *Etterpå* ser me om studentane faktisk kan bruka dei på å forklara verda, og då er det gjerne «anten/eller». Der er ikkje alltid rom for å prøva seg fram, og langt mindre eksperimentera for å *testa* si eiga umodne forståing i praksis.

Prosjektet, slik Papert ser det, *reinverterer* situasjonen og oppmodar studenten til å prøva seg fram, og gjerne leika med uferdige løysingar og perifere idéar. Studenten kan kanskje starta med å tenkja ein vilkårleg sirkel, og dermed stadfesta at han veit kva ein sirkel er, før han vurderer om sirkelen tilfredsstiller krava. På dette området er Papert ein pionér. Han stod bak programmeringsspråket LOGO som skulle ta programmeringa inn i skulen på 1980-talet. Litt som med det nyare Scratch, skulle elevane programmera ein figur (skjelpadda) til å gå rundt på skjermen, og gjerne teikna figurar bak seg med ein penn som ein kunne skru av og på. Det var eit system man skulle kunne leika med og umiddelbart sjå resultat av alt ein gjer.

[3] Her kan me lesa *forståing* nokonlunde synonymt med *kompetanse*. Omgrepet er tvitydig (Schaathun, 2022b), og me har ikkje plass til å problematisera det her.

Andre forfattarar forklarer paradokset på liknande måte, t.d. Leron og Hazzan (1998) som med referanse til Paperts arbeid frå 1980, ser programmering som ein arena for konstruktivistisk, eller «piagetisk», læring i ei mikroverd.

Det er likevel klart at overgangen frå instrumentelt problem til utforskande prosjekt ikkje fylgjer direkte frå oppgåveformuleringane frå Sfard og Leron (1996). På den eine sida kan ein fyrst løysa programmeringsproblemet som ein algebraisk formel, som er triviell å programmera. På den andre sida kan studentar som arbeider med særtilfellet med blyant og papir, gjerne prøva seg fram med å teikna sirklar og rekna ut radiar, medan dei reflekterer over korleis ein sirkel er definert. I praksis er der derimot få studentar som gjer det. Som Schoenfeld (1988) peikte på, forventar studentane gjerne at matematikkoppgåver skal løysast raskt og effektivt med ein kjend teknikk (for det gjer jo læraren på tavla). Om dei ikkje umiddelbart ser løysinga, gjev dei opp. Programmering ser ut til å setja studenten i ein annan og smidigare sinnstilstand.

Det utforskande prosjektet har mykje til felles med smidige metodar. Hazzan og Goldenberg (1997)[4] set paradokset i samanheng med to modellar for å handtera kompleksitet. Den vanlege modellen er ein lineær oppdeling i «atom». Den andre modellen, som kjem til syne i smidige metodar, kallar dei for *successive refinement*, altso ein gradvis foredling av forståing, der studenten heile vegen ser heile biletet og kan sjå delane i si rette kontekst.

2.2 Regel mot ekspertise

Dreyfusbrørne skreiv ikkje spesielt om matematikkdidaktikk og langt mindre paradokset åt Sfard og Leron, men modellen deira for ferdigheitslæring (t.d. H. Dreyfus & Dreyfus, 1988) er likevel ei interessant samanlikning. Dei set opp ein femstegsmodell basert på fleire empiriske studiar. Stikk i strid med Papert, finn dei at nybyrjaren som regel startar med generelle reglar som han fylgjer slavisk. Naturlig og intuitiv utforsking kjem seinare. Dette har nok samanheng med at Papert skriv om korleis barn lærer, medan dreyfusbrørne studerer vaksne. Særleg flygar- og køyreopplæring går igjen som døme, og det ville vore openbert farleg om ferske pilotar og bilførarar heilt utan instruksjon skulle utforska korleis fly og bilar oppfører seg i fart.

I dreyfusmodellen startar nybyrjaren med å fylgja generelle reglar. Den avanserte nybyrjaren (nivå 2) tek gradvis i bruk fleire reglar på vilkår, dvs. situasjonsavhengige til skilnad frå generelle reglar. Framleis ligg fokus på *korleis* ein gjer ting, og nybyrjaren på desse to nivåa ser helst problemet utanfrå. På dei tre siste nivåa, som kompetent, dugleg og til slutt ekspert, vert utøvaren meir og meir personleg og kjenslemessig involvert i mål og meining. Merksemda skiftar frå korleis til *kvifor*. Det fyrste som vert internalisert er utfallet eller målet. Den kompetente utøvaren dannar seg eit betre bilete av kva han ynskjer å oppnå, medan forståing og avgjerdsler stadig er basert på analytisk distanse. Den duglege utøvaren får ei meir involvert forståing av korleis situasjonen heng saman, men tek stadig analytiske avgjerdsler.

[4] Hazzan og Goldenberg siterer eit arbeid av Leron frå 1992, men me har ikkje klart å skaffa dette originalarbeidet.

