Ejerskab og læringsstrategier med visuel programmering som tredje sprog

Ture Reimer-Mattesen

master i ikt og læring

Masterprojekt, 4. Semester, Maj 2017

Vejleder: Thorkild Hanghøj

Ejerskab og læringsstrategier med visuel programmering som tredje sprog

master i ikt og læring

Udarbejdet af: Ture Reimer-Mattesen, 20132266

Masterprojekt, Master ikt og læring, 4. semester, Maj 2017

Vejleder: Thorkild Hanghøj, Lektor v. Institut for Kommunikation, København

Opgavens omfang: 143588/59,8ns

it-vest
samarbejdende universiteter

AALBORG UNIVERSITET

CBS COPENHAGEN BUSINESS SCHOOL HANDELSHØJSKOLEN

AARHUS UNIVERSITET

ABSTRACT

Through a Design Based Research methodology, a didactic design for visual programming in mathematics has been developed. The research project presents a didactic model and materials, which support teachers without prior experience in incorporating programming in mathematics. The model is based on didactic principles for computational thinking and theory of design framework. The project illustrates how students develop ownership of their own learning process, how the visual programming languages support their learning and in what ways the students develop computational thinking as a problem solving skill.

An intervention design in the form of 2 courses targeted a 5th and 7th grade has been developed and implemented. Qualitative interviews with pupils and teachers has been conducted as research method.

Using Vygotsky's socio-cultural theory, the visual programming language is described through theories of spoken language and written-language and their significance for cognition. The study shows signs of the students developing a strong context dependent, logically constructed "third language" with fix points in the semantics of the visual programming language. This language supports students in programming and thinking with code.

The study also shows that learning is supported by Scratch redundant multimediation of text, color and form. The study concludes that all the students in the study make use of the computational thinking strategies: Algorithmic thinking, decomposition, generalization and pattern recognition. The study does not show, whether the students could use these problem solving strategies across contexts. Further research is required.

The first iteration of the intervention design was completed and recommendations for the next iteration presented. Further iterations are recommended for further exploration of the findings and to make the didactic design more robust.

Keywords: Design Based Research, computational thinking, visual programming, scratch

1 INDHOLD

2 INTRODUKTION OG MOTIVATION

Rundt omkring på mange skole eksperimenteres der blandt mere nørdede ildsjæle med programmering, makerspaces, robotter og 3d printere. Foreningen "Coding Pirates" tilbyder fritidsaktiviteter, hvor eleverne udfolder sig på den store eksperimentelle, teknologiske klinge. Alineas læremiddelpris gik i år til materialet "Lær at kode" (Grynberg, 2016). "Coding Class" og foreningen IT-Branchen har sat programmering på skoleskemaet på en række skoler – dog som valgfag (IT-Branchen, 2017). Mange af de lande vi sammenligner os med, har indført "datalogi" som fag på skoleskemaet. Horizon Report 2017 (Adams Becker, S, Cummins, Freeman, & Rose, 2017) forudser, at to af de stærkeste læringsteknologiske drivkræfter i norden de næste 1-5 år er "Students as Creators" samt "Coding as Litteracy".

2.1 MOTIVATION

Undervisere er interesseret i, at levere spændende og aktuel undervisning, der ruster eleverne til fremtiden. Men folkeskolen står over for en motivationsudfordring. Op mod 60% af eleverne keder sig i undervisningen (Qvortrup, 2016) og danske skoleelever placerer sig under middel i "indsats og udholdenhed" i seneste PISA undersøgelser (Geisnæs & Kirkegaard, 2017). Folkeskolen har det seneste årti gennemgået en rivende it-udvikling. I dag har de fleste elever et digitalt device de bruger dagligt i undervisningen og lærerne er forpligtet på at distribuerer forløb gennem en "læringsplatform". Motivationsudfordringen skyldes således ikke, at lærerne ikke bruger moderne teknologi – men måske snarere måden den bliver brugt på.

Undersøgelser (Hansen & Bundsgaard, 2016) viser, at den prototypiske matematik- og danskundervisning med it er kendetegnet ved, at it anvendes til indlæring af færdigheder, reproduktion og individuelt arbejde. (Hansen & Bundsgaard, 2016, s. 21)

I diskursen om teknologiens rolle og potentiale i undervisningen, står en forestilling om et øget motivationspotentiale når eleverne samarbejder om at skabe, udvikle og arbejde innovativt og kreativt med teknologi stærkt.

I kraft af mit arbejde oplever jeg, at det en udfordring for mange undervisere, at koble skabende og kreative programmeringsaktiviteter til deres fag på en meningsfuld måde. Mange undervisere vil gerne en masse, men mangler teknisk og didaktisk know-how. Jeg oplever, at mange kreative teknologi aktiviteter, som f.eks. at programmere, i praksis foregår uden for fagene eller som noget sekundært i forhold til fagenes indhold. Udgangspunktet for min interesse er begrebet "Computational Thinking". Begrebet er toneangivende i diskursen om det læringspotentiale i form af efterspurgte problemløsningsstrategier, der beskrives som indlejret i det at programmere.

Antagelsen om, at hvis eleverne programmerer så lærer de Computational Thinking, ønskes undersøgt i dette speciale. Samtidig indeholder de fleste fortællinger om programmerende i undervisningen, beskrivelser af eleverne der arbejder engageret og selvstyrende processen. Undersøgelse af sammenhængen mellem programmering, udvikling af Computational Thinking og øget ejerskab i læreprocessen, er drivkraften for dette speciale.

Jeg er endvidere optaget af, hvordan undervisning med programmering didaktisk kan tilrettelægges, så fokus rettes på netop dette og jeg er optaget af, at forstå selve programmeringsprocessen i et kognitivt perspektiv. Hvordan lærer og tænker eleverne med programmering?

3 FORSKNINGSSPØRGSMÅL

Hermed specialets formelle forskningsspørgsmål:

Hvordan kan man anvende et didaktisk design med visuel programmering i folkeskolens matematikundervisning med henblik på at støtte elevernes ejerskab til læreprocesser, deres forståelse af programmering samt udvikle deres problemløsningsstrategier?

Problemformuleringen belyses gennem følgende fire arbejdsspørgsmål:

1) Hvordan udvikler eleverne ejerskab i læreprocessen?
2) Hvordan udvikler eleverne forståelse af programmering som "et tredje sprog"?
3) I hvilken grad og på hvilke måder udvikler eleverne problemløsningsstrategier?
4) Hvordan kan didaktik modelleres, så identificerede potentialer indenfor genstandsfeltet udnyttes?

4 BAGGRUND FOR INTERVENTION OG DATAINDSAMLING

4.1 DATAINDSAMLING

Dataindsamlingen er foretaget i en 5. og 7.klasse på en almindelig 3-sporet folkeskole. Klasserne er begge uden overvægt af elever med signifikante sociale udfordringer.

Det har været væsentligt for anvendeligheden af den viden der skulle udvikles i projektet, at afsættet var en almindelig folkeskoleklasse med en elevgruppe inden for det socioøkonomiske normalområde. Dette har at gøre med anvendelsen af det udviklede designs overførbarhed til lignende kontekster. Den aktuelle skole er valgt, fordi jeg har haft direkte adgang til praksis i kraft af min ansættelse, og jeg har derfor haft indgående domænekendskab ift. skolens kultur, miljø, elever og lærere. Dette har været en praktisk og tidsbesparende gevinst i forhold til at indgå samarbejdsaftaler og med kort varsel foretage ændringer, men omvendt indeholder det at være en integreret del den kontekst man undersøger også en række ricisi. Dette er beskrevet i afsnit 8.5.

Matematikfaget er valgt, fordi udviklingen af de problemløsningsstrategier der udspringer af forskningsspørgsmålet, har paralleller med de mål for de matematiske problembehandlingskompetencer der fremgår af fagets læseplaner (se afsnit 7.37.3). 5. og 7. klassetrin er valgt ud fra overvejelser over hvilke matematiske begreber, der skulle arbejdes med i forløbet. På disse klassetrin er koordinatsystem(5kl), variable og kombinatorisk sandsynlighed (7kl) områder eleverne har arbejdet med tidligere.

4.2 INTERVENTION

Dette speciale tager afsæt i en intervention med det formål, at opnå detaljeret og praktisk baseret viden om visuel programmering i en matematikfaglig kontekst.

De 2 udvalgte lærere har gennemført et undervisningsforløb(**Fejl! Henvisningskilde ikke fundet.**) med visuel programmering i matematik i hhv. 5. og 7.klasse. Forløbet strakte sig over 3-4 uger og var specifikt udviklet som en del af specialets intervention.

Eleverne har skulle programmere løsninger på virkelige udfordringer målrettet autentiske brugere. Der er programmeret i det visuelle programmeringssprog: "Scratch" (Afsnit 6). I 5.klassen skulle eleverne lave et digitalt læringspil, så mindre klasser kunne lære om koordinatsystemet. I 7.klasse

har eleverne skulle arrangere en casino dag på skolen med digitale spilleboder. Selvom der er opsat mål for de matematikfaglige *indholdsområder* i forløbene, og dette er et vigtigt element i forhold til relevansen at programmering i matematik, er der i dette speciale fokuseret på den matematiske problembehandlingskompetence i forhold til Computational Thinking. Det henvises til EMU portalen for uddybende beskrivelser af kompetencer og indholdholdsområder (EMU Danmarks læringsportal, 2017).

De 2 undervisere havde ingen anden erfaring med programmering, end at de hvert år deltager i skolens arrangement: "Hour of code" (https://hourofcode.com/dk), hvor eleverne et par timer stifter bekendtskab med programmering.

5 REVIEWS

Andre forskere har undersøgt forholdet mellem, hvordan der kan arbejdes med didaktik og programmering for at fremme forskellige læringspotentialer. Karen Brennan har i sin afhandling "Best of both worlds: Issues of structure and agency in computational creation, in and out of school" (Brennan, 2013) beskæftiget sig med begreberne Structure og Agency. Agency defineres som den "lærendes evne til at identificere og forfølge mål", og structure defineres som "regler, roller og ressourcer". Undersøgelsen belyser både faktorer der fremmer og hæmmer agency, og specielt i forhold til betydningen af structure i formelle læringskontekster i skolen. Brennan påpeger, at der er ofte en tendens til at lærere hæmmer elevernes agency og derved deres ejerskab og lyst til at lære, ved at anvende høj grad af struktur. I stedet for en enten-eller tankegang om begreberne anvendelse i pædagogiske sammenhænge, foreslår Brennan en både-og tilgang. Baseret på hendes interviews med både undervisere og børn, opstiller hun 5 didaktiske strategier, der strukturerer og fremmer agency i formel undervisning med Scratch.

1. Introducer muligheder frem for færdige læringskoncepter.
2. Støt eksperimentering
3. Støt adgang til relevante "i-rette-tid" hjælperessourcer.
4. Dyrk praksisfælleskab
5. Skab muligheder for refleksion.

Et andet aspekt inden for dette speciales forskningsspørgsmål handler om Computational Thinking i matematikfaglige sammenhænge. Dette har studiet "Developing Mathematical Thinking With Scratch" (Calao, Moreno-Leon, Correa, & Robles, 2015) søgt at belyse. Forskningsspørgsmålet er rettet mod transfer værdien af Computational Thinking som problemløsningskompetence. Forskningen klarlægger, i hvilken grad det gælder, at de problembehandlingskompetencer eleverne udvikler ved at programmere, kan overføres til problembehandling i matematik. En eksperimentklasse arbejdede med programmering i 3 måneder, mens kontrolklassen fortsatte deres almindelige undervisning. Der blev foretaget en effektmåling indenfor 4 matematiske områder: modellering, argumentation, problemløsning og opgaveløsning med brug af algoritmer og procedurer.

Resultatet var, at klassen der havde programmeret, klarede sig signifikant bedre end kontrolgruppen på alle matematiske områder. Særligt inden for de opgaver der skulle løses med algoritmer og procedurer, klarede eleverne sig voldsomt signifikant bedre end kontrolgruppen. Dette forklarer forskningen med, at denne type opgaveløsning genfindes i særlig grad når eleverne programmerer kode i sekventielle trin-for-trin scripts. En del af forklaringen på undersøgelsens resultater forklares også med, at eleverne var meget optaget af og personligt engagerede i at programmere.

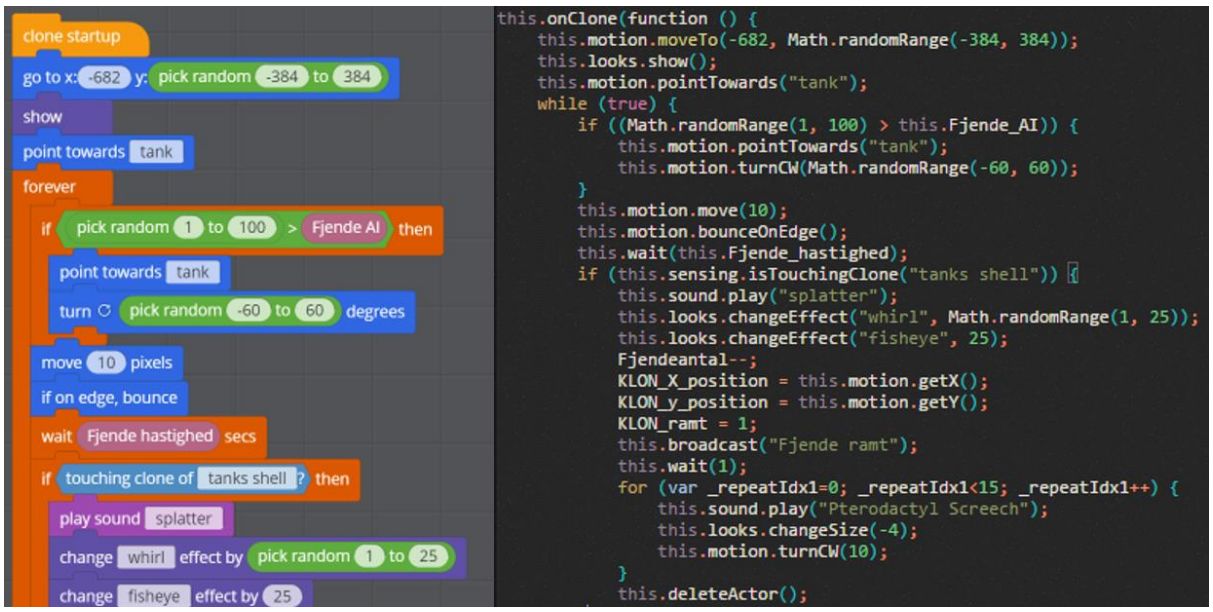
Et sidste omdrejningspunkt i specialet går på sammenhængen mellem ”karakteren” af elevernes tænkning og sprog når de løser udfordringer med visuel programmering. Howland & Good (Howland & Good, 2015) har undersøgt i hvilken udstrækning eleverne udvikler evnen til at kommunikere ”computationelt”, når de arbejder i det visuelle programmeringssprog Flip. Undersøgelsen viste at efterhånden som eleverne anvendte flere computationelle koncepter, udviklede de også et ”language of computation”, hvor de var stand til at fortælle om deres programmer i et naturligt sprog med en høj grad af præcision. Undersøgelsen viser endvidere, at denne præcision i sproget kunne overføres til andre kontekster, hvor eleverne var signifikant bedre til udtrykke og beskrive regler efter inventionen.

6 SCRATCH

Da der løbende vil blive refereret til begrebet ”visuel programmering” samt programmeringssproget ”Scratch”, vil disse kort blive præsenteret her.

”Visual programming language (VPL) is a programming language that uses graphical elements and figures to develop a program.” (Technopedia, 2017)

Visuel programmering alle programmeringssprog der anvender grafiske elementer i udviklingen af kode. I Figur 1 nedenfor, ses to forskellige programmeringssprogs repræsentation af samme kode i hhv. det visuelle programmeringssprog ”Tynker” (www.tynker.com) og det tekstbaserede programmeringssprog ”Java”:



Figur 1: Tynker vs Scratch

Visuel programmering er opstået på baggrund af et ønske om, at kunne reducere den tidskrævende syntaktiske kompleksitet der følger med tekstbaserede kodesprog som f.eks. Java. og derved flytte fokus til kreative og skabende proces.

Blokprogrammering er en særlig type visuelt programmeringssprog, som fungerer ved at man klikker logiske kodeblokke sammen. Scratch (https://scratch.mit.edu/) er et af de mest kendte og anvendte visuelle blokprogrammeringssprog i grundskolesammenhæng. Det er udviklet med et direkte afsæt i

Seymour Paperts konstruktionisme (se afsnit 11.1) og tager udgangspunkt i de tidligere pædagogiske programmeringssprog Seymour Papert var hovedmanden bag – f.eks. LOGO (Resnick, et al., 2009).

Scratch er udviklet af MIT Lifelong Kindergarten Group ud fra konceptet "Kindergarten Play" (Resnick M. , 2007). "Kindergarten Play" er opstået på baggrund af forskning baseret på iagttagelser af børns legende tilgang til læring. Kindergarten Play gennemløber i pædagogiske sammenhænge en cyklisk iteration mellem faserne: Imagine (forestil), Play (leg og byg), Share (del), Reflekter (Reflect). Den type læring har alle naturlige evner for. Kindergarten tilgangen til læring i en programmeringskontekst, hvor man er personligt engageret i at skabe med andre, udvikler kompetencer i både design, kreativ og systematisk tænkning, problemløsning og samarbejde (Resnick M. , 2007).

Med udgangspunkt i dette screenshot af Scratch opdelt i 5 områder (Figur 2), gennemgås brugerinterfacet og der perspektiveres til den legende og eksperimenterende Kindergarten tilgang.

Figur 2: Scratch interaktionsdesign

Når man starter et ny projekt (program) er der helt "blank skærm", ingen kode, ingen baggrund eller grafik. Man starter fra "Scratch" – dvs. fra bunden, og man skaber alting selv, på samme måde som når børn bygger noget i en sandkasse ud fra deres ideer og kreativitet. Derfor anvendes ofte en "sandkasse" metafor, om Scratch og det bærende "Kindergarden Play" koncept.

- **Vindue 1:** Her "afspilles" programmet grafisk, og det færdige program kommer til at se ud som det man kan se i Vindue 1. Man kan hele tiden teste ens kode i vindue 5. Man starter programmet ved at klikke på det grønne flag
- **Vindue 2:** De baggrunde og sprites (grafikelementer) man har brug for indsættes her. Hvert grafikelement programmeres separat i vindue 5.

- **Vindue 3:** Under fanen Scripts er programmeringsblokkene inddelt i 10 let forståelige og farvekodede kategorier som ”Udseende”, ”Bevægelse”, ”Styring” osv. Klikker man på en kategori, vises alle blokkene for den kategori i Vindue 4.
- **Vindue 4:** I vindue 4 kan man se de blokke der hører til en bestemt kategori. Man kan dobbeltklikke på dem for at eksekvere dem, så man kan se effekten på sin Sprite i vindue 1. Man trækker blokkene over i vindue 5.
- **Vindue 5:** Her klikkes blokkene sammen til scripts. Ikke alle blokke passer sammen, men blokkene er formet, så brugerens hjælpes i at samle klodserne på måder der ”fungerer”. I Scratch virker alt hvad man laver – men måske ikke helt efter hensigten.

Scratch er et visuelt programmeringssprog, der med sin sandkassemetafor som designprincip, opfordrer til en legende og undersøgende tilgang til programmering. En anden væsentlig årsag til at Scratch, til trods for andre visuelle blokprogrammeringssprogs fremkomst, stadigvæk er foretrukket hos mange, er det enorme online Scratch fællesskab hvor millioner af brugere deler projekter, hjælper hinanden, opretter studiegrupper osv.

7 COMPUTATIONAL THINKING

Computational thinking (herefter CT)T blev introduceret i 1996 (Repenning, Basawapatna, & Escherle, 2016) af Seymour Papert og er i høj grad blevet aktualiseret og konceptualiseret i artiklen ”Computational Thinking” af Jeanette Wing (Wing J. , 2006). I artiklen argumenterer hun for, at CT ikke blot er nogle teknikker ”computer-scientists” eller dataloger har brug for, men at det er en generel kompetence alle har brug for i det 21.århundrede. Hun definerer CT som:

“CT is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an informationprocessing agent” (Wing J. , 2011)

I forhold til disse løsninger præciserer Wing yderligere: *”The solutions can be carried out by a human or machine, or more generally, by combinations of humans and machines”* (Wing J. , 2011)

Wing beskriver CT som en måde at identificere problemer og skabe løsninger, det matcher samfund der i stigende grad digitaliseres. Hun taler for at CT er en helt generel problembehandlingskompetence der er anvendelig i helt almindelige hverdagskontekster. Hun påpeger, at hovedformålet med CT ikke er at lære at programmere som sådan. Mennesker kan også ”compute” og udføre ”kommandoer”. Det handler i højere grad om at kunne arbejde analytisk med problemer. Ved at tænke som en ”computer scientist” kan man beskrive og løse problemer på en måde, der vil kunne kodes, beriges og opskalleres af teknologi. Denne måde at problembehandle på indebærer at bryde problemer op i mindre delproblemer (dekomposition), identificere kernen i dem(abstraktion), findes eksisterende løsninger, der kan bruges på delproblemerne(mønstre) og udvikle generelle løsninger der kan bruges på andre tilfælde (algoritmer) (BBC Bitesize, 2017). Løsninger, der bygger på disse principper, er kendetegnede ved logiske, sammenhængende slutninger og trin-for-trin beskrivelser af fremgangsmåder, der netop er velegnede at programmere.