Det mest kontroversielle i dreyfusmodellen er kanskje tesen om at eksperten (nivå 5) ikkje handlar ut frå reglar som ein kan fylgja logisk, ikkje ein gong undermedvitne reglar. Eksperten handlar heilt og fullt intuitivt ut frå ein kroppsleg og kjenslemessig *veren*[5] i verda. Dette er den same tesen som Hubert Dreyfus brukte i kritikken av kunstig intelligens på 1960- og 70-talet (H. L. Dreyfus, 1992). Den gongen bygde KI-forskinga på Platon sin tese om at logiske reglar ligg til grunn for all intelligent tenking, og utfordringa var å finna dei. Med utgangspunkt i dei sokalla antifilosofane, hovudsakleg Heidegger og Merleau-Ponty, argumenterte Dreyfus for at dette ikkje er mogleg. Moderne KI byggjer heller ikkje på denne platonske tilnærming, men på statistikk og sannsynsmodellar.

Nok om det. Spørsmålet vårt er kva dreyfusmodellen fortel oss om matematikkundervisinga, som i all hovudsak er basert rundt teorem og rekneregler og sjelden oppmodar til intuisjon og kjenslemessig engasjement. Dersom dreyfusbrørne har rett, arbeider ikkje matematikkekspertane etter dei strenge logiske reglane som matematikkundervisninga formidlar, men intuitivt ut frå den situasjonen som dei er oppslukte i. Det er omtrent det same som Poincaré (1910) skreiv. Han meinte at han ikkje var so flink til å hugsa reglar og gjerne gjorde feil. Når han dugde som matematikar var det fordi han lét seg leia av den generelle gangen i resonnementet.

Programmeringsprosjektet oppmodar studentane til å prøva og feila, dvs. til sjølv å forma forklaringsmodellar og hypotesar som dei kan testa ved å variera programmet. Dette stimulerer til utvikling på nivå 3 og 4 i dreyfusmodellen. Når matematikkundervisinga berre dreier seg om å fylgja rekneregler, tvingar ho studentane til å halda seg på nivå 1 og 2 i modellen. Dette kan gjerne vera ei god tilnærming på eit tidleg stadium i opplæringa, og dermed er det ikkje sikkert at Papert har rett i at den naturlege tilnærminga alltid er gjennom utforsking. Derimot, for å oppnå den intuisjonen som krevst av eksperten på nivå 5, so må studenten før eller sidan ta steget vidare og la seg oppsluka av problemet og finna mål og meining for seg sjølv.

2.3 Oppdaging mot prov

H. Dreyfus og Dreyfus (1988) merkjer seg at ekspertise vert stadig mindre verdsett i vår tid. Store verksemder krev argumentasjon som fleire personar kan ettergå. Industrileiarar må forsvara avgjerdslene sine overfor styret og aksjonærane, og offentlege byråkratar må kvalitetssikra kvart vedtak fleire gongar. Dette krev òg at ein går tilbake til lågare nivå, og støttar seg på reglar og annan proposisjonell kunnskap, som kan forklara vedtaket.

I matematikken, og like gjerne i programutvikling, ser me eit tydeleg skilje mellom ei oppdagingskontekst og ei grunngjevingskontekst[6]. Når ein finn eit nytt resultat i matematikken, må ein prøva og feila. Ein gissar og freistar å prova gissinga, og ein omdefinerer symbol og omgrep etter som ein finn feil i resonnementet. Her er det intuitive ekspertisen som gjeld. Når resultatet skal

[5] Jf. *Dasein*-omgrepet hjå Heidegger

[6] *Context of Justification* og *Context of Discovery*, sjå Reichenbach (1938)

publiserast, må ein reinskriva provet og validera det etter logiske reglar. Det vert ståande som autoritativ dokumentasjon på at resultatet er korrekt, men seier ofte lite om korleis forfattaren kom på resultatet i fyrste omgang. Situasjonen i programutvikling liknar, gjerne med eigne testarar i grunngjevingskonteksta.

Forholdet mellom oppdaging og grunngjeving fører til eit anna paradoks i skulematematikken (Papadopoulos & Iatridou, 2010). Undervisninga fylgjer som oftast modellen etter Bourbaki (ibid.), heilt og fullt henvist til grunngjevingskonteksta. Oppdagingskonteksta vert neglisjert, og studentane får ofte ikkje sjå korleis matematikarar arbeider i praksis, sjølv ikkje på universitetet etter meir enn tolv år med skulematematikk. Difor får me den vanlege misoppfattinga som Schoenfeld (1988) peikte på. Elevane freistar å finna løysinga slik som læraren demonstrerer ho, og ventar at dei skal klara det like raskt og på fyrste forsøk. Mange gjev opp dersom dei ikkje ser løysinga med ein gong, og det er ikkje rart når læraren aldri har vist dei korleis han sjølv prøver og feiler når han løyser eit problem for fyrste gongen.

Paradoksalt nok observerte Adner (1990) at *både* oppdagingsprosessen og argumentasjon kjem betre fram når studentane programmerte. Studien hennar liknar Sfard og Leron (1996)[7], men ho gav same spørsmål som algebra- og programmeringsoppgåve utan å skilja mellom eit generelt problem og eit særtilfelle. Studentane hadde same bakgrunn, med det same innslaget av matematikk- og programmeringsopplæring, og hovudkonklusjonen hennar var at observerte skilnader mellom programmering og algebra *ikkje* kjem av at observerte studentar kan ha ulik bakgrunn, men av sjølve mediet. Når studentane i Adners studie programmerte, delte dei problemet opp i einskildsteg, namngav alle variablar til bruk i mellomrekningar og forklarte framgangsmåten med kommentarar i koden. Når dei skreiv algebraiske modellar, gav dei ofte berre den komplette modellen, utan forklaring og med mellomrekningane forenkla bort.