7.1 CT – DE 4 HJØRNESTEN

BBC Bitesize (BBC Bitesize, 2017) er en gratis studieunderstøtte online ressource for studerende i Storbritannien. BBC Bitesize udleder ”4 hjørnesten” i CT. Disse 4 hjørnesten præsenteres her,

sammen med en eksemplificering af, hvordan det kan komme til udtryk i programmeringssammenhænge:

1. **Algoritmer (Algorithms) – trin-for-trin, regler**
   De scripts eleverne skaber i Scratch eksekveres alle sekventielt fra toppen og ned. Det betyder, at et script fungerer som en algoritme for en specifik programfunktion.
2. **Dekomposition (Decomposition) – at nedbryde i enkeltdele**
   Når eleverne står med en ide de skal programmere, er dekomposition et udtryk for, at analysere deres ide og udlede alle de elementer der skal "spille sammen". Hver del er en udfordring der skal løses, og ved at have overblik over hvilke delelementer deres ide består af og hvordan de enkelte dele spiller sammen, skaber de bedre løsninger. Dekomposition foregår også på script niveau bl.a. når eleverne fejlfinder deres kode ved at skille den ad i dens enkeltdele.
3. **Mønstre og generalisering (Patterns and generalisation) – at spotte og anvende ligheder**
   Når eleverne møder udfordringer i deres scripts er det oplagt at kigge efter fungerende algoritmer, der løser deres udfordring. Dem kan de finde masser af på Scratch hjemmesiden.
4. **Abstraktion (Abstraction) – at udlede de væsentligste**
   Når eleverne programmerer, er det at kunne udvælge hvilke data en computer skal simulere eller modellere og hvilken repræsentation der skal vise det, en meget vigtig kompetence i forhold til om løsningens kvalitet. Her handler det om at fokusere på det væsentlige. Udvikling af programmer som "proof of concept" (det kan lade sig gøre!) eller prototyper er udtryk for abstraktion

7.2 CAS BAREFOOTS DIDAKTISKE PRINCIPPER TIL CT

CAS Barefoot er et statsstøttet projekt, der blev etableret i 2014 for, at støtte i Storbritanniens lærere i, at undervise i faget "Computing" med afsæt i fagets CT funderede curriculum (CAS Barefoot, 2014). CAS Barefoot har udviklet en række tilgange til at arbejde med CT (CAS Barefoot - CT Approaches, 2014). Tilgangene skal forstås som væsentlige didaktiske principper i en undervisning, hvor eleverne skal have mulighed for at udvikle kompetence i CT. Disse didaktiske principper indgår, som designprincipper i udviklingen af interventionen (se afnit 12). Her præsenteres de kort, og med inspiration fra CAS Barefoot beskrivelser, eksemplificeres de i en programmeringskontekst.

- **Eksperimentering (Tinkering)**
  I det didaktiske design for programmering, skal eleverne ikke have at vide præcist hvordan de skal skabe deres program . Åbne spørgsmål og opgavetyper, hvor der ikke kun er et rigtigt svar eller en løsningsmetode opfordrer til eksperimentering, kreativitet, divergent tænkning, personligt engagement og vedholdenhed i læreprocessen. (CAS Barefoot - CT Approaches, 2014)
- **Kreativ skaben(Creating)**
  Programmering er en kreativ proces. Det involverer både originalitet og værdiskabelse. Det handler om, at det man skaber er én løsning blandt mange som man selv har skabt, at det man skaber skal løse et problem eller en udfordring for andre eller at det har et reelt publikum. Det at kunne være original betyder, at det man skaber er unikt og at man ikke blot udfylde en skabelon eller blot kopier andres 1:1.
- **Fejlsøgning (Debugging)**
  At lave fejl er helt grundlæggende uundgåeligt når man programmere unikke løsninger. Det

handler om at undervisning med programmering fokuserer på fejlsøgning som en vigtig komponent i at udvikle bevidste problemløsningsstrategier (som f.eks CT)

- **Vedholdenhed (Perservering)**
  Programmering er hårdt! Det kræver villighed til ikke at give op selv om det er svært og frustrerende. Undervisning med programmering skal indtænke disse frustrationer, som kilde til udvikling af bevidsthed om egen læring. Samtidig må eleverne heller ikke blive *for* frustrerede. Den overordnede didaktiske ramme må tage hensyn til dette.
- **Kollaboration (Collaboration)**
  "Par programmering" ses som en særlig effektiv måde at programmere på og denne eller andre kollaborative arbejdsformer skal indtænkes i undervisning med programmering – også selv om eleverne arbejder ved hver deres skærm.

7.3 CT I MATEMATIK

I dette speciale fokuseres der *mindre* på om eleverne lærer de matematiske indholdsområder koordinatsystem, variable og kombinatorisk sandssynlighed, og *mere* på, om eleverne udvikler relevante matematiske problemløsningsstrategier.

Som det fremgår af målbeskrivelserne for det udviklede undervisningsforløb (**Fejl! Henvisningskilde ikke fundet.**), er en del af målsætningen for begge klasser centreret omkring den matematiske problembehandlingskompetence (EMU Danmarks læringsportal (2), 2017):

4.-6. klassetrin

Matematiske kompetencer

Problembehandling
Fase 2

Færdighedsmål
Eleven kan anvende forskellige strategier til matematisk problemløsning

Vidensmål
Eleven har viden om forskellige strategier til matematisk problemløsning, herunder med digitale værktøjer

7.-9. klassetrin

Matematiske kompetencer

Problembehandling
Fase 1

Færdighedsmål
Eleven kan planlægge og gennemføre problemløsningsprocesser

Vidensmål
Eleven har viden om elementer i problemløsningsprocesser

Som det fremgår af ovenstående delmål for forløbene, handler disse mål i høj grad om at kunne anvende strategier i problemløsningsprocesser. Jeg vurderer, CT strategierne er relevante ind i matematikfagets problembehandlingskompetence. Her henviser jeg bl.a. til den anderkendte matematikdidaktiker Pernille Pind (Pind, 2010), der beskriver 9 sikre strategier til problemløsning:

Reducer problemet, Optrævling, Luk åbne problemer, Opstil og løs ligninger, Gæt og prøv efter, Vær konkret, Brug logik, Udtøm alle muligheder, Et skridt ad gangen

Jeg mener, der er helt klare sammenhæng og overlap mellem Pinds 9 sikre strategier og CT strategierne. Det vil ikke blive uddybet i dette speciale, men det har været vigtigt at få formidlet

denne sammenhæng til de implicerede lærere med henblik på at de kunne se CT og programmering som meningsfuldt i matematikfaget.

8 DESIGN BASED RESEARCH (DBR)

Dette projekt tager udgangspunkt i Design Based Research (herefter DBR) (Christensen, Gynther, & Petersen, 2012) som den metodologiske ramme viden genereres inden for.

DBR søger gennem designinterventioner i praksis på samme tid både at *forstå* og *udvikle* praksis. Forskningsspørgsmålet handler både om at udvikle et didaktisk design, der skal facilitere lærere i at integrere programmering i matematikundervisningen, samtidig med, at forskningsspørgsmålet også ønsker, at forstå de processer, der udfolder sig når eleverne programmerer. Denne dobbelthed i forhold til på samme tid at *forstå* og *udvikle* praksis, er en grundlæggende antagelse i DBR.

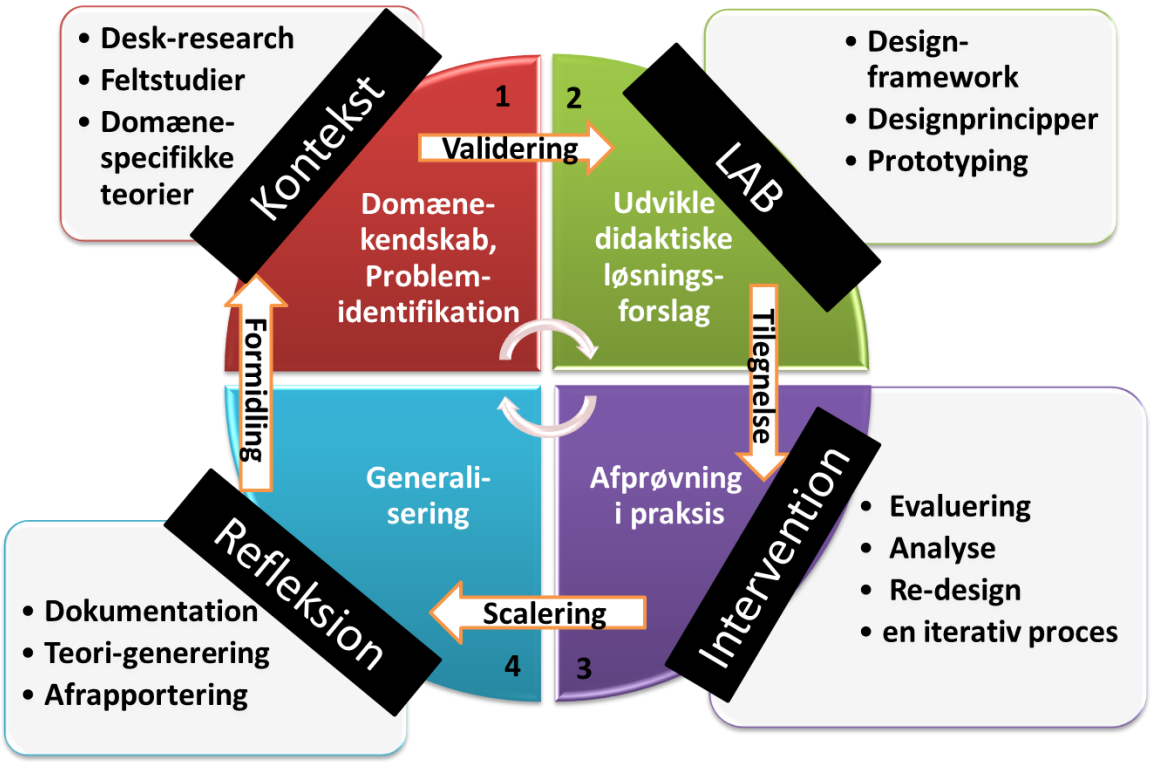
Denne dobbelthed drives i DBR af designinterventioner. At intervenere i praksis betyder, at der udvikles nye didaktiske designs, der afprøves i praksis. For at forstå de processer der er på spil i undervisning med programmering, må man ændre ved praksis – og for at forbedre designs for programmering i undervisningen, man forstå hvad der sker i praksis. Det særlige ved DBR tilgangen er, at designintervention gennemløber flere iterationer fra prototype mod et mere "robust" design, der vil kunne anvendes ud over den lokale kontekst.

Når designinterventioner skal udvikles og forbedres, er det afgørende at forskeren besidder domænespecifik indsigt eller viden om den kontekst der ønskes forbedret. Derfor skal der i DBR etableres et tæt samarbejde med deltagere fra praksis således at forskeren kan co-designe med praktikere.

DBR er en pragmatisk forskningsmetodik, som har til formål at forbedre en helt konkret praksis. Der fokuseres direkte på en helt konkret udfordring eller et problem i praksis. Gennem eksperimenter og analyser, prøver man at komme med et sandsynligt bud på forbedringer, der kan anvendes i lignende kontekster. DBR tager i denne sammenhæng udgangspunkt i eksisterende domæneteori – i dette speciales kontekst læringsteorier – samt teorier om design frameworks og design metodologier, for at øge chancen for, at det udviklede design er effektivt og realiserbart.

I forhold til denne teoriorientering, er det helt afgørende at være opmærksom på, at formålet med DBR ikke kun er at udvikle et løsningsforslag på en konkret udfordring eller problem på baggrund af eksisterende teori. Formålet med et DBR projekt er i høj grad at *udvikle* teori på baggrund af de designeksperimenter der udføres og på baggrund af eksisterende teori. Formålet er, at udvikle domænespecifikke læringsteorier og designteorier, der er anvendelsesorienterede direkte mod det problem eller den udfordringer praktikere oplever i en helt bestemt kontekst.

Et DBR forskningsprojekt gennemløber 4 faser hvilke beskrives herunder med inspiration fra DBR innovationsmodellen (Christensen, Gynther, & Petersen, 2012, s. 7).



8.1 Fase 1: Kontekst - Domænekendskab og problemidentifikation

Forskeren skal i denne fase have viden om det domæne, hvori genstandsfeltet befinder sig. Her handler dem om at have viden og indsigt i rammerne for folkeskolens undervisning, for matematikfaget, for skolens kultur, lærere og elevers tidligere erfaringer med programmering, specifikke forhold omkring elevgruppen osv. Det er også i denne fase der skabes et forskningssamarbejde mellem praktikere og forsker, for at kunne identificere oplevede udfordringer og problemer. I forhold til denne fase i DBR, skal det bemærkes, at jeg er kombinationsansat på den skole hvor projektet udspiller sig og har både erfaring og indsigt i både domænet og forskningen genstandsfelt. Dette forhold vil blive udfoldet i afsnittet "Forskerposition og forestillet læringsvej.

Dette speciales teoriafsnit og reviews er et udtryk for anvendelse af domænespecifikke teorier. Herudover har jeg interviewet 2 matematiklærere for at kunne foretage problemidentifikation.

En særlig udfordring i denne fase var, at de interviewede lærere ikke havde nogen erfaring med visuel programmering i matematik og kun meget overfladisk erfaring med programmering i det hele taget. Derfor var det vanskeligt at basere DBR i et oplevet problem i praksis, for der eksisterer helt banalt ingen praksis med programmering i matematik på skolen. Problemstillingen funderes derfor i lærernes formodninger om hvad de *tror* vil være en udfordring ved at programmere med eleverne i matematik og hvorfor de afholder sig fra det. Derfor støtter interventionsdesignet sig i høj grad op ad designprincipper (se fase 2) i form af CAS Barefoots tilgange til programmering (se afsnit 4).

Afsluttende kan man herudover sige, at min egen ansættelse og deltagelse i skolens praksis er en form overordnet og udokumenteret feltstudie.

8.2 Fase 2: LAB - Udvikle didaktiske løsningsforslag

Med afsæt i domænespecifikke teorier og interview med praktikere, er der udviklet en første intervention. Designinterventionen kombinerer domænespecifikke teorier med didaktisk designframework og didaktiske modeller for designprocesser (Afsnit 12.3). Herudover, har CAS Barefoots 5 didaktiske principper været styrende designprincipper (Afsnit 7.2).

8.3 FASE 3: INTERVENTION - AFPRØVNING , EVALUERING, ANALYSE OG ITERATION

Intentionen i denne fase er, at designet afprøves, evalueres, analyseres og redesignes iterativt frem mod et design, der rummer en vis robusthed og legitimitet. Processen forløber fra skitse, til at være en delvis løsning og i bedste fald blive et fuldt fungerende læringskoncept for visuel programmering i matematik.

Dette forskningsprojekt præsenterer en overordnet evaluering af det udviklede didaktiske design, men der er ikke udviklet en ny iteration af designet der er justeret i forhold til disse anbefalinger endnu. Der er således heller ikke gennemført flere iterationer i praksis .

8.4 FASE 4: REFLEKSION – GENERALISERING

Hvor fase 3 er præget af formative evalueringer i processen frem mod udvikling af et robust design, handler Fase 4 handler om, at foretage en summativ evaluering af designet og den teori det har genereret. Fase 4 handler om, at afgøre hvor robust designet er i forhold til *forskellige* typer af kontekster.

Da dette forskningsprojekt stopper efter 1. iteration og flere iterationer er anbefalet på baggrund af undersøgelsens konklusion, vil der ikke blive evalueret summativt.

8.5 FORSKERPOSITION

Som nævnt i under gennemgangen af DBR fase 2 om domænespecifik viden, er det et væsentlig forhold, at jeg underviser i matematik på samme skole som designinterventionen udfoldes i. Dette kan ses som en styrke og som en svaghed. For det første har jeg dybere indsigt – måske i form at "tavs viden" – om den lokale skoles forhold, kultur og miljø. Da det er mine kollegers praksis jeg forsker i, har det også betydning for den tætte samarbejdsrelation der er påkrævet i DBR. Selv om dette er en styrke, er det også en overhængende risiko for, at jeg er blind og forudindtaget i forhold til kontekstens betydning og at min forskning påvirkes af kollegiale magtstrukturer og lokale kulturelle bindinger.

Jeg har ud over dette domænekendskab endvidere erfaring og indsigt i forskningens genstandsfelt: visuel programmering i pædagogiske kontekster. Dette betyder alt andet lige, at jeg har nogle personlige erfaringsbaserede antagelser om hvilke didaktiske udfordringer og læringsmæssige potentialer visuel programmering i matematik indeholder. Dette udfordrer min forskerrolle som fallibist og vil have betydning for alle faser i DBR – og måske især på analysen af interventionens resultater.

Her har jeg fundet inspiration i begrebet "forestillet læringsvej" (Misfeldt, 2010). Begrebet forestillet læringsvej, er et begreb der knytter an til DBR som metodologi. Det er forskerens forestilling om, hvordan den designintervention der er udviklet, vil blive brugt i praksis og hvilke erfaringer brugerne vil gøre.

Den forestillede læringsvej skal ifølge Misfeldt altid funderes i teori. Jeg taler for, at den forestillede læringsvej, altid i større eller mindre grad også er funderet forskerens personlige erfaringer og

holdninger, som ikke kan holdes udenfor. I stedet for at fortrænge disse mere subjektive kim til den forestillede læringsvej, er det derimod vigtigt, at de bliver ekspliciteret. Et krav til forskerens erfarings- og holdningsbaserede input til den forestillede læringsvej, må være, at de holdes op i mod det teoretiske fundament DBR hviler på. Det er ikke "anything goes". Kan forskerens erfaringsviden forklares gennem teorien, er disse gyldige input til den forestillede læringsvej og ellers må de holdes ude af forskningsprojektet. Pointen er, at dette netop kan kun lade sig gøre, hvis de er bevidstgjorte og ekspliciterede. Dette har jeg været opmærksom på i dette speciale.

9 VIDENSKABSTEORETISK TILGANG

Dette speciale har pragmatikken som tydeligt og bevidst dominerende videnskabsteoretisk position. Der også trukket konstruktiv interpretivistisk videnskabsteori ind som sekundær teori, med det formål berige videnskabelsen med fortolkningsviden på baggrund af kvalitative interviews. Pragmatikkens forsker er engageret i at ændring, interpretivismens forsker er engageret i at forstå (Goldkuhl, 2012). Sammentænkning af primær og sekundær videnskabsteori er gjort med inspiration i Goldkuhl, og formålet er at berige den pragmatiske videnskabelse kvalitativt.

9.1 PRAGMATISME

Ifølge Dewey (Brinkmann, 2006) er viden ikke noget der erkendes gennem passiv beskuen. Mennesket "vender udad" (Brinkmann, 2006, s. 33) hvilket betyder, at det godt kan være at sandheden findes derude, men at den erkendes subjektivt gennem aktiv handlen. Den problemløsning som vores erkendelse er vævet ind i forsøget på at begribe verden er grundlæggende social. Pragmatismens udgangspunkt er, at viden om verden er "sand" så længe fællesskabet har gavn af den i praksis.

Pragmatismen er ikke interesseret i at finde den endegyldige "sandhed", men mere optaget af hvad der mest effektivt kan sige os noget sandt om den konkrete situation man befinder sig. (Egholm, 2014) Dermed er det antydet, at erkendelse og viden i høj grad både er situativ (bestemt af konteksten), relationel (social) og processuel (Egholm, 2014, s. 173). Pragmatismen er derfor også ideografisk, hvilket betyder at man er interesseret i den viden der skabes på baggrund af enkelttilfælde.

Viden skabes i pragmatismen gennem "abduktion". Abduktion forener de induktive og deduktive principper. Man forholder sig både til det deduktive og erkender at viden og teorier aldrig kan være "værdifri", netop fordi det er opstået på baggrund af handlinger, der har vist sig værdifulde for fællesskabet. På den måde skaber teorier en forståelseshorisont for de handlinger der udspiller sig. Samtidig vægter man i særlig grad de induktive slutninger, hvor handlinger og praksis skaber fundament for hypotesedannelse og teoriproduktion. Abduktion handler om, at man ikke anvender en enstrenget logik i forhold til deduktive slutninger (dvs. fra praksis til teori), men at man går eksperimentelt og "detektivisk til værks" på et mikroniveau og har blik for enkelttilfældene. I produktionen af viden opstiller man mange derfor mange dristige, kvalificerede gæt og hypoteser i forsøget på at forklare handlinger og deres konsekvenser. En vigtig pointe inden for pragmatismen er yderligere, at forskeren skal være fallibist – dvs. besidde evnen til at accepterer det uforklarlige og de "anomalier" der kan udfordre forskerens gældende hypoteser, teorier og forståelser.

Hvis vi perspektiverer pragmatismen til DBR som metodologi, må man konstatere, at der er helt tydelige paralleller i den måde videnskabelse betragtes på. DBR søger at skabe viden ud fra et pragmatisk perspektiv. DBR metodologien tager netop udgangspunkt i enkelttilfældet og vægter

iterationer af "hypotetiske" designs frem mod en sandsynlig viden og teoriskabelse, men vedkender sig at et udviklet design kun er "sand" så længe det er til gavn for fællesskabet – hvilket vil sige i andre kontekster end den lokale, hvori den er studeret. Disse iterationer er bl.a. et udtryk for pragmatismens processuelle forståelse af viden.

9.2 SOCIALKONSTRUKTIVISME

Inden for det interpretivistiske paradigme, er det sekundære videnskabsteoretiske udgangspunkt socialkonstruktivismen (Brinkkjær & Høyen, 2011). Inden for denne position, er viden konstrueret af mennesker og deres sprogbrug. Meninger og erkendelser konstrueres ud fra sociale kontekster og i interaktion med andre mennesker. Mennesket erfarer ikke direkte, men konstruerer deres forståelser gennem de sammenhænge de deltager i. Virkeligheden er dermed afhængig af den sociale kontekst i bredeste forstand og viden er derfor både situativt, kulturelt og socialt betinget. Viden skabes ved at undersøge tilblivelsen at sociale fænomener, og hvordan der tales om det og henvises til disse fænomener. Sandhed kan afprøves i dialogiske samtaler, men da viden er subjektiv, er det et spørgsmål om subjektiv fortolkning.

10 METODE

10.1 INTERVIEW SOM METODE

I DBR fase 2, handler det om at analysere, hvordan designinterventionen udspillede sig i praksis og på den baggrund udvikle et redesign til næste iteration.

I forhold til specialets sekundære videnskabsteori, er der i specialet valgt interviewet som kvalitativ forskningsmetode for at kunne indfange den viden, der er indlejret i konteksten. Jeg har i denne sammenhæng valgt en metodetilgang, der fokuserer på en interesse i at forstå sociale fænomener ud fra hvordan de italessættes gennem dialog med fortolkningen som omdrejningspunkt (Goldkuhl, 2012).

Kvale og Brinkmann skelner mellem intervieweren som minearbejder eller som rejsende (Kvale & Brinkmann, 2009, s. 67). Minearbejderen prøver at bevare informantens viden som "intakt" og ubesmittet af interviewerens spørgsmål. Her ses viden, som rene klumper der venter på at blive gravet frem og i den forbindelse er det er væsentligt, at interviewerens spørgsmål er neutrale og ikke ledende.