2.4 Vurdering mot eksperiment

Ei anna styrke ved programmering er at det skaper det som Brousseau (1997) kaller eit adidaktisk potensial. Det vil seia at situasjonen gjev naturleg tilbakemelding uavhengig av læraren. Leron og Hazzan (1998) peiker på at datamaskina gjev objektiv respons på det som studenten gjer utan på nokon måte å døma eller vurdera verdi. Det vil mellom anna seia at studenten kan fokusera på korleis maskina og programmeringsspråket verkar, utan å måtte tenkja på kva læraren *meiner*. Til skilnad frå lærare, er datamaskina lydig men obsternasig, og krevjande men konsekvent og forutseieleg (Sfard & Leron, 1996). Slike adidaktiske situasjonar gjev ein form for *fair play* som er vanskeleg å etterlikna (Samaniego & Barrera, 1999). Læraren sine verdisyn, kjensler og andre subjektive oppfatningar vert tekne heilt ut av situasjonen. Naturen og teknologien lever sitt eige liv, og studenten får samhandla med det utan innvending. Studenten sine for-

[7] Adner nemner òg fleire liknande studiar tidleg på 1980-talet.

søk vert dermed omsett til opplevingar[8] og skaper ein umiddelbar samanheng mellom analytisk og empirisk kunnskap[9].

Mariotti (2002) ser på programmeringa som mediering, der maskina medierer mellom studenten og den abstrakte matematikken. Røynde matematikarar kan moglegvis ha liknande oppleving gjennom tankeeksperiment i abstrakte medium som algebra og matematisk analyse, men mindre modne matematikkstudentar treng rimeligvis mediering i eit meir konkret medium.

Ei utfordring i prosjekttilnærminga er at mange studentar ikkje er tilstrekkeleg kritiske. Kolikant og Pollack (2004) observerer korleis studentar som arbeider direkte på maskina, utan forutgåande analyse, ofte argumenterer for at løysinga er korrekt basert på svært få testar. Dei ser ein konflikt mellom dei positive resultata rapportert av Sfard og Leron (1996) og eigne observasjonar. Ben-Ari (2001) kallar denne *bricolage*-tilnærminga direkte harmfull.

2.5 Drøfting

Programmeringsparadokset framhevar to ulike tankesett. På den eine sida har me den regelstyrte, logiske fornufta, som er hensiktsmessig både for nybyrjarar og i grunngjevingskonteksta. Dreyfus og Dreyfus nemner òg at han er relevant for ekspertar som står overfor nye eller overraskande problem. På den andre sida har me intuitiv prøving og feiling, som forsovidt både kan gjelda uvørne nybyrjarar som ofte må prøva på nytt, og eksperten med god flyt og få feil.

Det er rimeleg å tru at heller ikkje prosjekttilnærminga, trass i Paperts optimisme og Sfard og Lerons positive observasjonar, er noka sylvkule. Det er positivt at programmeringsprosjektet inspirerer studentane til å prøva og feila *meir*[10], men dei må òg læra å sjå etter og oppdaga dei feila som dei gjer, og prøva på nytt. Det ser ikkje ut til at alle lærer det på eiga hand.

Det regelbaserte tankesettet byggjer på proposisjonell kunnskap. Me kan forklara til studentane korleis dei validerer eit prov. Intuisjon krev ei forståing, eller taus kunnskap, som me ikkje kan forklara. Der er ulike meiningar om kva intuisjon er. Simon (sitert av Kahneman, 2011, s. 11) hevdar at det ikkje handlar om noko anna enn å kjenna igjen reglar eller døme som ein har møtt tidlegare. S. E. Dreyfus (2014) skil derimot mellom prosedyreminne og deklarativt minne, og hevdar at intuitive handlingar spring direkte ut frå prosedyreminnet utan å gå vegen om deklarativ kunnskap. Medan Simon og Dreyfus legg vekt hovudsakleg på røynsler som kjelde til lærdom, meiner Kemp (2013, s. 180f) at ogso den tause forståinga[11] kan formidlast, men berre i direkte kontakt mellom menneske.

Kemp (2013) og Schön (1987) presenterer undervisning og læring som *kreativ imitasjon* til skilnad frå kopiering. Målet er ikkje at studenten skal kopiera løysingane åt læraren. I staden er målet å imitera læraren sitt tankesett når han løyser oppgåver. Det er ikkje nok å kopiera dette på liknande oppgåver, for kopien er ikkje overførbar til nye situasjonar. Det som er overførbart er førebiletet

[8] Jf. Wittgenstein (1986)
[9] Hoyles og Noss (2003) siterer diSessa (2000).
[10] Sjå Heid (1997) som refererer til Blume og Schoen (1988).
[11] Kemp ser her på *hermeneutisk* forståing, sjå Schaathun (2022b).

og tankesettet åt læraren, men dette må tolkast på nytt kvar gong ein står i ein ny og ukjend situasjon.