Jeg forsøger, at anlægge den rejsendes perspektiv, hvilket betyder jeg, at jeg indtager en mere aktiv rolle, der spørger udforskende ind til informanternes livsverden. Dette har væsentlig betydning for både den måde interviewene er struktureret på og for hvordan den empiriske analyse er foretaget. Begge perspektiver uddybes nedenfor. Jeg betragter således ikke informanterne som passive videnskar, men som aktive rejsefæller, hvor interviewet er som en social kontekst hvori viden skabes gennem dialogen. Dette syn på videnskabelse genfindes i den socialkonstruktivistiske videnskabsteori.

10.1.1 Semistrukturerede interviews

Intervieweren som minearbejder, ser interviewet som et dataindsamlingssted fuldstændigt isoleret fra analysen. Metaforen om intervieweren som "rejsende" fører derimod til, at interview og analyse ses som sammenvævede processer. I kvalitative forskningsinterviews begynder en del af analyseprocessen med at forstå informanten allerede i selve interviewet (Brinkmann & Tangaard,

2015). Fordelen ved et semistruktureret interview, er at have friheden til at kunne forfølge, fortolke og analysere de narrativer der udfoldes samtidig med, at forskeren holder fast i interviewets overordnede temaer og intentioner. Målet er at komme så tæt som muligt på *"interviewpersonens oplevelser og i sidste ende, at formulere et kohærent og teoretisk velinformeret tredjepersons perspektiv på oplevelsen."* (Brinkmann & Tangaard, 2015, s. 31)

For at holde fast i de temaer jeg gerne ville have viden om, har jeg udarbejdet flere interviewguides (**Fejl! Henvisningskilde ikke fundet.**):

- Indledende interviewguide til de 2 lærere inden interventionsdesignet blev udarbejdet.
- Interviewguide til eleverne
- Efterfølgende interviewguide til lærerne.

Der er i alt interviewet 13 personer: 6 elever fra 5.klassen, 5 elever fra 7.klassen samt klassernes 2 matematiklærere.

Da eleverne, som konsekvens af designinterventionen, skulle arbejde i par, valgte jeg parvise interviews. Dette valg er truffet, fordi flere af interviewtemaerne inddrager klassefælleskabet og elevernes indbyrdes dialog. Derfor er det væsentligt at få begge elevers fortællinger og perspektiver på netop dette centrale tema. Eleverne kan endvidere under interviewet samtale og forhandle deres forståelse indbyrdes og sammen med interviewerен nå frem til flere nuancer og perspektiver. Denne fælles videnkonstruktion underbygges af specialets sekundære socialkonstruktivistiske videnskabsteori.

De interviewguides, der var målrettet eleverne, indeholdt endvidere henvisninger til bestemte scratchprogrammer, som eleverne havde programmeret. Disse konkrete artefakter kunne eleverne støtte sig til og forklare ud fra, når bestemte (og vanskelige) temaer skulle forklares.

10.1.2 Refleksioner over interviewets kvalitet

I forhold til kvaliteten i de gennemførte interviews støtter jeg mig op ad Kvale og Brinkmanns kvalitetskriterier for interviews (Kvale & Brinkmann, 2009, s. 186).

En indikator for kvaliteten afgøres bl.a. af graden af *"spontane, righoldige, specifikke og relevante svar"* og *"graden af korte interviewspørgsmål og længere svar fra interviewpersonens side"*. Især i forhold til at beskrive deres egen tænkning i forbindelse med problemløsning, var det en udfordring at få eleverne til at fortælle på en måde, hvor svarene var specifikke og faldt inden for analysetemaerne. Størstedelen af eleverne svarede her ret kortfattet og jeg var flere gange været nødt til at omformulere spørgsmålene eller uddybe dem med eksempler. Enkelte af disse "udfoldelser" af spørgsmål, samt nogle af de mere improviserede og afklarende spørgsmål, har haft karakter af ledende spørgsmål. Ledende spørgsmål er en legitim del af et kvalitativt interview. Nogle af dem har været uheldigt anvendt og har formet elevernes svar. Andre har haft en kvalitet i interviewet, fordi de har valideret eller kontrolleret tidligere svar. Her et eksempel fra transskriptionen der viser uhensigtsmæssig brug af ledende spørgsmål:

I: Når I bytter rundt på klodserne.. hvordan gør man så? Eksperimenterer man?

Her er der overhængende risiko for, at få svar der bekræfter, at de eksperimenterer. Det var ikke sikkert, det var det, de ville have fortalt.

Det fremgår desuden af transskriptionerne, at jeg var "begejstret" for bestemte svartyper. F.eks. anvendte kommentarer som "super", "godt", "fint" og "spændende", når eleverne beskrev noget

præcist eller spændende om interviewtemaerne. Dette kan have betydet, at eleverne har luret hvad læreren (intervieweren) gerne ville høre og at det har påvirket deres svar.

Eleverne svar er betydeligt mere righoldige og relevante, når samtalen var knyttet til de artefakter i form af scratchprogrammer eleverne havde programmeret, og mine uddybende spørgsmål var her også mere konkrete og specifikke. At inddrage artefakter i interviewene har været en kvalitet.

I forhold til det at interviewe i par, viste det sig, at eleverne i nogen grad fortalte om deres erfaringer ved at "bygge videre på hinandens" fortællinger og på den måde skabte fælles fortællinger. Dette vurderes som en kvalitet. Dog var en af grupperne en 3-mands gruppe, og her var der særdeles sparsom deltagelse fra en af elevernes side. For at høre denne elevs stemme, ville det uden tvivl have været mere givende, at have foretaget et enkeltperson interview eller måske benyttet en helt anden kvalitativ metode. Dette vidner om de udfordringer, der også kan være ved at interviewe flere på en gang i forhold til elevernes sociale relationer og elevernes opfattelse af sig selv i sociale sammenhænge.

For mig kan interviewdelen sammenlignes med et designprojekt. Ved projektstart er ens viden mest begrænset og de beslutninger man træffer vejer tungest. Hen imod projektets afslutning er ens viden om genstandsfeltet stor - empirien er indsamlet og analyseret - men det er vanskeligt at foretage radikale ændringer i metoden. Her har DBR modellens iterative blik for udvikling af designmetodikker en stor styrke. I en fremtidig iteration, ville en anbefaling være, at inddrage flere kvalitative metoder, såsom observation eller videoobservation. Dette foreslås ud fra, at eleverne forståeligt havde vanskeligt ved at beskrive forhold vedrørende deres læring og tænkning. Interviewene har afgjort kvalitet i dette forskningsprojekts første DBR iteration, men empirien kunne beriges kvalitativt med observationer direkte i processen, hvor elevernes tænkning og læring kan udspiller sig.

I analysen af interviewene skinner ovenstående forhold igennem, da elevernes udsagn nogle steder, er blevet analyseret på mikroniveau – helt ned på formuleringer og ordvalg. Dette gør, at analysens konklusioner også er båret af mulige hypoteser. I den pragmatiske tilgang til viden, er der ikke noget i vejen med hypoteser, så længe man er tydelig omkring det. Iterationerne i DBR har til hensigt netop at generere både udviklings og forståelsesviden, der kan bekræfte, justere eller forkaste disse hypoteser. Pointen er her, at andre supplerende kvalitative metoder, givet ville kunne producere højere grad af evidens bag hypoteserne.

I forhold til interview af eleverne, er det også af betydning, at de ved, at jeg er lærer på deres skole. Eleverne er alt andet lige præget af en skolelogik, hvor de udmærket er klar over, at det at opføre sig ordentligt og det at gøre sig umage – måske endda det at kunne det rigtige svar - opfattes af de fleste lærere som meget værdifuldt. Eleverne ved, at jeg er involveret i det forløb, de har arbejdet med og der er en risiko for, at deres svar kan bære præg af, at de ønsker at fremstå som dygtige og arbejdsomme. Eleverne ved også godt, at jeg kollegialt er tæt forbundet med deres egen matematikunderviser, og deres svar kan også bære en intention om, ikke at ville skuffe deres underviser eller en bekymring for, om jeg "går videre" med de ting de fortæller og derfor udelader visse detaljer.

10.2 Analysens faser og aktivieter

I forhold til at planlægge, gennemføre og analysere interviewene, har jeg delt aktiviteterne op i 3 faser:

- Forberedelse

- Interviews
- Analyse og fortolkning

10.2.1 Forberedelse

Som en del af forberedelsen af de korte lærerinterviews blev der afholdt et indledende møde. Her blev de 2 lærere præsenteret for, hvad specialet drejede sig om, og rammerne for interviewet. Der blev ikke gjort andet i forberedelsen til dette interview som f.eks. at de skulle have afprøvet Scratch. Dette for at tage udgangspunkt i den eksisterende praksis uafhængigt af designinterventionen.

En væsentlig del af forberedelsen af elevinterviewene var at udvælge elever. Dette har stor betydning for empiriens karakter og kvalitet. Det ville have været optimalt, hvis alle eleverne blev interviewet. Dette har, grundet tidsmæssige faktorer, ikke været muligt i dette forskningsprojekt.

Pragmatismen interesserer sig for viden skabt på baggrund af enkelttilfælde i en bestemt kontekst, og derfor kan få interviewpersoners handlinger godt være værdifulde som bidrag til velbegrundede hypoteser. Da empirien dannes på baggrund af et begrænset antal informanter, er det væsentligt at sikre den empiriske kvalitet. Derfor er der i blevet udviklet nogle kriterier for, hvilke elever der skulle udvælges som interviewpersoner.

1. Elevernes niveau eller kvaliteten i deres produktion måtte *ikke* være et kriterie.
2. Eleverne skulle have deltaget med en tilfredsstillende grad af arbejdsindsats og seriøsitet.
3. Eleverne skulle have færdiggjort deres designproces og have et færdigt produkt.
4. Eleverne skulle vurderes gode til at fortælle og reflektere over deres arbejdsproces.

I samarbejde med underviserne blev 2-3 elevpar fra hver klasse valgt. Kriterie 1, der angår elevernes niveau, har især været vigtigt i forhold til at øge sandsynligheden for, at det udviklede design kunne anvendes i andre lignende kontekster. Dette ville ikke være tilfældet, hvis der blev tegnet et glansbillede. Kriteriene 2 – 4 blev især valgt, fordi de ses som en forudsætning for, at kunne generere empiri af en vis kvalitet.

10.2.2 Indsamling af empiri

Elevinterviewene blev gennemført i op til 3 dage efter forløbene var afsluttet i klasserne, hvor eleverne har forløbet i frisk erindring. Eleverne blev forud for interviewets start orienteret om interviewets formål, at kun deres fornavn ville fremgå i specialet og at de lydfiler der blev optaget ikke ville blive delt med andre. Dette kan have haft betydning for, om eleverne ellers ville have været tøvende med at sige deres ærlige mening eller i det hele taget have lyst til at deltage.

De afsluttende lærerinterviews blev gennemført efter elevinterviewene og inden for 1,5 uge efter forløbet var afsluttet i klasserne. Interviewene blev optaget som lydfiler og efterfølgende transskriberet.

10.2.3 Analyse

Analysen blev foretaget ud fra analysetilgangen ”thematic analysis” (herefter ”tematiske analyser). (Braun & Clarke, 2006). Der er overordnet to tilgange til tematiske analyser: en induktiv og en deduktiv. Den deduktive tilgang tager forskeren udgangspunkt i teoretiske begreber, og koder det empiriske materiale med teorien som linse. Jeg har valgt den induktive tilgang, hvor jeg har prøvet at kode empirien med en åben tilgang. Jeg har bevidst søgt at lægge ”teorierne på hylden” og på den baggrund har jeg identificeret temaer, der ligger fjernt fra det de forskningsspørgsmål der blev stillet og de svar der blev givet. Jeg er bevidst om, at forskerens teoretisk tankegods ikke alligevel har en vis betydning for identifikationen af temaer. Pointen er, at den induktive tilgang generelt producerer rigere og mere varierede beskrivelser, hvorimod den deduktive tilgang producerer mere detaljerede

og dybdegående beskrivelser (Braun & Clarke, 2006, s. 84). Jeg har valgt den induktive tilgang, fordi jeg oprigtigt var nysgerrig på datamaterialet og endnu ikke havde den teori, der kunne beskrive de antagelser jeg f.eks. havde om et visuelt programmeringssprogs betydning for tænkningen. Samtidig understøtter den induktive bottom-up tilgang et pragmatisk syn på videnskabelse.

En væsentlig sondring i en tematisk analyse er hvorvidt temaerne bliver identificeret på baggrund af kodning af empiriens semantik – altså helt ordret hvad der blev sagt og intet mere – eller man har kodet efter latente temaer gennem fortolkning af semantikken (Braun & Clarke, 2006, s. 84). Jeg har anvendt den sidste tilgang, da jeg som tidligere nævnt, har haft behov for at fortolke semantikken på et mikroniveau i forhold til elevernes tænkning med visuelle programmeringssprog.

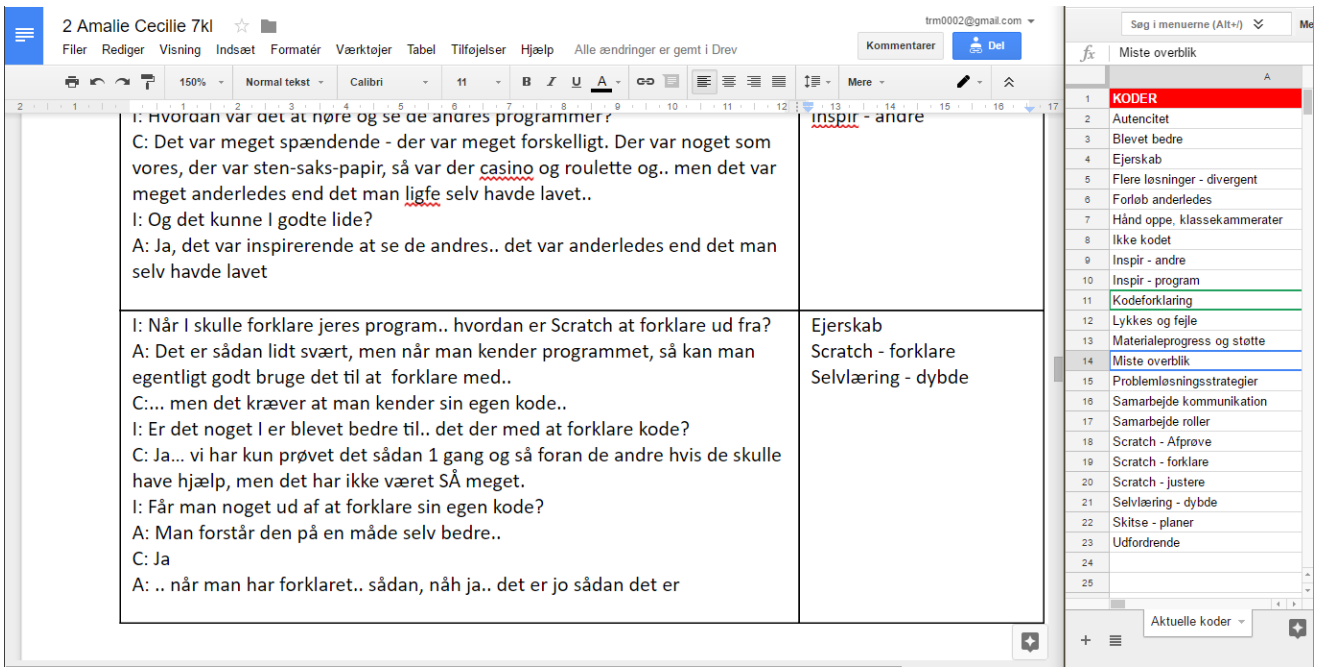
En tematisk analyse forløber over 6 faser (Braun & Clarke, 2006, s. 87). Grundet den måde jeg digitalt har arbejdet med analysen, var fase 4 og 5 en og samme proces. Dette fremgår nedenfor. Faserne har jeg oversat til dansk.

10.2.3.1 Fase1: Kendskab til data

Denne fase er fundamental for resten af faserne. Det er i denne fase, der skabes overblik over det empiriske materiale ved at dykke ned i den og lære den at kende. For mit vedkommende bestod denne fase i, at transskribere samtlige interviews. Dette arbejde førte til identifikation af de første mønstre. Herefter gennemlæste jeg transskriptionerne igen og begyndte at notere stikord og de første koder.

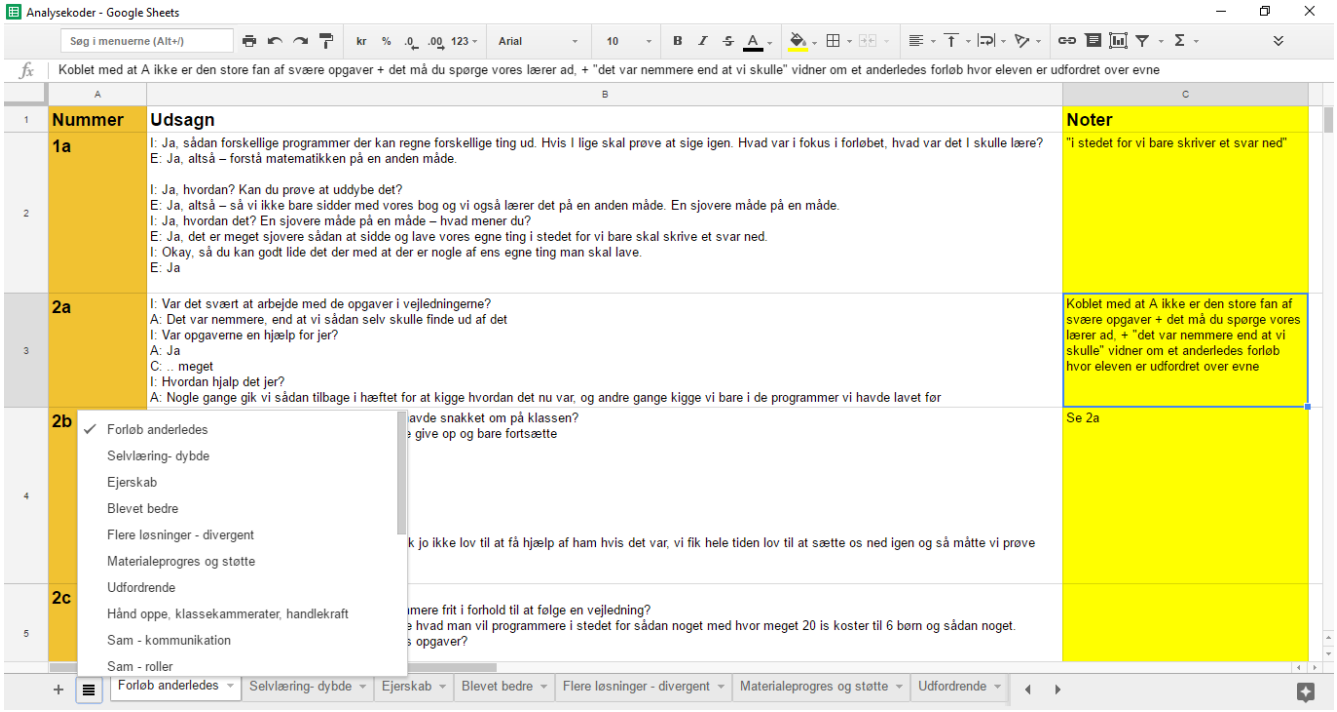
10.2.3.2 Fase2: Indledende kodning

I denne fase kodes interessante træk ind i hele datamaterialet. Dette arbejde foregik systematisk, og alle anvendte koder blev løbende samlet og ajourført i et regneark (Figur 3).



Figur 3:Kodning af interview transskription.

Af Figur 3 fremgår det, at transskriptionerne var inddelt i "bidder", der hver især er kodet. De samtalebidder der bar de samme koder, blev efterfølgende samlet i et stort regneark (Figur 4)

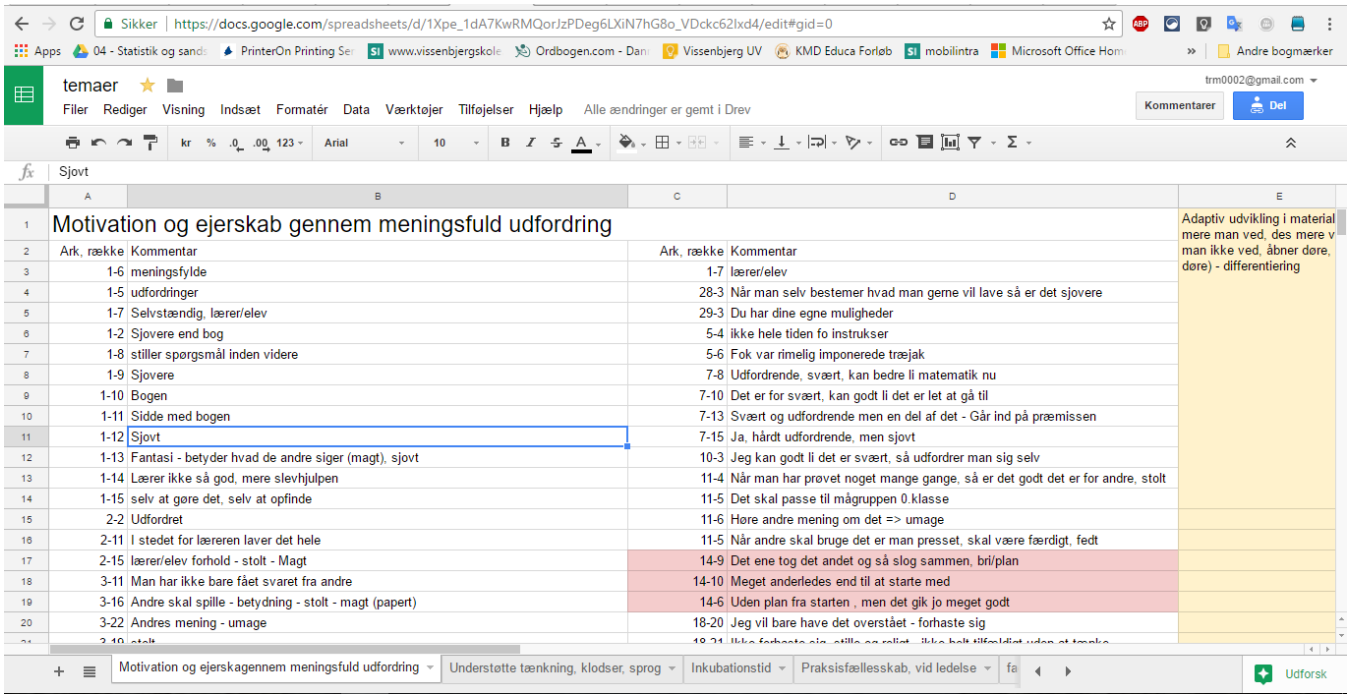


Figur 4: Samling af koder og tekst

Figur 4 viser et screenshot af det regneark, hvori alle tekstbidder blev samlet. I bunden af regnearket ses koderne som "ark" i regnearket, ovenfor er data samlet for hver kode. I kolonnen til venstre (Nummer), holdes der styr på hvilke elevgrupper hver databid tilhørte og til højre i regnearket, blev der i kolonnen "Noter" skrevet uddybende noter, ideer og fortolkninger.