Forståing kan nok like fullt krevja røynsler, men Kemp og peiker på to viktige poeng. For det fyrste handlar forståing ikkje om å kopiera løysingar eller fylgja reglar, men å tolka gamle løysingar og reglar på nytt i nye situasjonar. For det andre er det mogleg for læraren å bidra til denne utviklinga som førebilete. Det siste krev derimot, som ogso Rogers og Freiberg (1994) peiker på, at studenten vert verdsett for å tenkja nytt og originalt, og ikkje berre for å gjera det same som alle andre.

3 Forskingsdesign

Programmeringsparadokset reiser fleire interessante spørsmål. Det store og langsiktige spørsmålet for oss er,

> *Q⋆ Korleis kan me best leggja til rette for at studentane kan læra å arbeida med matematikk i oppdagingskonteksta?*

Den primære målgruppa for oss er forkurset til ingeniørstudiet, og altso studentar som les naturfagleg matematikk for VGS (R1 og R2).

For å svara på $Q\star$ treng me derimot meir informasjon om korleis studentane faktisk arbeider med programmering og med matematikk, og korleis dei ser samanhengen mellom dei to. Nesten tretti år er gått sidan Sfard og Leron gjorde sitt arbeid, og det er ikkje sikkert at resultatet framleis er relevant i ei ny tid, på ein annan plass og for nye studentgrupper. Dette vert dermed det mest umiddelbare forskingsspørsmålet.

> *Q*1 *Korleis arbeider forkursstudentar midt på 2020-talet med matematiske problem og med matematiske programmeringsoppgåver?*

Meir konkret vil me òg sjå spesielt på kva matematikkonsept og -kompetansar[12] studentane bruker når dei programmerer. Dette adresserer eit spørsmål frå Heid (1997), om læring frå programmeringsprosjektet er overførbare til matematikk. Adner (1990) observerte at overføringa ikkje skjer av seg sjølv. Med betre kunnskap om kompetansen som studentane allereie overfører, kan me sidan byggja på dette for å stimulera til auka overføring. Difor spør me:

> *Q*2 *Kva matematikkompetansar og -konsept bruker studentane når dei programmerer?*

Me skal skissera ein studie som me planlegg å gjennomføra til våren, mot slutten av forkurset, for å svara på $Q1$ og $Q2$.

[12] Med *matematikkompetansar* tenkjer me på taksonomien etter Niss og Jensen (2002), samt liknande taksonomi frå læreplanen.

3.1 Designforsking som metode

Det langsiktige forskingsspørsmålet ($Q\star$) er eit designspørsmål. Det spør ikkje om korleis verda er (korleis studentane lærer), slik som vitskapane vil gjera, men heller om korleis verda (utdanninga) *bør* vera. Det er difor naturleg å byggja forskingsmetoden på designfagleg metodeteori. Cobb mfl. (2015) presenterer designforsking (*Design Research*) i pedagogikk og utdanningsforsking. Dette skil seg frå det meste av metodeteorien i t.d. industridesign, ved å leggja like stor vekt på deskriptive (vitskaplege) som normative mål. Dette er likevel ikkje ein kategorisk skilnad. All design krev kunnskap om *status quo*, og må ofte gjera deskriptive studiar, men utdanningsforskinga med sine røter i sosiologi og psykologi stiller gjerne større krav til vitskaplegheit i denne delen av forskinga.

Designforsking er ein iterativ prosess og byggjer på den same *iterative refinement* som me ogso nemnde som aktuell for læring. Typisk utviklar ein læringsaktivitetar for å nå konkrete læringsmål, for deretter å studera korleis studenten eller læringsmiljøet arbeider med aktiviteten. Det skal både gje ein betre forståing av korleis studentane lærer og forstår stoffet *og* informera vidareutvikling av læringsaktiviteten til neste runde.

I tillegg til Cobb mfl. (2015), byggjer me forståinga vår av designforsking på klassisk designmetode, og særleg på ein konsolidering av Donald Schön og Herbert Simon (sjå Schaathun, 2022a). Cobb *et al.* skriv berre om felteksperiment. For oss er det viktig ogso å bruka tankeeksperiment Schön (1983) i utviklinga av læringsaktivitetar, for å sikra best mogleg aktivitet med minst mogleg ulempe og risiko for studentane.

3.2 Gjennomføring av studien

Som det fyrste steget for å svara på $Q1$ og $Q2$, er det naturleg å gjenta studien etter Sfard og Leron, for å sjå om observasjonane deira stadig er relevante tretti år etter. Me skal samanlikna to klasser, der den eine løyser rekneoppgåva og den andre programmeringsoppgåva, under mest mogleg like tilhøve. Det kan vera aktuelt å la kvar klasse løysa den andre oppgåva etterpå, slik at alle får det same tilbodet.