10.2.3.3 Fase3: Temasøgning

Her blev koderne samlet i potentielle temaer. Temaer kan forstås som bredere begreber, der rummer de enkelte koder. Her handlede det om, at kunne se mønstre mellem koderne og bruge dette til at skabe potentielle temaer.



Figur 5: Skabelse af temaer

I Figur 5 ses de potentielle temaer i bunden af regnearket som ark. Ovenfor er henvisninger til ark og rækker regnearket fra fase 2 (Figur 4), hvor de kodede databidder kan genfindes. Ud for hver henvisning er der en (meget subjektiv) kommentar til mig selv, så jeg hurtigt kan erindre hvad henvisningerne refererer til. Dette kan kun lade sig gøre, fordi jeg på dette tidspunkt i analysen efterhånden kender datamaterialet indgående.

10.2.3.4 Fase4: Gennemgå de potentielle temaer, Fase5: Navngiv og definer temaerne

Disse to faser foregik som en samlet proces, og det var vanskeligt at skille dem ad. Jeg oplevede, at dette havde at gøre med den digitale strukturering af temaerne, hvor redigering og flytning af temaer og datahenvisninger forløb som en samlet proces. I denne del af analysen, gennemgik jeg temaerne i forhold til de kodede databidder og i forhold til hele datamaterialet. Flere temaer blev omdøbt, slået sammen eller slettet for bagefter at oprette nye med det formål at skabe præcise temaer, der så præcist som muligt indfangede essensen af, hvad der var på spil i hvert tema. De endelige temaer kan genfindes i specialets 3 analysetemaer (afsnit 13).

10.2.3.5 Fase 6: Produktion af forskningsrapport.

Denne fase bød på den sidste mulighed for analyse af empirien. Udvælgelsen og formidling af de bedst egnede ekstrakteksempler fra datamaterialet medførte en endelig analyse, der relaterer tilbage til forskningsspørgsmålet og teorien.

11 LÆRINGSTEORI

I dette afsnit vil jeg primært præsenterer to teoretiske positioner. Det ene er konstruktionismen, som en særlig konstruktivistisk position repræsenteret ved Seymour Papert. Papert er en helt fundamental nøgle, til at blive opmærksom på de processer, der er i spil, når elever skaber og lærer ved at programmerer. Paperts fodaftryk er tydelige i hele specialet lige fra CAS Barefoots konceptualisering af CT strategierne, til valg af Scratch som visuelt programmeringssprog og valg af teori.

Da CAS Barefoots didaktiske principper indgår i udviklingen af designinterventionen, kan man betragte denne som et ønske om at genskabe Paperts observationer i praksis. Men en ting er, om forløbet i 5. og 7.klasse afspejler nogle af de beskrivelser og observationer der bliver fremsat af konstruktionismen, noget andet er at forstå hvad årsagen er til dette. Konstruktionismen er fattig på svar på dette område.

Derfor inddrages den sociokulturelle læringsteori. Vygotsky leverer teoretiske begreber, der kan komme nærmere en forklaring af de "observationer" Papert gjorde når elever programmerer. På den baggrund vil jeg udvinde virksomme teoretiske analysebegreber, der kan hjælpe med af besvare forskningsspørgsmålet.

11.1 KONSTRUKTIONISME

11.1.1 Et konstruktivistisk udgangspunkt

Seymour Papert, der er hovedmanden bag konstruktionismen, tog i særlig grad afsæt i hans tidligere kollega Jean Piagets konstruktivisme i udvikling af konstruktionismen (Ackermann, 2001). Konstruktionismen er derfor en konstruktivistisk position inden for det kognitive erkendelsesparadigme med Piaget som stærk inspirator. For at forstå konstruktionismens forhold til kognition og læring er det først væsentligt kort at præsentere, hvad Papert og Piaget var helt enige om:

I den konstruktivistiske læringsteori ses læring som en kognitiv progression fra enkle til komplekse mentale modeller. Det er den lærende selv, der aktivt konstruerer forståelse og viden med udgangspunkt i, hvad den lærende ved i forvejen. Individet vil altid søge, at forstå nye erfaringer ved at erkende og forstå dem på baggrund af eksisterende kognitive "skemaer". Denne proces kaldes assimilation og skemaer er en begrebsmæssigt konstruktion, der forklarer hvordan aktiviteter fæstner sig som repræsentationer i hjernen. Hvis fænomener ikke kan forstås på baggrund af eksisterende skemaer, må nye skemaer bygges eller eksisterende omstruktureres. Dette kaldes akkomodation eller akkomodativ læring. Viden kan således ikke overføres 1:1, men er et aktivt subjektivt konstruktionsarbejde.

11.1.2 "N" som i konstruktionisme

Papert definerer selv konstruktionismen således:

" *Constructionism--the N word as opposed to the V word--shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe"* (Papert & Harel, 1991)

Papert lægger vægt på, at den konstruktivistiske læreproces har de bedste vilkår, når den lærende ikke blot eksperimenterer, manipulerer eller undersøger konkrete fænomener og genstande, men når den lærende er *bevidst fordybet* i at *skabe* artefakter der skal *anvendes eller fremvises for andre*. Altså konstruerer artefakter for en offentlighed.

11.1.3 Konkret og formel tænkning

Papert ekspliciterer selv, er han enig i Piagets begreber om kognitive skemaer og strukturer (Papert, 1983 [1980]) og han er også enig med Piaget i, at børn gennemløber forskellige kognitive stadier i deres udvikling. Piaget er kendt for sin stadieteori, der helt kort kan skitseres således (Hermansen, 2005):

- Den sensomotoriske periode - (0-2 år)
- Den præoperationelle periode (2-7 år)
- Den konkret operationelle periode- (7-11 år)
- Den formelt operative periode (fra 11 år)

Det er ikke relevant, at gå nærmere ind i beskrivelserne af faserne i dette speciale. Her vil jeg dog pointere, at erkendelsen ifølge Piaget udvikler sig fra udelukkende motorisk erkendelse til at erkendelse kan ske løsrevet fra en konkret kontekst. I fase 3 kan børn begynde at tænke logisk og reversibelt dog fortsat knyttet til en konkret, fysisk kontekst, hvorimod den afsluttende fase 4, præges af abstrakt kombinations- og "hypotetisk deduktiv" inde-i-hovedet tænkning.

Der hvor Papert primært er uenig med Piaget, er i de 2 sidste faser (Papert, 1983 [1980]). Når Piaget forklarer, at komplekse begrebers langsommere udvikling hos børn skyldes, at begrebet befinder sig på det formelle plan og at barnets kognition derfor ikke er modnet til at løsrive tænkningen fra det konkrete plan, er Papert ikke enig. Papert kritiserer Piaget for ikke i tilstrækkelig grad, at have øje for redskaber og mediers betydning for erkendelsen.

Dette omtaler Paperts som "infrastruktur. Den infrastruktur, som Papert henviser til, er en digital infrastruktur med computeren som "materialet" eller mediet. Dette medie giver mulighed for, at forbinde konkret og formel tænkning på måder, der overskrider Piagets faseteori.

"With technology we have the infrastructure to enable us to realize dreams like Leonardo da Vinci's. At last, Dewey's idea that you can learn everything through curiosity and passion for a particular subject area can be fulfilled" (Papert, 1998)

I konstruktionismen er et skridt på vejen mod at forstå hvordan børn lærer, at se dem som håndværkere. Håndværkere har brug for materialer, for at kunne udrette noget, på sammen måde som børn har brug for konkret omgang med materialer, for at kunne "konstruere" deres egne forståelser. Når børn er sene til at lære mere abstrakte begreber, så skyldes det i mindre grad barnets kognitive modning, men at barnet ikke har de *materialer* til rådighed der gør, at de med støtte fra disse materialer kan erkende på et formelt plan.

Computere har den egenskab, at de kan simulere alle andre medier, og når eleverne kan programmere en computer til at gøre det *de* ønsker den skal gøre, repræsentere eller modellere, åbner det op for, at et væld af abstrakte fænomener kan simuleres helt konkret gennem elevskabte computerprogrammer. Computere giver børn mulighed for at skabe, manipulere, eksperimentere og gøre frugtbare erfaringer med abstrakte begreber, som det simpelthen ikke er muligt at gøre i samme omfang i den virkelige verden. Endvidere har den lærende med computeren mulighed for selv, at konstruerer de konkrete artefakter og værktøjer, der bedst understøtter den lærende i, at gøre erfaringer med fænomenet.

11.1.4 Akkomodativ læring

Ifølge Ackermann (Kafai & Resnick, 1996, s. 27) er en del af nøglen til at forstå konstruktionsmen i dens stærke interesse for akkomodative læreprocesser. Papert er optaget læringens dynamik især i akkomodativt orienterede læringssituationer, hvor man aktivt er optaget af at forstå nye og ukendte fænomener. Ifølge Papert er det at være intelligent således i høj grad at kunne være "in-situ" – dvs opslugt af situationen, forbundet og opmærksom på ændringer i omverdenen. I den konstruktionistiske forståelse er det at "blive et" eller blive "forbundet" med det fænomen man studerer nøglen til læring. Og det gør man bedst ved selv at skabe noget i processen.

Når elever selv skaber og eksperimentere sammen med andre og er "opslugt" i læreprocessen, siger det næsten sig selv at det har indflydelse på elevernes engagement og motivation. Men for Papert, drejer det sig ikke bare om motivation og engagement som et mål i sig selv, men om, at eleverne indgår i et ny forhold til det der læres. Når et barn lærer ved at skabe og læreprocessen er karakteriseret ved meningsfuld "forbundethed" ændres hele læreprocessen. Kundskaberne erhverves med et personligt mål for øje og barnet bliver mere aktivt og selvstyrende i læreprocessen. Forholdet til viden ændres, så barnet indser, at selv komplekse konstruktioner og begreber er skabt af en person, og med den rette læring og de rette erfaringer, kunne den person være barnet selv. Børn oplever at viden og læring er en kilde til magt som de selv ejer og kontrollerer (Papert, 1983 [1980], s. 25; Barnes, 2017, s. 18-21).

11.1.5 At tænke som en computer

Papert så et helt særligt potentiale i, hvordan særlige pædagogisk udviklede programmeringssprog kan understøtte elevernes tænkning. Ved at programmere lærer eleverne noget om hvordan de selv lærer. Det sker ifølge Papert, fordi programmering pr. definition *er* problemfyldt og fuld af fejl. Når eleverne arbejder med meningsfulde aktiviteter kendetegnet ved mange veje til en løsning og der ikke er en rigtig måde at tænke og lære på, opdager eleverne, at det at lære ikke handler om at kunne levere *det* rigtige svar, men at det at lære også handler om, at lave masser af fejl. En stor del af at programmere er at "debugge" altså finde og rette fejl, og mens man gør det er man engageret i at problemløse. Herigennem lærer eleverne at løse problemer og udfordringer og at sætte ord på deres egen læring og tænkning.

Når Papert taler om, at børnene ved at udvikle computerprogrammer lærer at tænke som en computer (Papert, 1983 [1980], s. 31), er der ikke langt fra Wings forståelse af CT strategier og betoningen af "at tænke som en computer scientist". Eleverne oplever meget konkret, at computeren kun kan gøre det de beder den om og at computeren kun kan udføre elevernes kommandoer sekventielt – altså fra toppen og ned. Det betyder, at de data eleverne skal "fodre" computeren med, skal tænkes på en helt bestemt systematisk måde. Dette er netop en af hovedpointerne i CT at kunne tænke og levere løsninger og data på en måde, så computere kan berige dem eller gøre noget med dem. Denne fornemmelse for "viden i samspil med computere" udvikles helt konkret gennem programmering. Det at tænke som en computer, kalder Papert at tænke "mekanisk", og henviser til den analytiske "trin-for-trin" tilgang der er påkrævet for at få computere til at udføre kommandoer sekventielt. I debugging aktiviteter, hvor elever fejlsøger deres programmer, er en sådan "trin-for-trin" problemløsningsstrategi ganske enkelt nødvendig på grund af programmets sekventielle struktur og det er ikke vanskeligt at se umiddelbare ligheder med flere af CT strategierne som f.eks. logisk tænkning, algoritmer og dekomposition, der er i høj grad indeholder samme logiske trin-for-trin tilgang.

Det Papert erfarede i sin forskning var, at børn der programmerer udvikler evnen til at kunne tænke og udtrykke sig om deres egen tænkning (metakognition). Derfor kunne man forestille sig, at et væsentligt element i konstruktionismen er, at eleverne konstruerede noget, der skulle vises frem, anvendes eller formidles til andre, således at elevernes kommunikation om proces og produkt kom i spil. Selvfølgelig betyder det, at noget skal vises frem også noget for elevernes oplevelse af autencitet og meningsfuldhed, hvilket vil blive udfoldet yderligere i beskrivelserne af flowteori (se afsnit 11.2.9). Jeg vil heller ikke her komme mere end på sprogets rolle i forhold til elevernes kognition. Dette udfoldes senere i afsnittet om sociokulturel læringsteori. Men Paperts pointe var, at ved at inddrage eleverne i deres egen tænkning blev de reflekterede erkendelsesteoretikere:

" Det vigtigste for mig i alt dette, er, at børnene gennem eksperimenter ville have gennemgået en læretid som erkendelsesteoretikere, hvilket vil sige, at de lærer at tænke og formulere sig om tænkning" (Papert, 1983 [1980], s. 32)

11.1.6 Programmering i sambaskolen

Papert beskriver en række observationer, når eleverne programmerer. Papert observerede bl.a. hvordan elever naturligt samarbejdede på samme måder som i de praksisfællesskaber han havde observeret på brasilianske samba-skoler og han beskrev hvordan elevernes samtaler med læreren understøttede et mere ligeværdigt forhold mellem lærer og elev. Papert havde på den baggrund en forestilling om, at læringsmiljøet i skolerne, skulle "indrettes" på samme måde som brasilianske sambaskoler (Papert, 1983 [1980], s. 181). Det der kendetegner disse skoler er, at man gennem reel deltagelse lærer af andre og ofte dygtigere dansere. Der er en social samhørighed, hvor både eksperter og novicer kan lære noget. Tilgangene til at skabe gennem programmering er, ifølge Papert, lige så varierede som måder at danse samba på og således kan en nybegynder programmere noget, der er nyt og spændende for eksperten.

I dette speciales kontekst, er det uden tvivl vigtige fund Papert har gjort, men da Papert holder sig på et beskrivende/observerende plan, uden nogen decideret teoriudvikling omkring det sociale læringsmiljø og merkompetentes rolle for elevernes læring, vælger jeg at indeholde Paperts observationer i afsnittet om sociokulturel læringsteori. Denne teori leverer et teoretisk begrebsapparat, der meget mere konkret kan kaste lys over Paperts fund.

11.2 SOCIOKULTUREL TEORI

Hvor konstruktivismen sætter individet i centrum i forhold til forholdet mellem din indre erkendelse og omverdenen, anskuer det sociokulturelle paradigme kognition som noget der er uløseligt forbundet med den sociale kontekst, situation, historicitet og kultur hvori læringen sker. Det er ikke bare omverdensfaktorer der kan "påvirke" kognitionen positivt eller negativt (Dysthe, 2003, s. 16). Sociokulturelle perspektiver på læring bygger på en konstruktivistisk forståelse, af at læring er en aktivt konstruerende proces.

Udgangspunktet er i dette afsnit som sagt Vygotsky, men andre tolkere og teoretikere inden for dette felt leverer også væsentlige begreber til brug i empirianalysen og til udvikling af det didaktiske design. Der udfoldes 2 hypoteser i teorigennemgangen, der søges afprøvet i analysen.

11.2.1 Medieret læring

Vygotsky var af den opfattelse, at kognition og læring er medieret. Det betyder, at den lærende ikke har direkte adgang til omverdenen, men at omverdenen medieres gennem forskellige redskaber. Medierende redskaber bliver brugt om alle typer støtte eller hjælp i læreprocessen, og dækker over både fysiske artefakter, værktøjer, sprog, symboler, visuelle programmeringssprog og mentale tænkeredskaber (Dysthe, 2003, s. 52). Alle disse redskaber er kulturelt frembragt og bærer på en historicitet og det er samspillet mellem deltagelse i sociale praksisser og disse medier, at kultur og historie i den sociokulturelle teori spiller en væsentlig rolle for menneskets kognition. Læring er en form for indkulturering.

11.2.2 Sprog og tænkning

Vygotsky sondrer mellem medfødte, elementære psykiske funktioner og højere psykiske funktioner der er menneskeskabte, socialt og kulturelt medierede. De højere psykiske funktioner handler om bevidst at kunne rette sin opmærksomhed, kunne fokusere sin hukommelse, kunne vælge strategier

i problemløsning osv. – i det hele taget de funktioner, der handler om at bevidst at kunne ”styre” sin kognition.

Vygotsky fremhæver, at enhver funktion i et barns udvikling optræder to gange: *Først* på det sociale niveau (interpsykologisk) og *derefter – derivativt* – på det individuelle niveau (intrapsykologisk) (Dale, 2006). Det er igennem sproget som medie i sociale interaktioner, at der bygges bro mellem den ydre kommunikation og den indre tænkning.

For Vygotsky er det vigtigste redskab for tænkningen semiotiske tankeredskaber. Det vil sige redskaber, der er båret af forskellige tegnsystemer. Hos Vygotsky er det allervigtigste tankeredskab sproget. Sproget er hovedgrundlaget for menneskets tænkning og det er gennem sproget, at mennesket kan kontrollere sin egen kognition. I dette speciale er dette interessant og værd at dykke ned i, da et visuelt programmeringssprog må være at kategorisere som et semiotisk tegnsystem og derved have potentiale for at være betydningsfuld for metakognition som indebærer bevidst at kunne anvende forskellige problemløsningsstrategier.

Evnen til metakognition er selve hjørnestenen i Vygotskys teori (Bråten, 2006, s. 94) . Papert iagttog, hvordan elever der programmerer, udviklede deres evner som erkendelsesteoretikere, der var i stand til at kommunikere om deres tænkning. Da dette speciale handler om bl.a. at undersøge udvikling af CT problemløsningsstrategier er det væsentligt at forstå begrebet metakognition.

11.2.3 Komplekser, begreber og refleksiv bevidsthed

Det at udvikle konsoliderede begreber, hænger uløseligt sammen med udvikling af ”refleksiv bevidsthed” (Bråten, 2006, s. 71-73) som en højere psykisk funktion, der er afgørende for f.eks. metakognition. Det at have stærke begreber opbygger en kognitiv struktur, hvor kreativ tænkning og evnen til at skabe ny erkendelse øges. Vygotsky skelner mellem to typer af begreber. ”Komplekser” og ”Akademiske begreber”.

Komplekser er knyttet til det umiddelbart perciperede, det konkrete. Når ord mangler de hierarkiske forbindelser, der gør at det muligt at kategorisere og forbinde ordet til underbegreber, tænkes der i komplekser. Når man tænker i akademiske begreber med den præcision, grundlæggende forståelse og hierarkiske struktur konsoliderede begreber har, bliver begreberne tænkeredskaber, der muliggør formel, abstrakt tænkning. Vygotsky ser undervisningens fornemmeste opgave, at skabe forbindelse mellem udvikling af akademiske begreber og udvikling af refleksiv bevidsthed (eller metakognition):

”Reflective consciousness comes to the child through the portals of scientific concepts”, skriver Vygotsky (Bråten, 2006, s. 71-72). Skolens opgave er at undervise i fag der indeholder begreber med en vis ”tyngde”. Der skal så at sige være nogle faglige begreber eleverne skal arbejde med gennem forskellige typer af udfordringer og problemløsning. Programmering og matematik indeholder i høj grad akademiske begreber der kan tjene som udviklingskilde for elevernes kognition.

Når eleverne tilegner sig det visuelle programmeringssprog og omsætter matematik til dette sprog, arbejder de med at udvikle akademiske begreber. Ved at iagttage deres egen tænkning i samarbejde med andre, udvikles deres refleksive bevidsthed og derved deres metakognitive evner. Eleverne bliver f.eks. i stand til at gøre problemløsningsprocesser til genstand for refleksion.

Det Papert observerede, kan forklares ud fra udvikling af refleksiv bevidsthed, hvor eleverne udvikler deres problemløsningsstrategier og evne til at kommunikere om deres tænkning, som en ”funktion” af, at de tilegner sig de akademiske programmeringsfaglige og matematiske begreber. Måske derfor er det for Papert væsentligt, at eleverne arbejder skaber offentlige artefakter, fordi det giver meningsfulde lejligheder til, at eleverne sætter sprog på deres forståelse i den offentlige formidling.

Eleverne skal i formidlingen have tænkt over deres forståelse og dette arbejde kvalificeres gennem sociale interaktioner. Papert oplevede desuden, at eleverne "spontant" indgik i samarbejdsrelationer der mindede om samba-skolernes praksisfællesskaber. Her konsolideres de akademiske begreber gennem social interaktion – først eksisterer begreberne "uden" for eleverne i sociale kontekster, hvorefter de gradvist internaliseres gennem en kombination af erfaringer med programmeringssprogets eksekvering og gennem kommunikative interaktioner med læreren og andre elever. I takt med at eleverne konsoliderer de akademiske begreber inden for programmering, kan de udnytte både begreberne og deres refleksive bevidsthed kreativt i skabelsen af ny erkendelse.

Spørgsmålet melder sig: Hvordan kan man forstå Scratch som visuelt programmeringssprogs rolle i forhold til mediering og udvikling af refleksiv bevidsthed? Her er det vanskeligt at finde noget entydig forskning der kan forklare dette. I det følgende henter jeg inspiration fra skriftsprogets betydning for kognitionen. Jeg prøver at placere visuelle programmeringssprog som semiotisk tegnsystem mellem polerne: talt sprog og skriftsprog, for at nærme mig en forståelse af det visuelle programmeringssprogs betydning for kognitionen.