Forskingsspørsmåla er hovudsakleg kvalitative. Me har ingen spesifikk førehandshypotese om kva kompetansar, konsept og arbeidsformar me vil finna. Ved å bruka kvalitative metodar, kan me nærma oss studentane med eit ope sinn. Den mest objektive informasjonen får me ved å observera studentane medan dei arbeider med oppgåvene. Då får me data om kva studentane faktisk gjer, til skilnad frå ulike intervjumetodar som berre fortel kva studentane hugsar og kva dei opplever i ettertid.

Ulempa med observasjon er at det er vanskeleg å sjå kva studentane tenkjer. Intervju kan løysa dette problemet, men fangar likevel ikkje tankane under sjølv oppgåveløysinga. For å observera kva studentane tenkjer, må me få dei til å prata medan dei arbeider. Høgttenkjingsprotokollar vert brukt ein del i designforsking, men me må vakta oss for å påleggja studentane å gjera ting som tek merksemda bort frå sjølve oppgåva. Dei er der for å læra og ikkje for å informera forskinga.

Me legg i staden opp til at studentane arbeider to og to, med instruks om heile tida å forklara kvarandre korleis dei tenkjer, og verta samde om vidare løysing. I programmering er denne tilnærminga kjend som parprogrammering, der to programmerarar arbeider på éi maskin. Dette gjev ofte gode resultat, sjølv om det er omdiskutert (t.d. Wei mfl., 2021). Som me skal koma tilbake til i læringsdesignet, har slikt samarbeid og sosiale læringsaktivitetar fleire didaktiske føremonar, og me reknar difor med at dette opplegget ogso gjev best mogleg læring. I den analytisk-matematiske oppgåva kan ein kanskje få ein effekt som minnar om parprogrammering, ved at to studentar arbeidar saman om å føra løysinga på éi tavle, som i det tenkjande klasserommet som Peter Liljedal Liljedahl og Wheeler (t.d. 2023) reklamerer for.

I tillegg til å observera kva studentane seier, er det naturleg å observera kva dei skriv, både på skjermen og på tavla. Ved å bruka skjerm og tavle, er det mogleg å ta *screencast* og videoopptak av tavla, i tillegg til lydopptak, som gjer det mogleg å fulltranskribera ein protokoll og anonymisera eit datasett for framtidig bruk. Me legg opp til å observera tre studentpar for kvar av dei to oppgåvene (prosjekt og problem). Deltaking er sjølvsagt friviljug, og deltakarane vert fordelte tilfeldige på dei to oppgåvene.

4 Læringsdesign

Forkurs for ingeniør- og sivilingeniørutdanning er eit årsstudium som kvalifiserer studentane til vidare ingeniørutdanning. Fullt forkurs inneheld faga norsk, engelsk, matematikk, fysikk, samt teknologi og samfunn. Kurset skal dekkja mykje innhald frå vidaregåande skule, og matematikken tilsvarar i hovudsak R1 og R2, med mindre tilpassingar til ingeniørutdanninga.

Fagfornyinga (LK20) frå 2020 (Utdanningsdirektoratet, 2020) styrkar to element som gjer programmeringsparadokset aktuelt. For det fyrste skal studentane no konkret læra å programmera som ein del av matematikkfaget. For det andre legg læreplanen vekt på matematikkompetansar[13] eller kjerneelement, som går på tvers av dei tradisjonelle kunnskapsområda. Særleg merkjer me oss utforsking og problemløysing som programmeringsprosjektet kan henda kan vera med å fremja. For å møta LK20, fekk forkurset nytt emneinnhald og nye lærebøker i matematikk i 2022.

Forkurset ved NTNU i Ålesund har to matematikklærarar som underviser to klasser med om lag 40 studentar kvar. Kvar klasse har inntil tolv undervisningstimar i veka over to semester, i flate klasserom. Kurset krev fysisk oppmøte (høgst 20% fråver). Timane vert brukte til både lærarstyrte aktivitetar og til sjølvstendig arbeid under rettleiing, og lærarane vert normalt godt kjende med studentane våre. Slik minner forkurset på mange måtar meir om vidaregåande skule enn høgare utdanning.

[13] Forståinga av matematikkompetansar her i landet og i Europa byggjer i stor grad på arbeidet åt Mogens Niss i KOM-prosjektet (t.d. Niss & Jensen, 2002). Ordbruken i læreplanen er ikkje heilt den same, men elementa er gjenkjennelege.

4.1 Læringsaktiviteten

Oppgåvene frå Sfard og Leron (1996) lyder på norsk som fylgjer:

- **P1:** Gjeve tre punkt, $(2, 3)$, $(-1, 4)$ og $(0, -1)$ i planet, finn sentrumet og radien til sirkelen gjennom dei.
- **P2:** Skriv eit dataprogram som tek inn tre vilkårlege punkt i planet (gjevne ved koordinatane deira) og returnerer sentrumet og radien til sirkelen gjennom dei.

Mot slutten av skuleåret skal studentane kunna både programmeringa, geometrien og algebraen som dei treng for å svara på både P1 og P2. Som repetisjonsoppgåver bør dei difor vera godt egna. Tidlegare i skuleåret kan dei derimot vera vanskelige å bruka, fordi studentane må kombinera kunnskap frå fleire område av pensum.

Fordi studentane er venta å prøva seg fram, er det sannsynleg at dei treng relativt mykje rettleiing. For å frigjera ressursar til dette, gjennomfører me øvinga som ein del av stasjonsundervising, der klassa vert delt i tre grupper som rullerer mellom tre stasjonar. På den eine stasjonen arbeider dei med P1 eller P2, og på dei to andre får dei oppgåver der dei er venta å vera mest mogleg sjølvhjelpte. Aktiviteten vert gjennomført i ein firetimarsøkt, slik at dei har ein skuletime per stasjon, og i tillegg tid til ein *debrief* i plenum til slutt. Denne plenumsdelen er venta å fokusera hovudsakleg på P1/P2 og mindre på dei andre stasjonane.

Til datainnsamlinga tek me ut éi gruppe til observasjon på eige rom frå kvar runde i stasjonsundervisinga, slik at me observerer tre grupper frå kvar klasse.

4.2 Den didaktiske situasjonen

Målet med oppgåvene er å trena studentane i problemløysing, der løysinga ikkje er openberr, og dei vil trengja å utforska situasjonen, vurdera alternativ, og prøva og feila. Dette inneber ein risiko for at studentane bruker for mykje tid på blindspor (sjå t.d. Sotto, 2007). Det er viktig å regulera den didaktiske situasjonen slik at alle studentane oppnår eit rimeleg læringsutbyte innanfor oppsett tid.

Det er naturleg å ta utgangspunkt i teorien om didaktiske situasjonar (TDS) etter Brousseau (1997). Me har allereie vore inne på det adidaktiske potensialet, som er eit aspekt frå TDS. Den didaktiske situasjonen er bestemt av både læraren, studenten og omgjevnadene, der det siste omfattar både oppgåva, utstyr som datamaskin og tavle, samt klassemiljø med medstudentar, m.m. Det adidaktiske potensialet, som me har nemnd, er eigenskaper ved situasjonen der læraren ikkje medverker. Dette er viktig ikkje berre fordi læraren ikkje kan hjelpa alle studentane til ei kvar tid, men òg fordi det gjer studentane meir autonome i situasjonen, slik at dei kan sjå problemet for seg sjølv utan læraren si fortolking.

Den didaktiske kontrakten, som kan vera meir eller mindre uttrykkeleg formulert, regulerer forholdet og forventingane mellom student og lærar. Oppgåveteksta er ein sjølvsagt del av kontrakta, men like viktig er studenten si (mis)oppfatning av kva slags svar læraren forventar. Den didaktiske kontrakten

på forkurs er prega av tradisjonell matematikkundervising, der læraren forklarer kjende rekneteknikkar, og studentane bruker dei til å løysa relativt standardiserte oppgåver. Problemløysinga i P1/P2 krev ein annan kontrakt, men det er ikkje trivielt å bryta med den etablerte. Som Schoenfeld (1988) har vist, ligg der gjerne inngrodde førestillingar til grunn for korleis studentane arbeider med matematikk.

Sjølv om der er grenser for kor mykje ein kan avvike frå den vande kontrakten på kurset, kan me freista å endra ein del ting for den aktuelle øvinga. Det gjer me når me forklarer oppgåva og køyrereglane for økta. Dette kaller me for devolusjon i TDS, fordi det overfører agensen frå læraren, som har planlagd økta, til studentane som skal arbeida mest mogleg autonomt under gjennomføringa. Ved vellukka devolusjon overtek studenten heile eigarskapet til problemet og dei utfordringane som det byd på. Vonleg gjev det indre motivasjon (Brousseau, 1997). Det store spørsmålet er korleis me skal utforma køyrereglane og kontrakten for å oppnå dette, slik at studentane faktisk prøvar seg fram i problemet.

Sfard og Leron (1996) la vekt på at læringsdesigneren fyrst og framst må kjenna pulsen på læringsmiljøet, med dei normane og verdiane som gjeld. Studentane deira opplevde klasserommet og datalabben som ulike verder. Det er truleg ikkje lenger tilfellet, sidan me ikkje lenger bruker datalabbar, og studentane sit med eigne maskiner i vanlege klasserom. Målet med økta er heller ikkje å introdusera dei til eit nytt arbeidsmiljø, men å innføra utforskande problemløysingsteknikkar med prøving og feiling i det ordinære arbeidsmiljøet. Det fysiske læringsmiljøet vil difor fylgja den faste didaktiske kontrakten. Alle slags hjelpemiddel er tilgjengelege, med lærebok, formelhefte, notat, kalkulator, bærbar datamaskiner og andre einingar med nettverkstilkopling. Dei programmerer i Python og Jupyter Notebook, som vert brukt vidare i studiet.

I den didaktiske situasjonen vert studentane overlatne til oppgåva, og for mange er det vanskeleg å koma i gang. Det er her me meiner at gruppearbeid har verdi, fordi to hovud tenkjer betre enn eitt, og når ein student køyrer seg fast, kan ein håpa at den andre kan sjå vidare. Dette krev at studentane deler idéar og ikkje er redde for å gjera feil. Ein dårleg idé kan ofte hjelpa makkaren til å sjå ein mindre dårleg idé. Me kan gjera dette til ein konkret del av den didaktiske kontrakten ved å krevja at studentane arbeider saman på éin flate, dvs. dei som løyser P2 arbeider saman på éi maskin, og dei som løyser P1 arbeider på éi tavle. Idéar, mellomsteg og halvferdige løysingar *skal* gjerast synlege for båe studentane, og dei har felles ansvar for at dei forstår korleis den andre tenkjer.