11.2.4 Tale og skriftsprog

I børns udvikling af skriftsproget sammenligner Vygotsky skriftsproget med talesproget, og konkluderer, at der er op til 6-8 års forskel mellem de 2 sprogformers beherskelse. Børn har langt sværere ved at udtrykke sig igennem skriftligt sprog til trods for at de to sprog hos barnet bunder i præcis samme ordforråd, syntaktiske opbygning osv. (Vygotskij, 1971)

Årsagen til dette findes i, at de to sprogs udviklingsprocesser intet har til fælles og at ligheden kun er på det ydre plan. Selv simpel tekstproduktion kræver et højt abstraktionsniveau, bl.a. fordi det mangler intonation og ekspressivitet, fordi samtalepartneren ikke er til stede og fordi det er et sprog der ikke betjener sig af ord, men derimod af forestillinger om ord. Skriftsproget er abstrakt i forhold til det talte sprog på samme måde som formel tænkning er abstrakt i forhold til konkret tænkning. Selvom det talte sprog og skriftsproget deler syntaks og ordforråd, er tankens indre meningsfulde grammatik meget anderledes en skriftsprogets grammatik. Tænkningens indre sprog er maksimalt reduceret og (sikkert) uforståeligt for andre end den tænkende, hvorimod skriftsproget kræver maksimal udfoldelse af sproget (Vygotskij, 1971, s. 275).

I forholdet mellem talesprog og skriftsprog er der til gengæld nogle værdifulde paralleller til udviklingen fra komplekser til akademiske begreber. Tilegnelse og produktion af skriftsprog foregår langt mere viljestyret end tilegnelse af det talte sprog, og det er netop denne kognitive viljestyring, der tvinger barnet til at handle mere intellektuelt (Dale, 2006, s. 67; Vygotskij, 1971, s. 278). Skriftsproget er en abstrakt dialog og en kreativ bevidstgørelse af sproget, der i sig selv fungerer som et medierende tænkeredskab, der støtter eleverne i at udvikle akademiske begreber – og som funktion heraf: refleksiv bevidsthed eller metakognitive evner.

Hvordan kan sammenstillingen af talesproget og skriftsproget belyse hvordan det visuelle programmeringssprog Scratch kan forstås i et kognitivt perspektiv?

11.2.5 Oversættelse af symbolsprog.

Når eleverne skal lære et nyt sprog – som et programmeringssprog er – minder Vygotsky os om, at alle nye sprog fungerer som "sprog af 2.orden". Sprog af 2.orden er det sprog som man endnu ikke har helt klare begreber for eller "løst koblet" i de kognitive strukturer. Dette vil gælde, når eleverne skal tilegne sig det visuelle programmeringssprog.

Sprog af 1.orden er det umiddelbare, tilgængelige sprog, samt de repræsentationer eleven anvender – som f.eks. at "tælle på fingre", tegne, gestik eller tale. Vygotskys pointe er, at sprog af 1.orden altid fungerer som oversættelsesled når nye sprog og begreber skal læres. Dette gør sig gældende i tilegnelsen af skriftsproget, hvor talesproget fungerer som sprog af 1.orden. Matematikdidaktikeren Marit Høines (Høines, 2004) påpeger vigtigheden i, at symbolsprog – som både matematiksproget og et programmeringssprog er – læres bedst, hvis elevens sprog af 1.orden fungerer som oversættelse af symbolsproget. Hun pointerer endvidere, at tegninger(regnehistorier) er et foretrukket 1.ordenssprog for mange børn. Pointen er klar – der bygges bro mellem 1.ordenssprog og 2.ordenssprog ved at vælge et 1.ordenssprog, der er let og naturligt udtrykke sig igennem. Ved at det nye forstås på baggrund af elevens eksisterende naturlige repræsentationer, opnås ejerskab til udviklingen af akademiske begreber og kundskaben opleves meningsfuld og brugbar (Høines, 2004, s. 77).

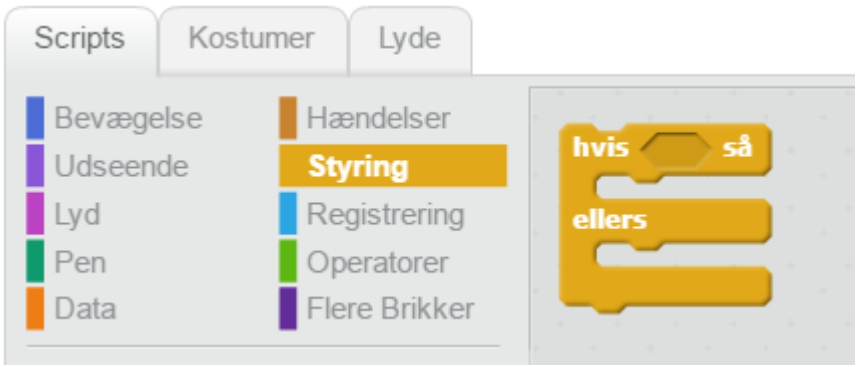
Når eleverne skal programmere, har de et bevidst mål for øje. Programmet skal kunne noget. Ideen til elevernes program befinder sig "inde i hovedet" på nøjagtigt samme måde som en ide til en skriftlig produktion. Når eleverne går i gang med at programmere (f.eks. et spil eller en interaktiv fortælling) eller de skriver en historie i et tekstbehandlingsprogram, eksternaliserer de deres tænkning ind i et symbolsprog. Det er den samme bevægelse fra indre sprog til ydre symbolsprog uanset om de skriver eller programmerer. Dog er der her væsentlige forskelle.

11.2.6 Eksperimenterende oversættelse af symbolsprog

Det særlige ved det visuelle programmeringssprog, er, at eleven kan eksperimentere med sin forestillinger om klodsens betydning. Ved at eksperimentere med klodsen og eksekvere koden, kan eleven umiddelbart se om hypoteserne om klodsen funktion holder stik. Det er et potentiale i det visuelle programmeringssprog, at eleverne kan afkode betydning gennem eksperimenter og på den baggrund danne begreber og i processen udvikle bevidsthed om deres læring. Denne eksperimenterende oversættelse gælder også for de fagfaglige begreber eleverne skal lære. Når eleverne arbejder med variable, koordinatsystem eller sandsynlighed som akademiske begreber, kan de gennem deres program selv udvikle de værktøjer(programmer), der gør det muligt for dem at eksperimentere med disse. Dette kan være med til at danne teoretisk forståelsesramme for Paperts påpegning af computeren som et særligt kraftfuldt medie der kan gøre det abstrakte konkret og herigennem overskride Piagets faseinddeling.

11.2.7 Hypotese 1: Scratch og redundante koblinger til 1.ordens sprog

Hypotesen udfoldes her. I det visuelle programmeringsprog i Scratch fungerer redundant. Det vil sige, at det medierer samme betydning på flere måder og åbner for flere koblinger til 1.ordenssproget som oversættelsesled. I Scratch repræsenteres programmeringsklodserne gennem både ord, farver og form:



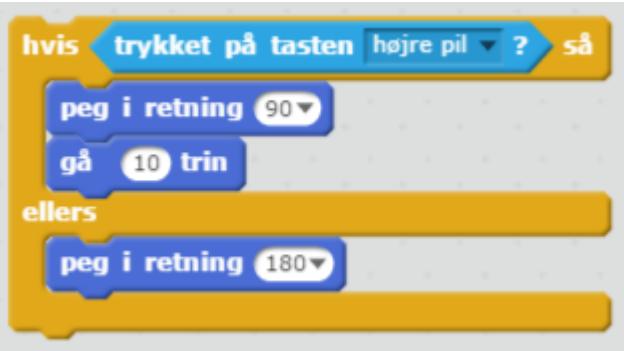
Figur 6: Hvis-så-ellers

I Figur 6 ses en ”hvis, så, ellers” blok. Farven gul medierer, at denne klods kommer fra kategorien ”styring”. Teksten medierer, hvad klodsen mere præcist kan gøre. Klodsens form med de 2 ”indhak” i midten, medierer at denne klods kræver, at der er plads til at noget skal ske ”hvis” en hændelse eller tilstand indtræffer, og ”ellers” skal der ske noget andet.

Via det talte sprog kan klodsens funktion læses højt og mediere nogle ”kognitive stikord”, eleven kan forbinde til sit 1.ordenssprog om ordene ”hvis, så, ellers”. Man kan sige, at konstruktionen ”hvis, så, ellers” ikke bærer nogen mening i sig selv. Det er eleven der selv skal oversætte disse ”kognitive stikord” til meninger i forhold til det mål eleven har for øje og i forhold til elevens erfaringer med andre ”hvis, så, ellers” situationer i elevens sociale hverdagsliv. Her spiller talesproget en væsentlig rolle som 1.ordenssprog. Der er således et kognitivt oversættelsesarbejde eleven må gøre (evt. i samarbejde med andre), før blokken overhovedet giver mening. Dette afkodningsarbejde støtter det visuelle programmeringssprogs redundante multimediering af tegn og symboler(tekst, form og farve) eleverne i at gøre.

11.2.8 Hypotese 2: At tænke med klodser.

For at forstå elevernes tænkning medieret af det visuelle programmeringssprog, tager hypotesen afsæt i klodsernes ”kognitive stikord” som helt centrale ankerpunkter kognitionen. Hypotesen tager udgangspunkt i man som med skriftsproget lære at skrive(kode) ved at læse(kode), og ved at læse(kode) støttes man i at skrive(kode). Efterhånden som eleverne tilegner 2.ordens programmeringssproget – gør det til deres 1.ordens sprog i en kombination af eksperimentering og social interaktion med andre – bliver de gradvist i stand til at kunne transformere det indre sprog – deres tanker – til ydre meningsfulde repræsentationer i Scratch. Deres tænkning kan med afsæt i klodsernes ”kognitive stikord” afkodes og formidles gennem talesproget i særlige ”scratchsætninger”. F.eks. vil denne meningsfulde og syntaktisk korrekte sætning: ”Hvis jeg trykker på højre piletast så skal katten dreje mod højre og gå 10 skridt og ellers skal den bare pege fremad.” i Scratch kode se således ud:

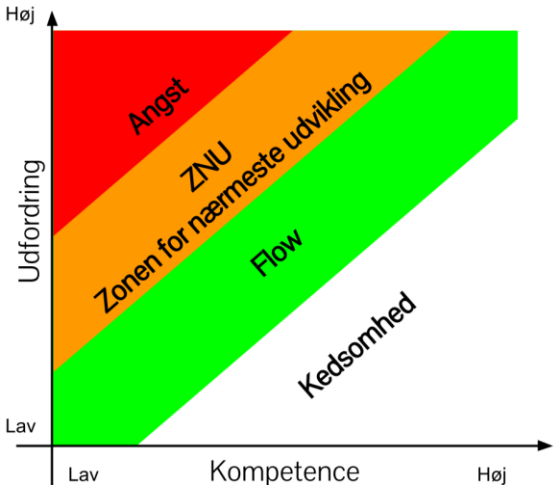


Figur 7: Kognitive stikord i Scratch

Hypotesen der vil blive afprøvet i analysen er, at det at lære scratchprogrammering og blive dygtig til det, er at konstruere logiske og sammenhængende sætninger, man tænker og udtrykker sig igennem. Sætningerne tager afsæt i klodsernes "kognitive stikord" og kode (2.ordens sproget) kan tænkes og udtrykkes via oversættelse gennem talesproget af 1.orden. Tankens indre sprog, den ydre tale og programmeringssproget bliver sammenvævet og understøtter og udvikler hinanden. Viser eleverne tegn på, at de udtrykker deres tænkning og forståelse af kodesekvenser gennem meningsfulde "scratchsætninger" bundet sammen af disse kognitive stikord, vil det være en indikator for at eleverne udvikler og internaliserer programmeringsfaglige begreber og som funktion af deres bestræbelser udvikler refleksiv bevidsthed og metakognitive kompetencer. Disse "scratchsætninger" kan betegnes som et hybridsprog, et 3.sprog eller et særligt "scratchsprog".

11.2.9 Zonen for nærmeste flow

Flowteoriens (Andersen, 2006) udgangspunkt er, at for at eleverne "trives" og er engageret i læreprocessen (og derfor lærer), skal eleven arbejde med udfordringer, der *matcher* deres niveau. Zonen for nærmeste udvikling (ZNU) (Dysthe, 2003, s. 83) ser derimod kilden til udvikling og læring som det der ligger lige *over* elevens aktuelle niveau, og som eleven med støtte, vil have mulighed for at tilegne sig. Zonen for nærmeste flow (ZNF) (Basawapatna, Repenning, Koh, & Nickerson, 2013) handler om, hvordan denne støtte kan gives, så eleverne bedst fastholdes i læringsflow eller trivsel, når eleverne møder udfordringer, der er over deres niveau:



Figur 8: ZNF som buffer mellem Angst og Flow

Forskning viser, at sandsynligheden for at befinde sig i en optimal læringsmæssig flowtilstand øges, hvis disse seks kriterer være opfyldt. (Knoop, 2004):

1) **Høj grad af indflydelse på læreprocessen.** Når eleverne er med til at vælge aktiviteter og træffe beslutninger øges, og har mulighed for at "styre sig selv".
2) **Konkrete, frigørende mål.** Mål og kriterier skal være tydelige, meningsfulde og disciplinerende og på samme måde rummer mulighed for, at eleverne oplever at de er i kontrol over processen.
3) **Enkle og meningsfulde regler**. Tydelige og let forståelige didaktiske rammer hvad angår organisering, materialer osv.
4) **Udfordringerne skal passe til elevernes færdigheder.** For store udfordringer resulterer i angst, for lette udfordringer i kedsomhed.
5) **Positiv, fremadrettet og konkret løbende evaluering.** Eleverne skal løbende have feedback på, hvordan de klarer sig og om de arbejder sig frem mod målene. Denne løbende feedback skal ikke tage udgangspunkt i mangler, fejl og rigtigt eller forkert som let kan tage modet fra eleverne, men som en positiv proces med tydelige konstruktive, fremadrettede resultater af evalueringen (feedforward).
6) **Fjerne distraktioner.** Fjern unødvendige distraherende faktorer, så eleverne kan koncentrere sig.

Befinder man sig først i en læringsmæssig flowtilstand, opleves en indre motivation i og med at aktiviteten bliver et mål og en belønning i sig selv. Det tyder på, at de elever Papert observerede, befandt sig i en vis grad af flowtilstand.

ZNF pointerer, at forskellige stilladseringsstrukturer kan vedvirke til, at eleverne ikke forlader flowtilstanden, selvom de arbejder med udfordringer, der overstiger deres kompetencer. ZNU fungerer med de rette stilladseringsstrukturer, som en udvidet bufferzone for flow, der kan medieres gennem sprog i sociale interaktioner i klasserummet som Vygotsky primært taler for. Men også andre stilladserende medieringer kan bringes i spil. Digitale online ressourcer og Scratchs multimediering af tekst, form og farver er også en stilladsering. Med nye digitale medier er mulighederne for, at støtte eleverne anderledes, end da Vygotsky i 1930'erne skrev om zonen for nærmeste udvikling, selvom jeg grundlæggende er enig i, at dialogiske samtaler en meget vigtig komponent i stilladsering.

ZNF kan være med til at forklare det engagement og den nedsunkethed som Papert beskriver. I Paperts konstruktionistiske forståelse af læring og didaktik, rammer han meget præcist en række af flowteoriens kriterier. Eleverne lærer gennem selvinitierede eksperimenter, de har stor indflydelse på læreprocessen, der er tydelige rammer for hvad de skal lave og hvilke læremidler de skal bruge og programmeringssproget leverer instant-feedback på, om de lykkes. Også Paperts insisteren på at eleverne skal skabe offentlige artefakter, rammer ind i flowteoriens kriterier. Når det eleverne skaber skal formidles og måske bruges af andre, skaber det en meget tydelig ramme og målsætning for, hvad det man skaber skal bruges til. Samtidigt kan det bidrage til, at eleverne oplever aktiviteterne som meningsfulde med et "personligt formål" for øje, som Papert udtrykker det. Herudover beskriver Papert undervisningsmiljøet, som et praksisfællesskab der ligner det man finder i de brasilianske sambaskoler, hvor eleverne – til trods for de sidder ved hver deres computer – naturligt lærer af hinanden. Denne type social sproglig stilladsering i praksisfællesskaber rammer meget præcist ind i Vygotskys ZNU og det mere "ligeværdige" forhold Papert beskriver, der opstår når fokus flyttes fra at levere rigtige svar til løbende procesevaluering.

Men herudover er det tankevækkende, at præcis programmering skulle indeholde en særlig kilde til at eleverne befinder sig i en flowtilstand, sådan som Papert beskriver. Her kan en mulig forklaring være at programmering – og måske i særlig grad visuel programmering – indeholder nogle iboende potentialer både i forhold til flowteoriens kriterer og til selve stilladseringen inden for ZNU. Her

tænker jeg mere specifikt på om "sandkasse metaforen" - der som tidligere nævnt i afsnit 6 er en hjørnestenen i Scratchs interaktionsdesign – har en signifikant betydning for elevernes oplevelse af flow og hvorvidt sandkassedesignet fungerer som et stilladserende medie

11.3 OPSUMMERING OG HYPOTESER

I dette kapitel har jeg beskæftiget med teori inden for to paradigmer. I forhold til kognitivismen og det sociokulturelle paradigme, mener jeg, at det ikke er et "enten/eller" men et "både/og" i relation til læring som individuelle processer eller sociale processer. Det skal forstås på den måde, at både individuelle og og sociale aspekter ved læringen kan interagere og danne "gensidigt forstærkende spiralformede forbindelser" (Dysthe, 2003, s. 59) . Jeg er enig i, at læring grundlæggende er social, men et pædagogisk spørgsmål er, i hvor høj grad og på hvilken måde kontekst og sociale aspekter påvirke læringen og hvilket forhold der er imellem social og individuel læring. Dette kan didaktisk komme til udtryk i del Olga Dysthe kalder for "læringsmønstre" der veksler mellem primært individuelle mere lukkede opgavetyper i samspil med mere åbne og dialogiske aktiviter i klassefællesskabet. I næste kapitel om det didaktiske design, udfolder jeg lignende begreber som hhv. åbne og lukkede læringssekvenser (afsnit 12.3.2). Helt grundlæggende mener jeg, at nogle læreprocesser er gavnlige at anskue med udgangspunkt i individets tænkning og ikke udelukkende som en del af et socialt diskursfællesskab.

I forhold til hvad der skaber motivation for læring, betoner det kognitive paradigme, at børn er naturligt motiverede hvis de beskæftiger sig med meningsfulde aktiviter og at de motiveres af at lære nyt, når de oplever at noget ikke passer med det de forventer eller tidligere har lært (Paperts akkomodative læring). Det sociokulturelle paradigme, beskriver motivation som noget der har at gøre med de sociale forventninger, som barnet møder i den kultur og det samfund de er en del og hvor eleven oplever sig som værdsat som en der "kan noget" og en der "kan bidrage" til de andre. Motivation i det sociokulturelle perspektiv har altså at gøre med, om det man lærer opleves som vigtigt og det er igen afhængigt af, om de kundskaber man skal lære, bliver betragtet som vigtige i klassekulturen (og hjemme).

Afsnittet har givet anledning til 2 hypoteser der vil blive afprøvet i analysen:

- Eleverne udvikler et særligt "scratchsprog" som støtte for deres tænkning. Sproget formes gennem oversættelse af 2.ordens programmering sprog gennem 1.ordens talesprog og tænkningen kommer til udtryk i et hybridsprog med forankring i programmeringssprogets "kognitive stikord"
- Scratchs multimediering af tekst, form og farve er en stilladserende faktor, der bidrager til at holde eleverne i zonen for nærmeste flow

12 INTERVENTIONSDESIGN

12.1 LÆRERINTERVIEWS

Dette DBR interventionsdesign tager afsæt i en *formodet* praksis, da der ikke er nogen af lærerne der før har programmeret i undervisningen.

Der er gennemført indledende interviews med lærerne, hvor jeg kort vil fremstille lærernes antagelser om de *didaktiske* udfordringer ved undervisning med programmering:

- **Støtte, vejlede og hjælpe eleverne**
  Lærerne mangler de færdigheder, der gør at de tør gå i gang med programmering. Deres udfordring går på at være på "bagkant" med undervisningen, og derfor hverken kunne støtte, vejlede støtte og hjælpe eleverne.
  Lærerne kom selv med input til mulige didaktiske løsninger.
    - Identificer de dygtigste elever og tildel dem en formel vejlederrolle
    - Lade de dygtige elever "stå" for undervisningen. Begge lærere nævnte, at dette krævede en ny lærerrolle, hvor læreren i højere grad skulle "lærer med eleverne".
- **Evaluere faglighed og progression**
  Lærerne mangler viden om, hvordan programmering kan kobles til deres fag. Derfor vil de heller ikke kunne vurdere og evaluere elevernes proces og fagfaglige progression
  Lærernes havde ingen specifikke didaktiske løsninger på dette.