Idealet er at studentane løyser oppgåva på eiga hand, men det skjer ikkje alltid. Det er viktig at læraren lyttar og observerer, både for seinare å kunna dra samanhengar mellom opplevingane og matematisk teori, og dersom det skulle verta naudsynt, for å rettleia undervegs. Somme tider treng studentane oppmuntring på halvgode idéar før dei misser motet. Andre gongar treng dei ein heilt ny devolusjon fordi dei står heilt fast.

I TDS vert den didaktiske situasjonen fylgt av ein institusjonaliseringsfase der læraren igjen tek kontrollen for å systematisera og formalisera den kunnskapen som studentane vonleg har oppdaga eller vidareutvikla på eiga hand.

Dette inngår i *debriefinga* som nemnd tidlegare. Målet her er å fullføra dei fem praksisane skissert av Smith og Stein (2018), med forventa—observera—velja—setja rekkjefylgje—sjå samanhengar. Læraren kan observera studentane både i arbeidsøkta og i plenumsøkta, og har ansvar for å sjå samanhengane mellom ulike opplevingar og mellom opplevingane og teorien.

4.3 Kritiske vurderingar

Oppgåveformuleringa er kritisk for den didaktiske situasjonen, men me har førebels ikkje problematisert ho. Særleg P1 legg ikkje særleg til rette for den utforskande tilnærminga som me ynskjer. Vidare forsking bør sjå på korleis me kan utforma både P1 og P2 for å stimulera til den prosjekttilnærminga som Papert argumenterer for. Her lyt ein sjå på sokalla LIST-oppgåver (låg inngong stor takhøgd[14]).

Øvinga vert lagd til repetisjonsfasen våren 2025. Det kan vera at studentane då vert mindre kreative og mindre viljuge til å eksperimentera, enn dei er når dei les nytt stoff. Vidare forsking bør sjå korleis ein etter same prinsipp kan laga oppgåver som er egna i den fyrste innlæringsfasen. Repetisjon er sjølvsagt viktig for å styrka grunnleggjande ferdigheiter som studentane treng vidare i studiet, men det er neppe gunstig om studentane møter den utforskande problemløysingsmetoden for fyrste gong når dei er innstille på repetisjon.

5 Oppsummering

Når me legg fram ein reint teoretisk studie utan empiriske resultat, er det fordi programmeringsparadokset reiser so mange interessante spørsmål og moglegheiter, at det er verd å invitera til ein grundig diskusjon før me prøver eit nytt læringsopplegg på studentane. Stasjonsundervisninga gjer det mogleg å observera tre studentpar per klasse med minimal kostnad og minimal påverknad på studiet. Likevel er der alltid risiko når ein eksperimenterer med studentane, uavhengig av om det gjeld forsking eller utvikling, og risikoen lyt me minimera.

Observasjonane frå Sfard og Leron (1996) illustrerer fleire interessante utfordringar i matematikkundervisinga, men òg nye høve til å nå fleire læringsmål. Både utfordringane og høva er underbygde med uavhengig teori. Samstundes er undervising og læring aldri uavhengig av endra tilhøve i verda rundt oss, og tretti år gamle resultat er ikkje alltid relevante i dag. Den føreslegne studien tek sikte på å validera observasjonane frå 90-talet på forkurset vårt i dag.

Vidare studiar bør utvikla eit rikare utval av oppgåver for å testa om me får konsistente resultat. Både ungdomsskulen og høgare utdanning innfører meir programmering i matematikkundervisinga, og ein lyt gjera liknande eksperiment på alle nivå. Det er heller ikkje utenkjeleg at Paperts prosjektvisjon kan vera overførbare til andre fag enn matematikk, men det er eit meir langsiktig prosjekt. Me treng òg vidare forsking på korleis ein kan leggja til rette for å læra utforskande og oppdagande løysingsstrategiar i matematikk.

[14] Konseptet stammar frå NRICH i Cambridge, sjå https://nrich.maths.org/articles/low-threshold-high-ceiling-introduction

Referansar

Adner, H. (1990). Algebraic versus programming approach to solving mathematical modeling problems. *School Science and Mathematics*, *90*(4), 302–16.

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of computers in Mathematics and Science Teaching*, *20*(1), 45–73.

Brousseau, G. (1997). *Theory of didactical situations in mathematics.* Didactique des Mathématiques, 1970–1990 (N. Balacheff, M. Cooper, R. Sutherland & V. Warfield, Red.; Bd. 19). Springer Dordrecht. https://doi.org/10.1007/0-306-47211-2

Cobb, P., Jackson, K., & Dunlap, C. (2015). Design Research. I L. D. English & D. Kirshner (Red.), *Handbook of international research in mathematics education* (s. 481–503). Routledge.