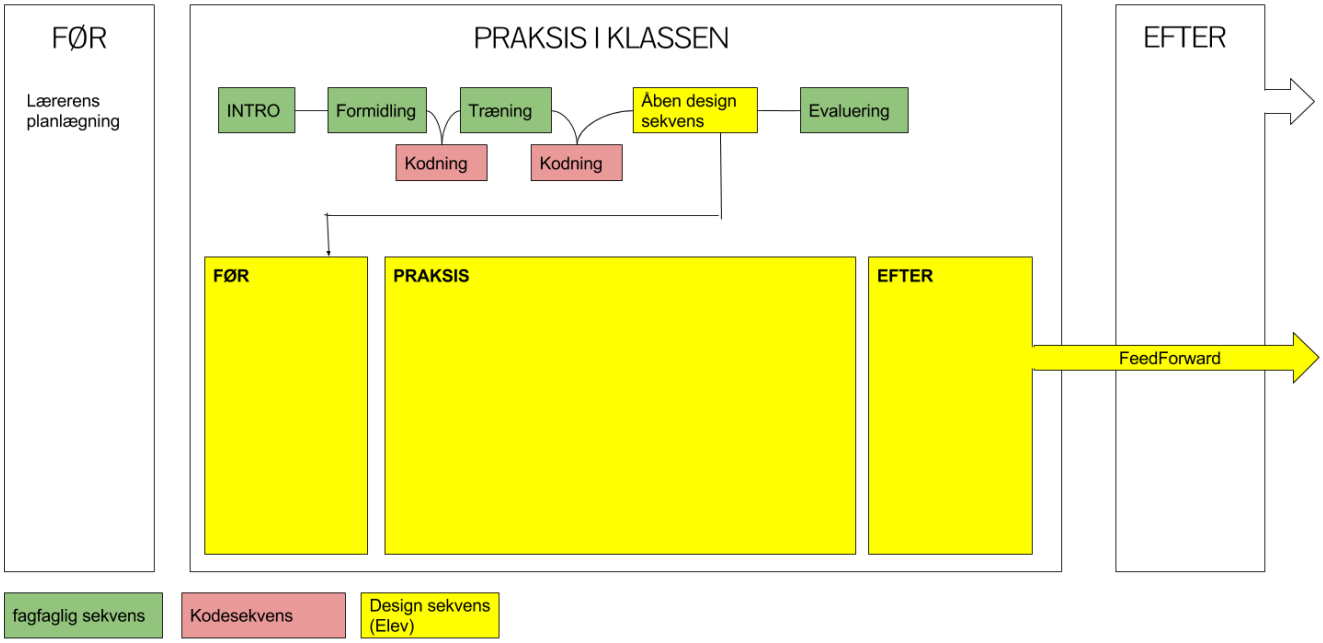
Det er på den baggrund vigtigt for lærerne at et didaktisk design for undervisning med programmering, skal have indtænkt både stilladserings- og procesevalueringselementer.

12.2 Afsæt

Interventionen bygges op som et didaktisk design, der tager afsæt i lærernes formodninger og anvender herudover CAS Barefoots 5 tilgange som designprincipper: Eksperimentering, Kreativ skaben, Fejlsøgning, Vedholdenhed og Samarbejde (Afsnit 7.2). Designet trækker på domænespecifik teori og designteori.

12.3 Didaktisk rammedesign

Didaktisk rammedesign tager afsæt i en tysk-nordisk didaktikforståelse (Sørensen & Levinsen, 2014, s. 29). Fokus er på, at mennesket udvikler kompetence til at tage styring i sit eget liv. Derfor skal eleverne møde muligheder for at træffe valg i forhold til egen læreproces og indgå i en aktiv og styrende rolle i forløbet. Ift. flowteoriens kriterier om indflydelse på læreprocessen, er der tydelige sammenfald. Når eleverne tildeles denne rolle, blive undervisningen mere uforudsigelig end hvis læreren har detailplanlagt alle aktiviteter. Derfor er det væsentligt, at et didaktisk rammedesign tager højde for dette. Det der kendetegner et didaktisk rammedesign er, at læreren ikke har detailplanlagt alle aktiviteter i forløbet, men at den didaktiske ramme kan justeres alt efter hvordan læringsscenariet udfolder sig.

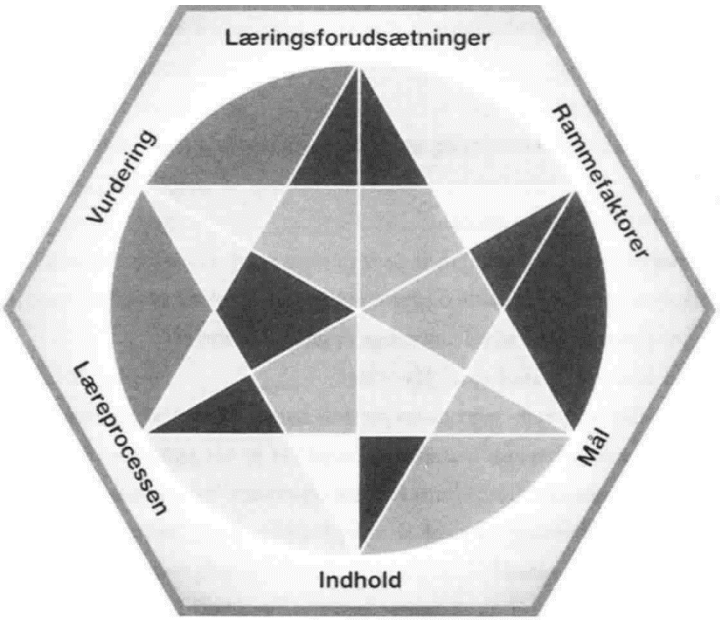


Figur 9:

Modellen består af 3 hvide "kasser", der repræsenterer de tre klassiske didaktiske kategorier: planlægning(før), gennemførelse(under), og evaluering(efter). Det er inden for disse 3 kasser, at læreren udvikler de overordnede didaktiske rammer for forløbet.

12.3.1 Før Fasen

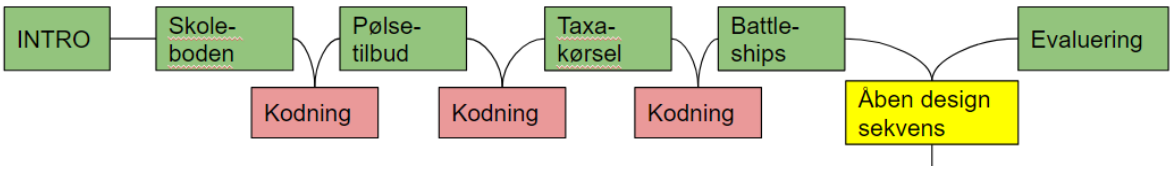
I før fasen har jeg planlægger underviseren overordnet forløbet, f.eks. i forhold til Hiim og Hippes didaktiske relationsmodel (Hiim & Hippe, 1997):.



Modellen uddybes ikke yderligere. Der henvises til undervisningsforløbet (**Fejl! Henvisningskilde ikke fundet.**) for målbeskrivelser og detaljerede beskrivelser af andre didaktiske kategorier.

12.3.2 Praksis i klassen

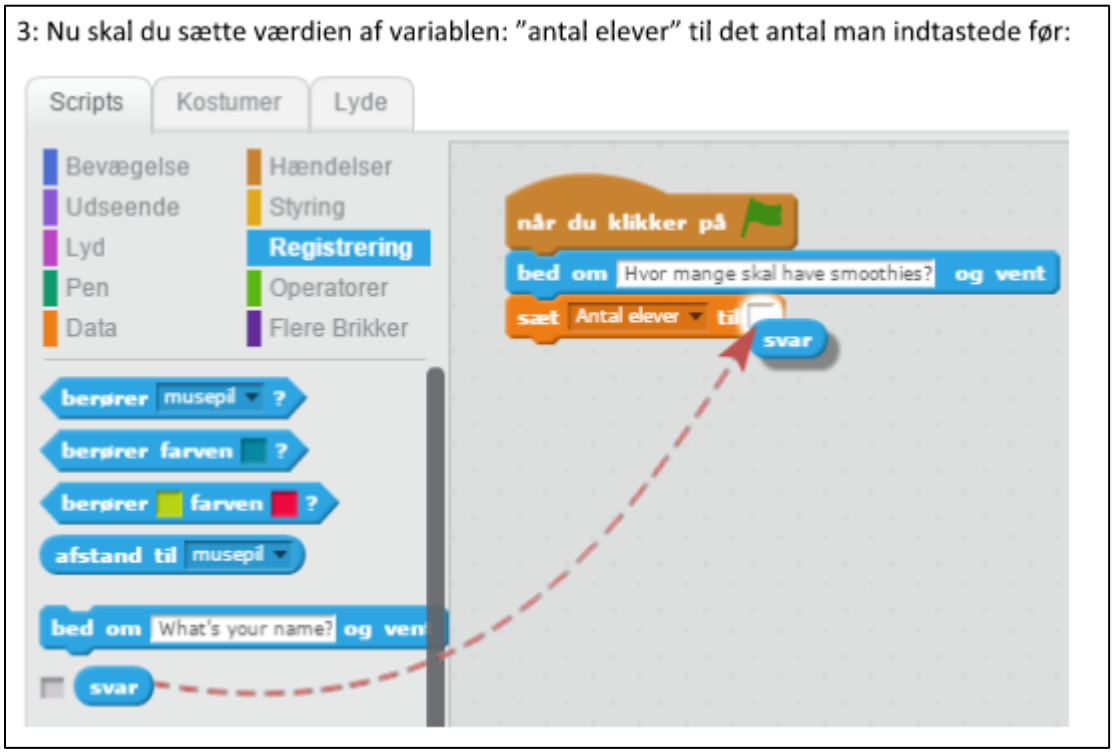
Når både eleverne og underviserne mangler begreber og færdigheder i programmering, kan det blive uoverstigeligt vanskeligt at overføre ens ideer til sit eget program. Der skelnes derfor mellem åbne og lukkede sekvenser i designet:



Figur 10:

Lukkede sekvenser(grønne) kendetegnes ved, at aktiviteterne er bestemt af underviseren. I de lukkede sekvenser præsenteres eleverne for matematiske og programmeringsfaglige begreber og færdigheder, inden eleverne arbejder i en åben sekvens (gul). I Figur 10 ses hvordan sekvenserne var implementeret i 5.klassens materiale. Efter en lærerformidlet introduktion arbejder eleverne med det udarbejde materiale bestående af 4 træningsaktiviteter, der først formidler matematikfaglige begreber og herefter omsætter dem i kodeaktivitet(rød). Efter dette arbejder eleverne med deres projekt (gul).

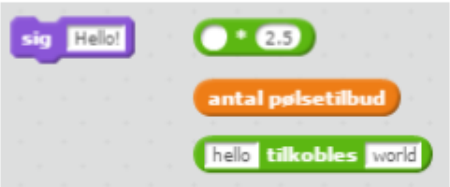
Progressionen i de røde kodesekvenser går fra "worked examples" til opgaver, der kræver stigende grad af forståelse:



Figur 11: "Worked example"

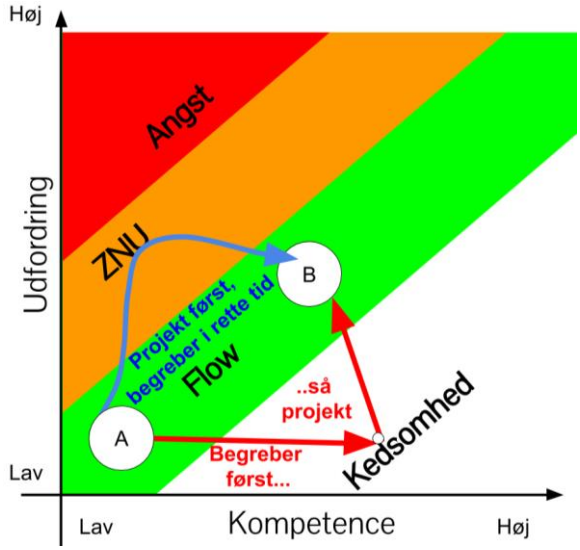
Hjælp 2: Programmet skal regne ud hvor meget 5b har tjent i alt på salget. Der får du brug for at vide hvor meget 5b tjener ved at sælge 1 pølsetilbud (opgave 5e), og så gange det beløb de tjener pr. pølsetilbud med det antal der er blevet solgt.

Her er nogle klodser du kan få brug for:



Figur 12: Progression fra "worked example" til opgavetype, der forudsætter erfaring og forståelse

Lukkede sekvenser ønsker, at ruste eleverne med tilstrækkelige færdigheder til at kunne lykkes med deres designprojekt. Her vender jeg kort blikket tilbage det forskningsprojekt hvori ZNF (Afsnit 11.2.9) blev udviklet. Projektet undersøgte to forskellige måder, at ruste eleverne med begreber: 1) Begreber først, så projekt. 2) Projekt først, begreber "i rette tid". Figuren nedenfor viser forskningsresultaterne.

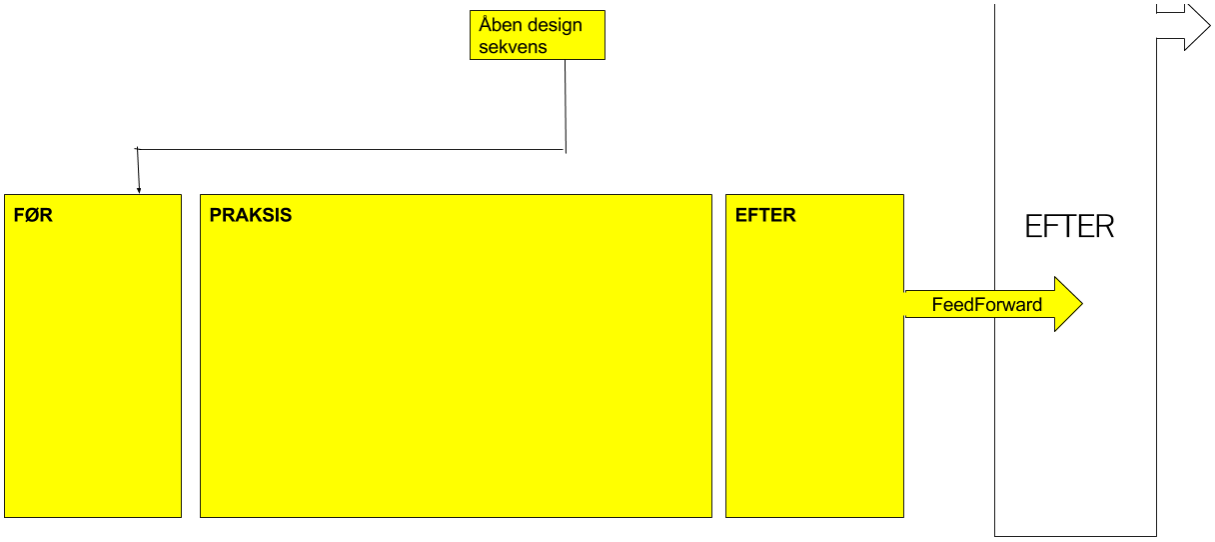


Figur 13: ZNF - Begreber først, begreber i-rette-tid

Figur 13: ZNF - Begreber først, begreber i-rette-tidFigur 13 viser, at med den rette stilladsering i form af begreber i-rette-tid, guides eleverne tilbage i flowtilstanden, når de møder udfordringer over fagligt niveau. Eleverne oplever begreberne som meningsfulde, og væsentlige og derfor ikke "kedsommelige" og frakoblet konkret formål. I dette didaktiske design, bliver begreberne leveret *først* (med overhængende fare for mangel af flow og dårligere indlæring), da lærerne ikke besidder den faglighed der gør, at de kan levere begreber i-rette-tid. Dette er et væsentligt opmærksomhedspunkt i designet.

Åben designsekvens(gul).

I denne sekvens agerer eleverne "didaktiske designere" (Sørensen & Levinsen, 2014, s. 29). Dette begreb dækker egentlig blot over, den medbestemmelse eleverne har, i forhold til at planlægge for deres egen læring(før) inden for læringsmålene, gennemføre disse planer(praksis) og evaluere deres egen proces og læring(efter):



Åbne designsekvenser, bærer en høj grad af uforudsigelighed, som underviseren ikke kan detailplanlægge sig ud af. Derfor er der i særlig grad behov for, at kunne justere den didaktiske ramme undervejs – f.eks. ved at indlægge forskellige formidlings, præsentations- og evalueringssekvenser ind på ad-hoc basis (Afsnit 12.5). Forholdet mellem åbne/lukkede sekvenser kan også justeres ad-hoc i processen, ved f.eks. at indlægge flere lukkede sekvenser.

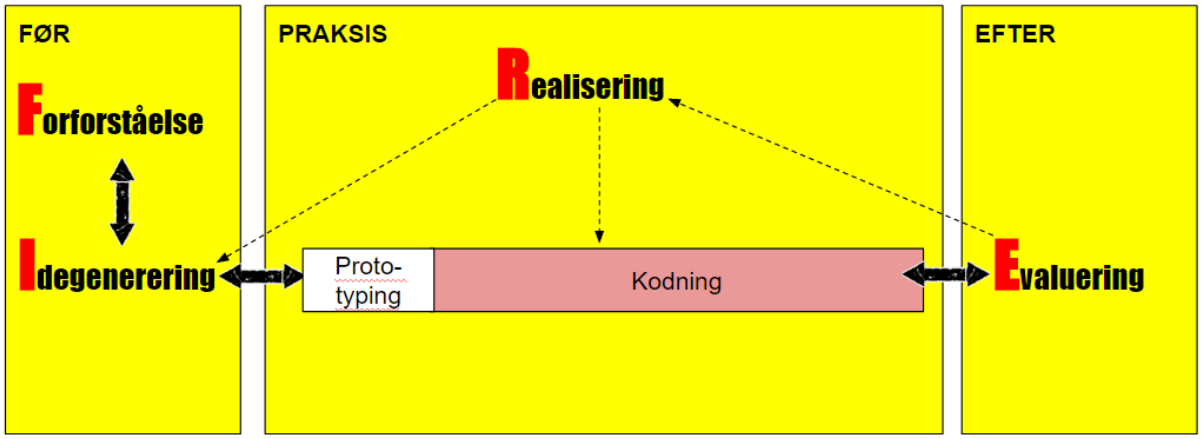
Eleverne skulle i forløbet samarbejde efter konceptet ”pair-programming” (Jeffries, 2011). Par programmering går ud på, at eleverne programmerer sammen 2 og 2 ved 1 computer. Eleverne skiftes i intervaller til hhv at programmere og holde overblik eller komme med gode ideer. I forhold til CT tilgangen kollaboration, sikres der her, at elevernes samarbejde ikke blot består i at uddelegere arbejdsopgaver (kooperation (Bang & Dalsgaard, 2005)), men udnyttelse de kollaborative synergier, der kan opstå ved, at flere beskæftiger sig med det samme på samme tid. Par-programmering begrundes i forhold til den sociokulturelle læringsteoris betoning af sproglige interaktioners betydning for læring.

12.3.3 Efter Fasen

Lærerens egen efterbearbejdning af undervisningen med udgangspunkt i bl.a. elevernes evaluering og feedforward til kommende forløb, er ikke indeholdt i dette didaktiske rammedesign. Det har været op til de undervisere der gennemførte forløbet, og er ikke indeholdt i dette speciale.

12.4 FIRE Design

I den åbne designsekvens møder eleverne muligheder for CT princippet ”kreativ skaben”, hvor det man skaber, skal løse et problem eller en udfordring for andre. Jeg vurderer modellen: FIRE design (Figur 14) (Rohde & Olsen, 2013), velegnet til at facilitere underviseren i at arbejde med ”eleverne som designere” og CT tilgangen ”kreativ skaben”:



Figur 14: FIRE design

Den praktiske indarbejdning af FIRE designmodellen i de gennemførte forløb findes i **Fejl! Henvisningskilde ikke fundet.Fejl! Henvisningskilde ikke fundet.**.

Forforståelse – Hvad ved eleverne på forhånd ved om udfordringens tema: Læringspil og Casinospil? I forhold til Vygotskys pointe om, at nyt altid forstås på baggrund af noget kendt, er det væsentligt at eleverne deltager i samtaler i klassefællesskabet om deres forforståelse

Idegenerering – Fokus er her på en bevægelse fra divergent tænkning til konvergent tænkning. Først "tænke ud af boksen" da vilde idéer ofte fører innovative tanker med sig. Herefter udvælger eleverne konkrete ideer der skal arbejdes videre med.

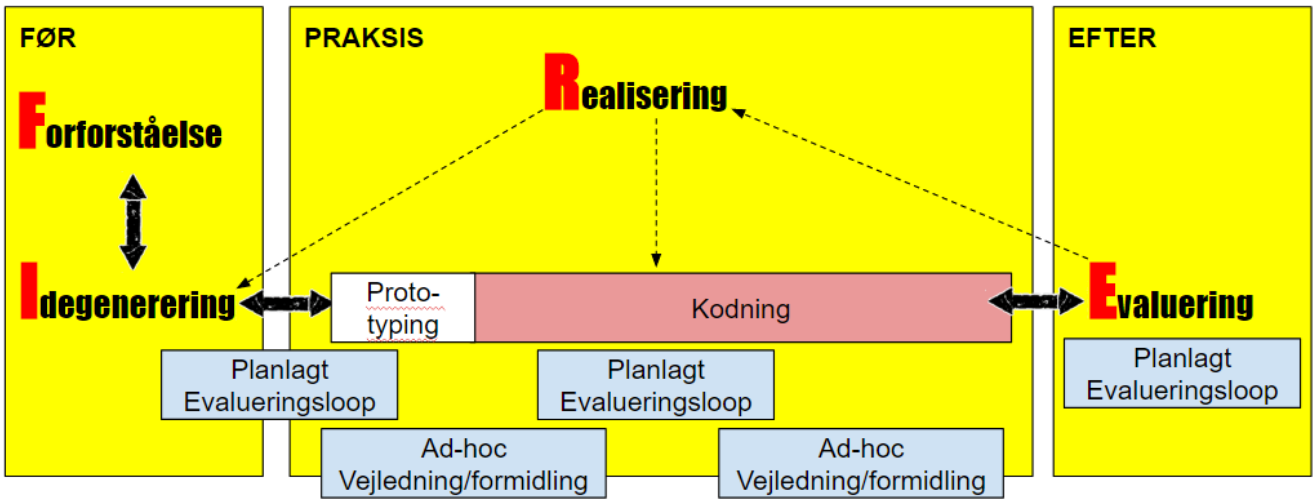
Realisering – Eleverne "lægger sig fast" på en idé. De udarbejder en prototype i form at skitser og tegninger af programmets forskellig elementer. Prototypen præsenteres og kvalificeres via feedback i klassefælleskabet, så viden og ideer spredes og samtidig kan læreren holde eleverne fast på opgavens formål og kriterier. Eleverne går herefter i gang med programmere.

Evaluering – Eleverne afprøver deres løsning på udfordringen, ved at lade målgruppen afprøve deres design. Elever fra en 3. klasse afprøver 5.klassernes læringspil, og 7.klassen afholder en Casino event for deres parallelklasser. På den baggrund evalueres deres design og processen.

FIRE designet har fokus på, at designe for en autentisk målgruppe ved at løse meningsfulde udfordringer. I forhold til det konstruktionismens tese om, at læring foregår bedste ved at skabe offentlige artefakter og det at præge egen læreproces inden for tydelige konkrete mål, vurderes denne model velegnet til at udvikle ejerskab og engagement i læreprocessen og optimere muligheder for flowoplevelse.

12.5 LØBENDE PROCESEVALUERING

Løbende fremadrettet procesevaluering er et kriterie i flowteorien og kan ses som en komponent i Vygotskys betoning af sociale interaktioners betydning for læring og de dialogiske samtaler med mervidende i ZNU. Løbende procesevaluering er også et nedslagspunkt i lærerens antagelser. Begrebet undervisningsloops (Gynther, 2010) operer med 3 loops: Evalueringsloop, Vejledningsloop og Formidlingsloop. Disse er indlejret i designsekvensen (Figur 14):

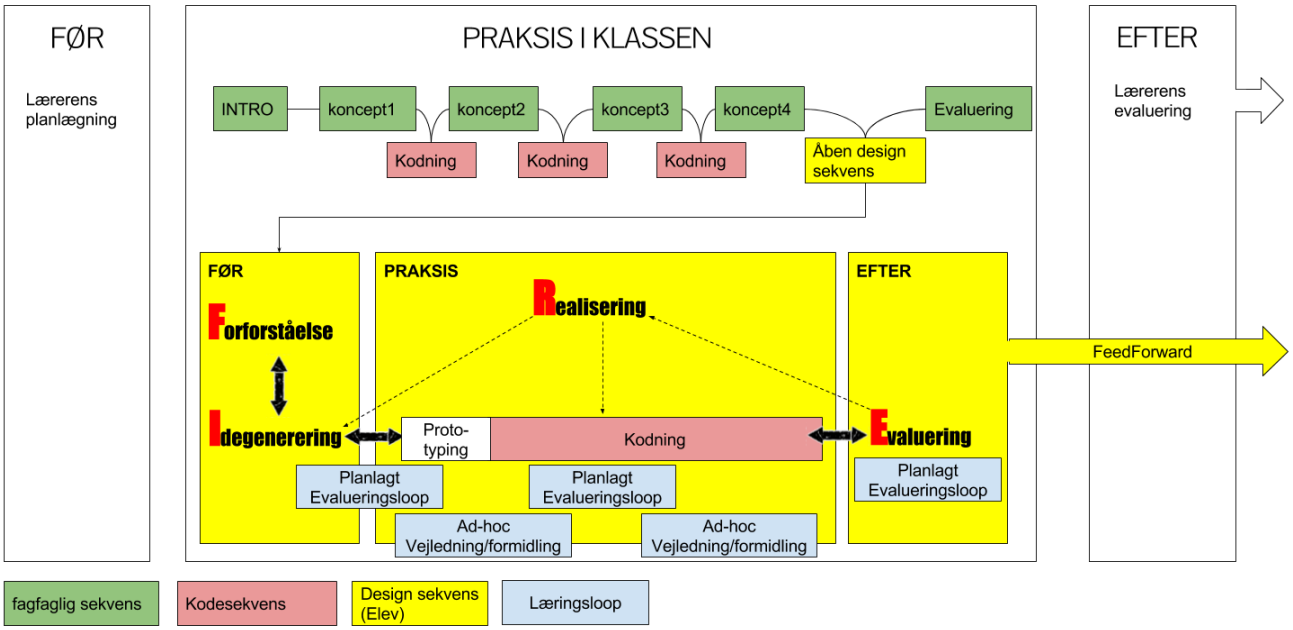


Figur 15: Undervisningsloops

I evalueringsloops præsenterer eleverne deres "work-in-progress" og får respons fra klassefællesskabet. I det det didaktiske design er evalueringsloops planlagt bevidst efter prototyping fasen, i kodefasen og til sidst, hvor eleverne præsenterer deres produkt. Det skal være med til at sprede viden og samtidig udnytte klassefællesskabets konstruktive feedback. Det giver underviseren et fingerpeg om elevernes progression og udfordringer og kan give anledning til ad-hoc vejledings- eller formidlingsloops.

I et formidlingsloop modtager grupper eller klassen faglig matematik- eller programmeringsfaglig formidling efter behov. Vejledningsloops har fokus på selve processen og de læringsveje eleverne vælger.

Her præsenteres her det komplette interventionsdesign:



Det er ikke alle 5 didaktiske CT principper er der direkte er indarbejdet i designet. Eksperimentering og fejlsøgning fremgår ikke eksplicit, fordi de afledt af selve læremidlet Scratch, der rummer muligheder for dette. CT princippet "vedholdenhed" er vanskeligt at designe for. I møder med

underviserne har vedholdenhed været et mundtligt formidlet omdrejningspunkt og empirien viser, at eleverne har været meget bevidste om dette.

13 ANALYSE

13.1 ANALYSETEMA1: EJERSKAB TIL LÆRING OG OPLEVELSE AF FLOW

Analysetemaet fokuserer på elevernes oplevelse af, hvad jeg vil kalde "ejerskab" til læreprocesser og oplevelse af flow. Det er en oplevelse, der i empirien kommer til udtryk i elevernes fortællinger om stolthed, engagement og om hvordan det er betydningsfuldt for dem, at vise deres produktioner frem for andre. Jeg ser desuden ejerskab til læring som et udtryk for at have oplevet en vis grad af læringsflow. Jeg mener, det er vanskeligt at forestille sig læringsflow uden udvikling af den "ejerskab" jeg finder i empirien.

13.1.1 Udfordringers betydning for udvikling af ejerskab og flow

Min dataanalyse viser, at de interviewede elever har oplevet forløbet som udfordrende både i forhold til det der handler om at skulle agere som elev i en "anderledes" form for undervisning – mere specifikt i FIRE designsekvensen - og om udfordringer af mere programmeringssteknisk karakter.

Denne dobbelthed i oplevelsen af udfordringer udtrykker Emil fra 5 klasse på denne måde:

> *I: Er der forskellige måder noget kan være svært på?*
> *E: Ja.. noget kan være svært sådan mentalt på en måde at øhhh… hvad skal vi sådan .. altså.. hvad fanden skal vi egentlig gøre lige nu? Og det andet, det er sådan, så at få sat det sammen på den rigtige måde og sådan*
> *I: Så det kan være svært at tænke "hvad skal vi i det hele tage gøre for at løse udfordringen" og så kan det også være svært at så gøre det?*
> *E+T: Ja*

For det første kan *"Øhh hvad fanden skal vi egentlig gøre lige nu?"* hentyde til overvejelser over hvordan ens program skal være og hvad det skal indeholde for at løse udfordringen. For det andet oplevede eleverne det samtidigt udfordrende, at skulle omsætte deres ideer til kode. *"..at få sat det sammen på den rigtige måde*", kan tolkes som selve det at mestre programmeringssproget.

Undersøgelsen viser, at eleverne havde en generelt positiv holdning til udfordringerne med den mere frie arbejdsform de mødte i FIRE designet.

Emil fra 5.klasse:

> *I: Hvilke ting synes I godt om i forløbet?*
> *E: At vi selv skulle prøve at lave det fra bunden af.*
> *T: Ja det er lidt federe selv at lave det*
> *E:.. i stedet for man bare har fået læreren til at sidde og lave det hele…*
> *T: Ja..*
> *I: Hvad var i fokus i forløbet, hvad var det I skulle lære?*
> *E: Ja, altså – forstå matematikken på en anden måde.*
> *I: Ja, hvordan? Kan du prøve at uddybe det?*
> *E: Ja, altså – så vi ikke bare sidder med vores bog og vi også lærer det på en anden måde. En sjovere måde på en måde.*
> *I: Ja, hvordan det? En sjovere måde på en måde – hvad mener du?*
> *E: Ja, det er meget sjovere sådan at sidde og lave vores egne ting i stedet for vi bare skal skrive et svar ned.*
> *I: Okay, så du kan godt lide det der med at der er nogle af ens egne ting man skal lave.*
> *E: Ja*

Amalie fra 7.klasse

> *I: Er der nogle ting I synes om ved at programmere frit i forhold til at følge en vejledning?*
> *A: Det er fedt at man bare selv kan bestemme hvad man vil programmere i stedet for sådan noget med hvor meget 20 is koster til 6 børn og sådan noget.*

Valde og Lucas, 7.klasse

> *I: Hvad synes I om at det tager udgangspunkt i jeres egne idéer når I skal lære noget?*
> *V: Det er fedt…*
> *I: Ja.. Hvorfor?*
> *V: Altså når det er, at man på en måde selv bestemmer hvad man gerne vil lave, så er det meget sjovere at lave det …*
> *L: Det er lidt mere frit*

Paula og Marillo, 5.klasse

> *I: Hvad synes I om at jeres projekt tog udgangspunkt i jeres egne ideer når man skal lære noget?*
> *M: Jeg synes det er sjovt*
> *P: Det var fedt og det var sjovt*
> *M: Sådan så man kunne lave noget som man .. sådan … selv havde fundet på agtigt.. nærmest*
> *I: Hvad betyder det?*
> *P: At det mere er for en selv .. det er mere ens egen*
> *M: ..ja, sådan at man kan sige: "Det der har jeg lavet"*
> *P: .. i stedet for at man har sagt at det var efter en vejledning eller sådan*
> *M: ja*

Eleverne oplever at de "selv at komme på banen", at de synes det er fedt og sjovt selv at bestemme hvad "man gerne vil". Det har hos alle interviewede elever en positiv betydning for deres oplevelse af forløbet og styrker oplevelsen af at "det er deres eget". Vender vi blikket tilbage mod flowteorien, kan elevernes udsagn forstås ud fra kriteriet om at have "indflydelse på egen læreproces" og at de oplevede målet for den åbne designsekvens, som ikke-besnærende, men frigørende – eller "mere frit" som Lucas formulerer det.

Der ligger i flere citater en antydning af, at den positive indstilling til arbejdsformen har at gøre med oplevelse af meningsfuldhed. F.eks. at det er sjovere end at finde ud af hvad *"20 is koster til 6 børn".* For det første er sætningen noget forkvaklet bygget op og giver begrænset mening. Man kan tolke det i retning af, at meningen er underordnet, fordi den type opgaver kan ses som et svar på en pseudo-opgave, der ikke har nogen reel betydning for eleven. Hovedsagen er, at eleven perspektiverer til opgavetyper der traditionelt kendes fra matematikbogen, og at eleven italesætter dette som mindre "sjovt" eller meningsfuldt. Meningsfulde aktiviter, er, som vi så det i både hos Papert og Vygotsky væsentlige for motivationen og for udvikling af ejerskab. Et tydeligt eksempel på det at opleve mening, formål og ejerskab til det man beskæftiger sig med, fortæller Amalie og Cecilie fra 7.klasse om:

I: Hvad betyder det at I selv har lavet det?
A: Det er sådan mere vores eget i stedet for bare at gå ind på en eller anden hjemmeside hvor der er spil som man ikke selv har lavet..
C: ..Så det er sådan lidt specielt..
A: ja..
I: Betyder det noget for den måde man arbejder på, at andre skal bruge det man laver?
C: Det har en betydning for.. Jeg tror det betyder noget for en, man er sådan lidt stolt over at man har bygget noget op.

Ovenfor er beskrevet hvordan eleverne oplevede FIRE designprocessen som positiv og hvordan deres positive oplevelse kan forstås ud fra nogle af flowteoriens kriterier. I forhold til denne del af udfordringerne at eleverne er blevet udfordret på deres faglige niveau. Eleverne var alle i stand til at kunne udfylde en åben didaktiske ramme og deres kreativitet og fantasi i processen. I forhold til *denne del* af udfordringerne befinder eleverne sig inden for kriteriet om flow, hvad angår sammenhængen mellem deres kompetencer og de stillede udfordringer.

Ved et andet aspekt af oplevelsen af at være udfordret – nemlig udfordret på det at beherske programmeringssproget – ser billedet ikke helt så entydigt ud. Der er en tydelig tendens til, at eleverne i 5.klassen oplevede udfordringer i programmeringssproget som primært positivt, hvorimod der er en knap så positiv tendens at spore hos eleverne i 7.klasse. På tværs af klasserne er der endvidere en sammenhæng mellem det at have klaret programmeringstekniske udfordringer og oplevelsen af ejerskab til deres produkt.

Amalie og Cecilie fortæller om hvad det betyder for ejerskab af at have løst programmeringstekniske udfordringer selv:

I: Hvad plejer I at gøre når noget er svært i matematik?
A: Så plejer man at spørge læreren.
C: Ja (griner)
I: Er det noget man gør som noget af det første?
A: Ja, men det er vi sådan holdt lidt op med.. Nu tror jeg bare vi prøver
I: Var det anderledes i det her forløb i forhold til almindelig matematikundervisning?
C: Ja her kunne vi ikke få hjælp til det, så vi var selvstændige til det og det var meget godt også
I: Hvad synes I om det?
A: Det var ret svært lige i starten fordi man var vant til at spørge læreren
I: Var det noget i vænnede jer til?
C: Ja
I: Og hvad synes I om det?
C: Det var meget rart for så kunne man selv finde ud af det og så kunne man selv se man var blevet bedre
I: Når man finder ud af nogle ting uden at spørge læreren, hvordan føles det så?
A: Det føles godt.. så er man sådan stolt fordi man har fundet ud af det

Det Amalie og Cecilie her nævner, er et tegn på noget af det der kendetegner følelsen af ejerskab. Efterhånden som eleverne erfarede, at lærerhjælp – specielt i 7.klassen – var stærkt begrænset, oplever de, at de selv overkommer nogle af udfordringerne. Cecilie forbinder det med en rar følelse, fordi man "selv kan se at man er blevet bedre". Det er en følelse der er centreret omkring hende selv. Det føles ikke rart af ydre årsager f.eks. fordi opgaven er overstået eller fordi man har gjort det læreren bad om. Den "rare eller gode" følelse opstår, når man kan mærke, at man *selv er blevet bedre*, og kombineret med Amalies beskrivelse af oplevelsen af stolthed, er citatet en god beskrivelse af den ejerskabsfølelse til egen læring og produkt, som langt de flest elever i varierede grader beskriver når de overkommer udfordringer i programmeringssproget. Her et eksempel fra 5.klassen der viser en lignende tendens:

Petrine 5.klasse:

I: Er det noget tidspunkt hvor I har oplevet helt selv at kunne løse et problem i gruppen? Så har I kommet igennem det alligevel selvom det var svært?
E+P: Ja
I: Hvordan opleves det?
P: Det føles sådan virkelig godt at man sådan har gjort det og sådan… man føler godt at man KAN noget.. og ikke bare sådan skal spørge folk

Alle eleverne oplevede programmeringstekniske udfordringer, men der er forskel på graden af frustration dette har medført. Hvor 5.klassen generelt så udfordringerne som noget man er "vokset" af, er der en tendens i 7.klassen til, at man har stået over for udfordringer, der har været så store, at man ikke er kommet ud på den anden side med en positiv oplevelse. Her først et par citater fra 7.klassen:

Amalie og Cecile, 7kl:

I: Hvad synes i om at gøre noget der er svært.. er det vigtigt når man skal lære?
A: altså… jeg er ikke så stor fan af det (griner). Jeg kan godt lide det er nemt at gå til.
C: Men det er også vigtigt at gøre noget der er svært - det er det man lærer noget af.
I: Bare det ikke er FOR svært?
A+C: Ja

I: Kunne I tænke jer mere programmering i matematik?
A: mmm (nikker)
C: Ja
A: Ja men bare ikke for meget
I: Synes I det har været et hårdt forløb ift at koncentrere sig og møde problemer?
C: Ja, på den led..

I interviewet med Amalie og Cecilie er det generelt tydeligt, at udfordringerne har givet anledning til frustration. De ved godt, at det kan være frugtbart at beskæftige sig med noget svært, men det må heller ikke være uden for deres rækkevidde. De giver udtryk for, at det var et hårdt forløb og at det godt må være nemt at gå til. Det kan være en forklaring på, at Amalie og Cecile synes godt om de lukkede sekvenser i det didaktiske design.

I: Hvad synes I om at skulle følge sådan en vejledning frem for bare selv at finde på?
A: Der var lidt mere styr på det og orden når man følger vejledningen
C: ja så man ikke vidste hvor man var, men vidst at man var nået til side dit og dat og havde mere styr på det.
I: Hvad er fordelene ved at følge sådan en vejledning?
A: Så man ved hvor man er nået til… Nogle gange når man selv laver det, så ved man slet ikke hvor man er nået til.

Eleverne i 5.klasse oplevede ikke udfordringer med programmeringssproget på samme måde. Selvom de havde en masse udfordringer fastholdt eleverne en positiv holdning. Emil fra 5.klasse:

I: Synes I at det har været et hårdt forløb… At man skulle koncentrere sig og at man skulle blive ved og at der var ting der var svære?
T: Ja det har det
E: ja, men har stadig været meget sjovt
I: Ja..
E: Altså jeg kan meget bedre li' matematik efter det her

13.1.2 Klassefællesskabet betydning for oplevelse af flow

En af årsagerne til at eleverne i 7.klasse oplevede frustrationer i særligt FIRE designprocessen, kan være, at de oplevede et fragmenteret forløb, hvor læreren ofte ikke var til stede og klassen havde forskellige vikarer. Flere elever har været udfordret over evne og der var begrænset stilladsering, der kunne bringe dem i ZNU og tilbage i flow. De dialogiske samtaler, hvor eleverne gennem samtale og

støtte fra en underviser, måske ville kunne overkomme deres udfordringer, var ikke tilstrækkeligt repræsenteret. De stilladserende faktorer der fungerede for 7.klassen, var derfor worked examples i de lukkede sekvenser. Nærmest bebrejdende udtrykker Amalie det således:

I: Det her med at prøve igen, var det noget I havde snakket om på klassen?
A: Ja, det med vedholdenhed og vi skulle ikke give op og bare fortsætte
I: Var det svært?
A: Meget!
I: Er I blevet bedre til det synes I?
C: En lille smule.. Ja
I: Hvordan er I blevet bedre til det?
A: Det må du nok spørge vores lærer ad! Vi fik jo ikke lov til at få hjælp af ham hvis det var, vi fik hele tiden lov til at sætte os ned igen og så måtte vi prøve igen.

Læreren har ikke været til stede og dette har givetvis præget 7.klassens arbejde og er blevet forstærket af, at kun begrænset har kunne udnytte støtte i klassefællesskabet.

C: ja der var en gang hvor vi skulle fremlægge hvor langt vi var nået sådan foran alle
I: Hvordan gjorde I?
A: Vi forklarede hvad vi havde lavet og hvad vi ikke kunne finde ud af og hvad vi skulle have hjælp til og om der var andre i klassen der kunne finde ud af det og kunne hjælpe
I: Var der noget hjælp at hente der?
C: Nej for det var nogenlunde det samme alle havde problemer med

For 5.klassen har klassefællesskabet i højere grad fungeret stilladserende i forhold til spredning af viden og det at kunne arbejde i ZNU med støtte fra mervidende. Når eleverne i 5.klassen fortæller om klassefællesskabets betydning for at løse programmeringstekniske udfordringer, udtrykker Paula og Marillo, 5kl det således:

P: Vi havde også et problem, vi ville gerne have det til at.. når man ramte et æble at der kom sådan oppe i hjørnet øhm hvor mange æbler man havde fanget , og det brugte vi også ret meget tid på at finde ud af hvordan man gjorde.
I: Fandt I så ud af det?
M+P: Ja
M: Og så hjalp vi så også en anden gruppe med hvordan man gjorde det
I: Ja..Fedt
I: Oplevede I at grupperne hjalp hinanden undervejs og gav hinanden råd?
P: Altså der var en som ret tit gav råd til andre
I: En af jeres klassekammerater som er dygtig til det?
P+M: Ja
I: Så ham brugte I meget?
P: nej vi gjorde ikke
M: Men mange af de andre .. vil du hjælpe med det her eller sådan hvordan laver man det.. sådan noget

I 5.klassen blev alle evaluerings og formidlingsloops gennemført. Dette var noget mere tilfældig i 7.klassen –nogle af eleverne kunne vanskeligt erindre det eller også tillagde de det kun lidt betydning for arbejdet. Gennemførelsen af evaluerings- og formidlingsloops betød at eleverne delte viden på klassen til fælles gavn. Eleverne i 5.klassen beskrev disse loops som betydningsfulde. Her udtrykt ved Emil og Tobias:

I: Ja. Undervejs der var der sådan at man lige fremlagde hvor langt man var kommet. Hvordan var det at høre om og se de andres programmer undervejs?
E:Altså, vi fik jo nogle nye ideer ud af det – sådan hvad vi kunne lave til vores eget spil.
I: Oplevede I, at der var andre der fik hjælp af andre i klassen?
E+T: Ja… det gjorde vi også selv
I: Nå øh – kan I lige prøve at fortælle om det?
E: Altså, vi vidste ikke sådan helt hvad … Vi var ikke nået så langt der vi skulle op og præsentere – så fik vi lidt ideer fra de andre.. med at den burger der den så rykkede sig og sådan noget og at det ikke kun var 1 spil og så det fortsatte.
I: Okay.. Så der var de andre gode til at hjælpe?
T+E: Ja…

I forhold til læreren kompetencer som støtteperson i ZNU, gælder det for begge klassers vedkommende, at læreren ikke var ekspert i programmeringssproget.

I: Hvad med det der med at spørge læreren?
V: Det brugte vi ikke så meget… for vores lærer var heller ikke lige så god til det der programmering
I: Nå okay (griner)
L: Der var også andre der skulle have hjælp af ham, så det gik ikke at spørge ham.. det gik alt for lang tid.. så er det bedre bare at prøve selv jo.

Det fremgår af interviewene at elevgrupperne i 5.klasse ofte spurgte læreren om hjælp, men at de også måtte finde andre strategier end at spørge læreren. Læreren kendte ikke altid svarene, og man kan tolke det i retning af, at lærerens samtaler med eleverne har båret præg af, at læreren "lærer" sammen med eleverne. Det kombineret med, læreren i denne klasse indgik i samtaler med eleverne i de gennemførte evaluerings- og formidlingsloops, som empirien viser, at eleverne oplevede værdifulde, kan inspirere til en kobling til Paperts beskrivelser af et mere "ligeværdigt" forhold mellem elever og lærere. Undersøgelsen kan også indikere, at indlagte loops med videndeling, er med til at gøre eleverne opmærksom på potentialet "i det de andre laver" og på den måde kan dette være medvirkende til opbygning af Paperts "samba-skole" miljøer, hvor eleverne er optaget af at hjælpe hinanden som 5.klasserne viser tegn på.