Dreyfus, H., & Dreyfus, S. E. (1988). *Mind over machine* (Paperback).

Dreyfus, H. L. (1992). *What computers still can't do: A critique of artificial reason.* MIT press.

Dreyfus, S. E. (2014). System 0: The overlooked explanation of expert intuition. I *Handbook of research methods on intuition* (s. 15–27). Edward Elgar Publishing.

Hazzan, O., & Goldenberg, E. P. (1997). An expression of the idea of successive refinement in dynamic geometry environments. I E. Pehkonen (Red.), *Proceedings of the Conference of the International Group for the Psychology of Mathematics Education* (s. 49–56, Bd. 3).

Heid, M. K. (1997). The technological revolution and the reform of school mathematics. *American Journal of Education*, *106*(1), 5–61.

Hoyles, C., & Noss, R. (2003). What can digital technologies take from and bring to research in mathematics education? *Second international handbook of mathematics education*, 323–349.

Kahneman, D. (2011). *Thinking, fast and slow.* Farrar, Straus; Giroux.

Kemp, P. (2013). *Verdensborgeren: pædagogisk og politisk ideal for det 21. århundrede* (2. utg.). Hans Reitzels Forlag.

Kolikant, Y. B.-D., & Pollack, S. (2004). Establishing computer science professional norms among high-school students. *Computer Science Education*, *14*(1), 21–35.

Leron, U., & Hazzan, O. (1998). Computers and applied constructivism. I D. Tinsley & D. C. Johnson (Red.), *Information and Communications Technologies in School Mathematics* (s. 195–203).

Liljedahl, P., & Wheeler, L. (2023). *Å bygge tenkende klasserom i matematikk - 14 praksiser for bedre læring.* Cappelen Damm akademisk.

Luntley, M. (2018). Play's the Thing: Wherein We Find How Learning Can Begin. *Journal of Philosophy of Education*, *52*(1), 36–53. https://doi.org/10.1111/1467-9752.12279

Mariotti, M. A. (2002). The influence of technological advances on students' mathematics learning. *Handbook of international research in mathematics education*, 695–723.

Niss, M., & Jensen, T. H. (2002). *Kompetencer og matematiklærin: Ideer og inspiration til udvikling af matematikundervisning i Danmark*. Undervisningsministeriets forlag.

Papadopoulos, I., & Iatridou, M. (2010). Systematic approaches to experimentation: The case of Pick's theorem. *The Journal of Mathematical Behavior*, *29*(4), 207–217.

Papert, S. (1996). An exploration in the space of mathematics educations. *Int. J. Comput. Math. Learn.*, *1*(1), 95–123.

Poincaré, H. (1910). Mathematical creation. *The Monist*, 321–335.

Reichenbach, H. (1938). *Experience and prediction: An analysis of the foundations and the structure of knowledge*. University of Chicago press.

Rogers, C. R., & Freiberg, H. J. (1994). *Freedom to learn* (Third). Macmillan College Publishing Company.

Samaniego, A., & Barrera, S. V. (1999). Brousseau in Action: Didactical Situation for Learning How To Graph Functions.

Schoenfeld, A. H. (1988). When Good Teaching Leads to Bad Results: The Disasters of 'Well-Taught' Mathematics Courses. *Educational Psychologist*, *23*(2), 145. https://www.tandfonline.com/doi/abs/10.1207/s15326985ep2302_5

Schön, D. A. (1983). *The Reflective Practioner*. Ashgate Arena.

Schön, D. A. (1987). *Educating the reflective practitioner*. Jossey-Bass San Francisco.

Schaathun, H. G. (2022a). Where Schön and Simon agree: The rationality of design. *Design Studies*, *79*, 101090. https://doi.org/10.1016/j.destud.2022.101090

Schaathun, H. G. (2022b). On Understanding in Mathematics. *Teaching Mathematics and its Applications: An International Journal of the IMA*, *41*(4), 318–328. https://doi.org/10.1093/teamat/hrac016

Sfard, A., & Leron, U. (1996). Just give me a computer and I will move the earth: Programming as a catalyst of a cultural revolution in the mathematics classroom. *International Journal of Computers for mathematical learning*, *1*, 189–195.

Smith, M., & Stein, M. K. (2018). *5 Practices for orchestrating productive mathematics discussion*. National Council of Teachers of Mathematics.

Sotto, E. (2007). *When Teaching Becomes Learning: A Theory and Practice of Teaching* (2nd). Continuum.

Utdanningsdirektoratet. (2020). *Læreplan i matematikk fellesfag Vg1 teoretisk* (tekn. rapp.) (gjeldande frå 1. august 2020). https://data.udir.no/kl06/v201906/laereplaner-lk20/MAT09-01.pdf

Wei, X., Lin, L., Meng, N., Tan, W., Kong, S.-C., & Kinshuk. (2021). The effectiveness of partial pair programming on elementary school students' Computational Thinking skills and self-efficacy. *Computers & Education*, *160*, 104023. https://doi.org/10.1016/j.compedu.2020.104023

Wittgenstein, L. (1986). *Philosophical Investigations* (3. utg.) [Translated by G. E. M. Blackwell. First published 1953.]. Basil Blackwell Ltd.