Eleverne i 5.klasses ytringer om deres lærers rolle, de andre elever og de fælles opsamlinger undervejs, peger på, at de opfatter de sociale interaktioner som værdifulde i deres egen proces. Perspektiverer vi dette til flowteorien, handler det også om at få "positiv evaluering" med et fremadrettet perspektiv. Det er f.eks. tydeligt at procesfremlæggelserne undervejs netop har fungeret som fremadrettet positiv kritik og hjælp for 5.klassen.

Der kan være mange komplekse faktorer der har forstærket eller forringet elevernes oplevelse af klassefællesskabet som understøttende læringsmiljø. Her spiller det sociokulturelle motivationsperspektiv (afsnit 11.3) på, hvorvidt den enkelte indgår i et socialt klassefælleskab, der opfatter det værdifuldt eller meningsfuldt at lære, en afgørende rolle. Der er indikatorer for, at lærerens fravær har været betydningsfuld i 7.klassens begrænsede oplevelse af flow, men der er mange andre aspekter ved en klassekulturs kontekst der har betydning for læring. Netop denne kontekst for læring er et grundlæggende element i en sociokulturel læringsanskuelse.

13.1.3 Ejerskab på baggrund at selv at skabe noget er skal bruges eller fremvises

Undersøgelsen viser at det har en betydning for ejerskab, at de selv skaber noget der har et publikum.

5.klassen skulle lave et læringsspil til mindre børn om koordinatsystemet, 7.klassen skulle fremstille digitale spil til en casinodag. For en enkelt af 5.klasses grupperne, Paula og Marillo, betød det dog ikke umiddelbart så meget, at det de lavede skulle bruges af andre:

I: Tænkte i undervejs på det der med at jeres program skulle bruges af andre?
P+M: Næææ..
M: nej
I: Det betød ikke så meget?
M: Nej

Selvom det ikke har betydet meget at andre skulle *bruge det*, så viser det videre interview, at det alligevel er betydningsfuldt for dem at *vise* det frem. At andre skal se det, og helst tænke at dem der har lavet det, faktisk godt kan noget og herved få anerkendelse fra andre:

I: Hvis I skulle nævne en ting man lærer ved at programmere, hvad er det så?
M: Det er nok at man kan få sat sin fantasi ned i nogle klodser kan man vel sige.. og selv lave noget man selv har fundet på og kunne vise det til andre og sige: "Ej se hvad jeg har lavet" sådan uden at prale helt vildt og så sige "ej det ser sjovt ud" og sådan noget
P: At man sådan føler sådan lidt at man har lavet noget som andre sådan synes er sjovt-agtigt… er godt fundet på

Paula og Marillo oplever af at have lagt deres "fantasi ned i nogle klodser". Det er ikke bare nogle tilfældige klodser de har lyst til at vise frem. Det er *deres* klodser, der indeholder deres tanker. De har eksternaliseret deres ideer ud i klodserne og dette kan i høj grad vidne om ejerskab.

Perspektiveret til motivationsfaktorerne i den sociokulturelle teori (afsnit 11.3), søger Paula og Marillo også blive anerkendt som nogen der "kan noget" og "kan bidrage" til de andre. Det er ikke så vigtigt for dem, om en 3.klasse skal prøve deres spil, men mere om det de har lavet opleves som deltagelse, og som accepteret og værdifuldt i den sociale arena de er en del af. 3.klassen er ikke en del af deres sociale arena.

Amalie og Cecile fra 7.klasse udtrykker betydningen af at skabe noget for andre således:

I: Når nu jeres program er færdig, skal det så bruges til noget?
C: Ja..øh.. vores parallelklasse de skal prøve at spille alle vores spil og så skal de se om de kan regne det ud om de selv vil kunne overskue om de vinder eller taber..
I: Hvad tænker I om det med at nogen skal bruge jeres kasinospil?
C: Det er sjovt! At sådan vide at andre skal prøve det når man sådan selv har prøvet det virkelig mange gange for at det skulle passe..
A: … også at man selv har lavet det!
I: Det betyder noget?
C+A: Ja
I: Hvad betyder det at I selv har lavet det?
A: Det er sådan mere vores eget i stedet for bare at gå ind på en eller anden hjemmeside hvor der er spil som man ikke selv har lavet..
C: ..Så det er sådan lidt specielt..
I: Betyder det noget for den måde man arbejder på, at andre skal bruge det man laver?
C: Det har en betydning for.. Jeg tror det betyder noget for en, man er sådan lidt stolt over at man har bygget noget op.

Interviewene med Amalie og Cecile viser, at de har været udfordret langt over zonen for nærmeste udvikling. Deres færdige program er et vidnesbyrd om disse anstrengelser. At vide at andre skal prøve det man har lavet, når man selv har prøvet det "*virkelig mange gange for at det skulle passe",* kan fortælle noget om, at de er stolte af at have bygget det op selv, selv at have overkommet forhindringerne og deres udtalelser vidner om ejerskab. Når de nu har lagt alle de kræfter i det, så er det jo også væsentligt at det ikke bare forsvinder og fremstår ligegyldigt. Andre skal prøve det og det skal demonstreres om det virker.

13.2 ANALYSETEMA2: TILEGNELSE OG TÆNKNING MED VISUEL PROGRAMMERING

Dette analysetema tager sit udgangspunkt i elevernes oplevelse af at kunne forstå og tænke med visuel programmering. I dette afsnit vil jeg undersøge de hypoteser der blev fremsat i afsnit 11.2:

1) Scratchs multimediering af tekst, form og farve er en stilladserende faktor, der bidrager til at holde eleverne i zonen for nærmeste flow
2) Eleverne udvikler et særligt "scratchsprog" som støtte for deres tænkning. Sproget formes gennem oversættelse af 2.ordens programmering sprog gennem 1.ordens talesprog og tænkningen kommer til udtryk i et hybridsprog, med forankring i programmeringssprogets "kognitive stikord". Dette sproget er et udtryk for, at eleverne har tilegnet sig det visuelle programmeringssprog som akademiske begreber.

13.2.1 Hypotese 1: Stilladsering gennem redundant mediering og eksperimentering.

Dette afsnit søger at besvare første hypotese. Datanalysen peger på, at eleverne alle betjener sig af en række identiske strategier, når de skal lære Scratch og udtrykke sig og tænke med Scratch. I nedenstående citat fortæller Marillo og Paula, 5.klasse om en række forskellige egenskaber ved Scratch deres strategier hviler på.

> *I: Er Scratch et godt program at afprøve nye ideer og ting i?*
> *M: ja*
> *P: ja det er sådan, at hvis man har en ide til at lave et eller andet spil eller noget andet, så ville det nok være en god ide at starte i Scratch*
> *M: Ja i stedet for at starte i sådan en et super mega avanceret programmerings program hvor man sådan overhovedet ikke kender noget af det og hvis man så også har lavet det i skolen så kender man nogle af de funktioner der er der*
> *I: Hvad er det i Scratch der gør det godt ift at få nye ting til at virke?*
> *P: Det er ikke så svært, fordi at de der brikker de viser sådan meget hvad man skal gøre og så kan man sådan også prøve sig frem… og så er der ikke et sted hvor alle brikkerne er, det er sådan delt op i bevægelse og lyd og så det der med at man selv kan lave nogle ting nede i "data" det synes jeg også er ret fedt.*
> *I: Hvad mener du med, at der er brikker der viser hvad man skal gøre? Prøv at fortæl om det*
> *P: Ja altså – der står jo på dem hvad de gør så kan man se .. altså "gå 10 skridt" og "gentag 5 gange".. man kan ligesom læse brikkerne og tænke sig frem.*
> *I: Klodserne har jo også forskellige former… de ser forskellige ud. Bruger man det til noget når man programmerer?*
> *M: ja…altså.. Man bruger det jo i vejledningerne, så kan man jo kigge på farverne og hvordan de ser ud og så finde det hurtigt inde i Scratch..*
> *P: Det er også nogle gange man bare kan se på klodsen hvad den kan bruges til fordi der er huller i dem eller sådan noget… så ved man at der kan puttes noget i*

Der er flere ting der gør, at Marillo og Paula synes, at Scratchsproget er nemt anvende og lære. De har en forestilling om, at Scratch er anderledes i forhold til andre "super mega avancerede" programmeringssprog – fordi man med Scratch allerede "kender noget af det" i forvejen. Marillo nævner at man måske kender noget af det fra skolen, men jeg forstår også Marillos svar som at man faktisk også "kender noget til det" første gang man bruger det. Det forklarer hun med, at brikkerne (klodserne) "viser hvad man skal" – og hvis man stadigvæk er i tvivl "kan man jo bare prøve sig frem". Når Paula fortæller at "brikkerne viser hvad man skal", så hentyder hun sandsynligvis til, at brikkerne viser hvad de kan fordi "det står på dem", og på den baggrund ved man hvad man skal bruge dem til. Her læser hun klodsens funktion med sproget af 1.orden. Klodsens funktion er ikke altid entydig, så selvom Paula kan læse et ord på klodsen og hun faktisk godt forstår ordet som begreb, er der stadigvæk et stykke vej til, at hun har dannet sig et begreb om klodsens funktion og anvendelsesmuligheder.

Figur 16: Gentag indtil blok

I klodsen ovenfor kan de fleste elever godt forstå stikordet "gentag indtil". Men derfra er der et stykke vej til at "sætningen" eller klodsens betydning er gjort færdig. Det skal Paula selv tænke sig til, det er ikke givet. Paula fortæller, at en hjælp til dette afkodningsarbejde kan findes i, at man kan "se på klodsens hvad den skal bruges til". Det første hun nævner, er at man kan se at klodserne er "delt op i bevægelse og lyd", hvilket betyder, at Paula bruger farverne til at afkode klodsen funktion. Den orange "gentag indtil" klods hører således til den orange kategori: styring – altså en klods til at styre med.

Figur 17: Blokkategorier i Scratch

Hun bruger således sit sprog af 1.orden til at afkode klodsens farve og oversætte det til "styring." Paula udtalelse, fortæller os noget om at "kognitive stikord" – altså ord der guider og igangsætter kognitionen – medieres på baggrund af klodsens form. "Der er huller i dem, så ved man at der kan puttes noget i" fortæller os, at Paula er opmærksom på, at for at kunne fuldende den påbegyndte "gentag/indtil" sætning, skal der fyldes noget i. Her bruger Paula igen sproget af første orden til at afkode programmeringssproget. Enkelte elever italesætter, som Paula, klodsernes redundante mediering og analysen viser således, at der er enkelte – men ikke entydige - tegn på, eleverne faktisk anvender dette som støtte i deres oversættelsesarbejde og som støtte i deres tilegnelse og tænkning med programmeringssproget. Dette giver indikationer for at hypotesen i denne undersøgelse holder stik, da den redundante mediering fungerer som en stilladsering der har betydning for om elevernes flowtilstand.

Ifølge Vygotsky, er det i arbejdet med af afkode sådanne begreber som klodserne og kategorierne i Scratch er et udtryk for, at eleverne udvikler refleksiv bevidsthed, metakognitive kompetencer – eller som Papert udtrykker det: erkendelsesteori.

Analysen peger endvidere på, at eksperimentering en gennemgående strategi, når eleverne skal tilegne sig Scratch og i deres bestræbelser på at få deres kode til at virke. Luca, 7.kl:

> *L: Nogle gange så er det også meget godt bare at prøve jo*
> *I: Prøv at fortælle lidt om det*
> *L: Altså bare prøve at sætte nogle ting sammen, bare prøve at se om det virker jo, bare prøve at få sådan en ide om hvad der.. sådan helt præcist hvad du laver egentlig*

Elin og Petrine 5.klasse, fortæller om hvordan de ved at eksperimenterer lærer klodsernes funktion – eller *"hvad de gør for sig selv"*:

I: Når i bytter rundt på klodserne.. hvordan gør man så? Eksperimenterer man?
E: Ja, det er sådan, hvad sker hvis jeg gør sådan og hvad sker der hvis jeg gør sådan.
I: Lærer man noget af det?
P: Ja - det synes jeg i hvert fald
E: Det gør jeg også..
I: Hvorfor?
P: For så kan man også se hvad de andre klodser gør for sig selv.. hvis det giver mening?

Når Petrine her fortæller om, hvad klodsen gør for sig selv, kan det tolkes som et udtryk for, at klodsen ikke står alene. Der eksperimenteres med positionen af klodsen i kodesekvensen, for at afkode gennem eksperimetering, hvad klodsen isoleret set gør. På den måde lærer Petrine programmeringssproget begreber ved at eksperimentere.

Der er et særligt potentiale i programmeringssprogets "sandkasse metafor" (Se afsnit 6) sammenlignet med skriftsproget, hvor der ikke er samme mulighed for at inducere sig frem til hypoteser om ukendte ords funktion og betydning. Empirien belyser dette på et overordnet plan, som vi så i Elin og Petrines forklaring og der er ikke fortællinger om enkelttilfælde, der belyser selve processen omkring afkodningen af en specifik klods og hvordan et bestemt begreb bliver afkodet ved at eksperimentere.

I forhold til hypotesen, viser analysen i dette afsnit stærke indikatorer for, at Scratch multimediering af tekst, form og farve er en stilladserende faktor for elevernes tilegnelse af Scratch – hvilket afledt heraf har en betydning for om de kan befinde sig i flow.

13.2.2 Hypotese 2: Et 3. hybridsprog som støtte for kognitionen

I dette afsnit søges hypotese 2 afprøvet. Størstedelen af interviewene viser tydeligt, at eleverne selv oplever at være blevet bedre til Scratch. Enkelte elever beskriver forskelle i måden de brugte scratch på i starten og ved afslutningen af forløbet. Elin og Petrine fortæller om det, på en måde der meget godt opsummerer flere af de tendenser der ses i interviewene:

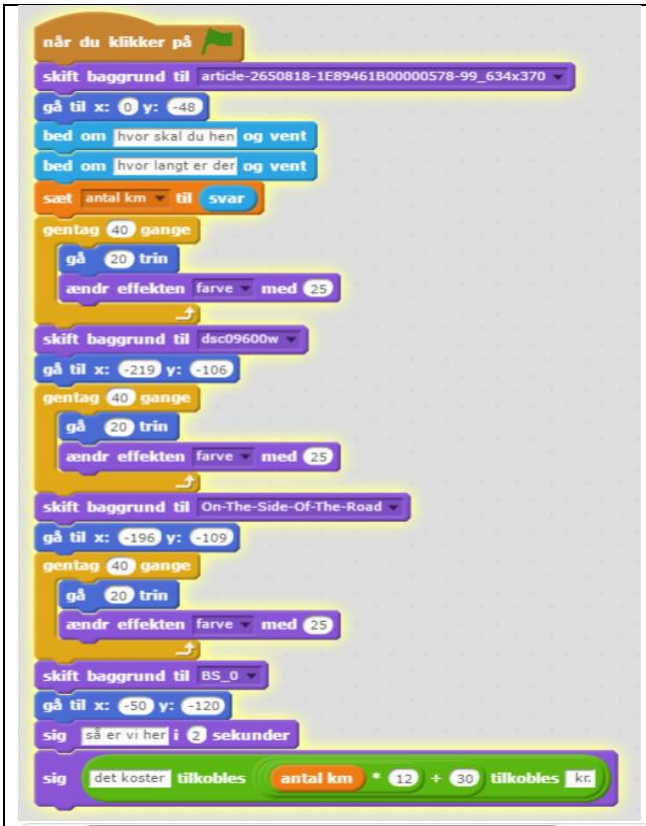
I: Vil i vurdere at I er blevet bedre til at løse problemer undervejs?
E: jeg er blevet 100% meget bedre
P: Ja.. det er jeg også
I: Hvordan ved I det? Hvordan kan I mærke det?
E: Fordi nu ved jeg - f.eks. inde i Scratch - nu ved jeg hvordan alle klodser virker, hvor før der satte jeg bare klodserne hvor jeg vil, men nu ved jeg lidt mere om hvordan det nogenlunde skal være og hvor lang tid den skal køre hvis det nu var en taxa
I: Hvad nu hvis der er noget der ikke virker er I så blevet bedre til at finde ud af hvad der kan være i vejen med det?
P+E: Ja
P: Og ellers så kikker vi på de gamle programmer og sådan noget og så tager vi derfra om vi kan bruge en måde at gøre noget på
I: Er der forskel på hvad I ville gøre, hvis I mødte et problem i starten af forløbet og så hvis i mødte det samme problem her i slutningen af forløbet?
P+E: Ja
I: Hvad tænker I, at I at I er blevet bedre til i forhold til at løse udfordringer? Ud over at I selvfølgelig kender klodserne bedre nu?
E: Man skal tænke rigtigt meget over, selvfølgelig, alle måder og metoder og sådan noget .. f.eks før, hvis jeg havde et problem så gik jeg helt amok og gad ikke at lave det for så gav jeg op med det samme, så skal man ligesom bare fortsætte hele tiden.. øhmm ja
I: Når du siger man skal fortsætte.. det kan jo være svært hvis man ikke ved hvad der er i vejen?
E: Man kan jo starte meget enkelt, det her med "når du trykker på flag" så går scriptet i gang eller sætte alle sine sprites ind så det er på plads, så kommer det egentlig efterhånden af sig selv, tror jeg - det gjorde det i hvertfald for os sådan at det nogenlunde skulle virke

Elin beskriver for det første, hvordan hun i starten af forløbet med Scratch mest ”sætter klodserne” hvor hun vil. Dette vidner om en ”trial and error” tilgang som strategi for, hvordan man ”oversætter” klodsernes funktion. Dette vidner om, at hun i begrænset omfang har støttet sig til de sproglige logiske sætningskonstruktioner som kodesekvenser i Scratch kan danne, ved at forbinde klodsernes sekventielle struktur med sprog af 1.orden. Når hun beskriver, hvordan hun har ændret denne strategi siger hun, at hun ikke længere ”bare sætter klodserne”, men at hun ”ved nogenlunde hvordan det skal være”. Hun har altså iagttaget en ændring i hendes afkodnings- og problemløsningsstrategi, hvilket indikerer metakognitive overvejelser. Hendes strategi i slutningen af forløbet har ændret sig til at ”*starte meget enkelt”,* hvilket betyder at starte fra toppen af kodesekvensen og *læse* koden højt *“når du ”trykker på flag” så går scriptet i gang…”.* Herefter forklarer hun at ”*så kommer det egentlig af sig selv”.* Dette fortæller hun *uden* at støtte sig til en skærm, hvor hun kan se nogen programmeringsklodser. Det er nærliggende at tro, at hun derfor visualiserer startklodsen på scriptet for sig:

En mulig tolkning er, at når hun fortæller om hendes nye strategi med at tage det roligt, tænke sig om, og læse teksten på klodserne, så handler det om, at Elin afkoder klodserne og deres sammenhæng sekventielt i et script gennem logiske, sammenhængende sætninger af 1.ordens sprog. Der er grund til at tro, at hun også programmerer ved at tænke i disse sætningskonstruktioner baseret på Scratchs ”kognitive stikord” og at hun som fejlsøgningsstrategi vurderer om meningen i ”scratchsætningerne” af 1.ordenssprog, svarer til den mening hun gerne vil have scriptet til at eksekvere. Sidstnævnte forhold omkring fejlsøgningsstrategier og ”scratchsætninger” har ikke været muligt finde tegn på i empirien.

Jeg forsøger herunder at udfolde denne hypotese yderligere.

Det er ikke vanskeligt for eleverne efter forløbet, at oversætte deres egne kodesekvenser til 1.ordens sprog. Herunder bliver Valde og Luca, 5.kl spurgt om de forstår deres egen kode, og om de kan forklare hvad der sker. I transskriptionen har jeg markeret de ”kognitive stikord” som de oversætter og giver mening gennem deres 1.ordenssprog. Valde og Luca kigger på koden mens de forklarer:



V: Jamen altså.. når man klikker på flaget så går bilen til et bestemt sted på skærmen, og så beder den om "hvor skal du hen" og hvor langt der er.

Så sætter den variablen antal kilometer til hvor langt der er - altså det man har svaret – og så gentager den "gå 20 trin" mens den skifter farve og går tilbage til udgangspunktet, mens den hver gang skifter baggrund for at vise at taxaen kører en lang tur igennem 3 forskellige baggrunde.

Så til sidst skifter den baggrund til en anden en end den, den starter på …

V: Så hernede..
L: Der regner den det hele ud…
V: Ja.. så siger den "Så er vi her" og så siger den "det koster" antal kilometer gange 12 kroner plus 30

I: Hvad betyder alle de tal der?
V: Det er fordi at det jo koster 12 kroner pr kilometer og startgebyret er 30 kroner. Så hvis der er 50 kilometer så ganger den det med 12 og plusser 30. Så siger den...se her (tænder koden)
L: "Så er vi her" og så "Det koster 630 kroner"
I: Okay..

Der er begrænset empiri, der viser, at eleverne ikke blot afkoder med deres 1.ordenssprog, men at de også gør indre meningsfulde "scratchsætninger" til ydre programkode når de programmerer. En mulig forklaring skyldes metoden, hvor elevernes tanker og sprog i arbejdsprocessen lang bedre ville komme til udtryk i et observationsstudie i praksis mens det udfolder sig. Men hvis man drager analogien med tilegnelsen af skriftsproget, hvor man lærer at læse(kode) ved at skrive(kode) og man lærer at skrive(kode) ved at læse(kode), vil en sandsynlig hypotese være, at dette forhold også gør sig gældende i forhold til tilegnelsen af det visuelle programmeringssprog. Jeg har vist små tegn på at dette sker hos Elin der visualiserer klodsen med startflaget for sig i tankerne, "Når jeg klikker på flag" og fortæller at "så kommer det af sig selv" – underforstået resten af den meningsfulde sætning kommer "af sig selv", når scriptet efterhånden bliver sat sammen til meningsfulde ytringer af 1.ordenssprog – på samme måde som Valde og Lucas oversættelse nærmest "kommer af sig selv".

Der ses også små tegn på det, da Emil og Tobias fortæller om et problem med koordineringen af 2 indbyrdes afhængige scripts. Her fortæller de, med kodesproget på et tankeplan, *uden* selv at kunne se deres kode. Igen er de "kogntive stikord" i scratch understreget.

SCRIPT FOR BURGER  SCRIPT FOR STJERNE  	T: Altså det var nok lidt svært med at få den til at rykke rundt og sætte det ind fordi .. så blev den bare sat til 48.000 eller sådan noget.. det var mega mærkeligt. I: Så det var en af de store udfordringer I havde. Hvordan fik I det så løst? E: Ved at sådan tænke os om på en måde. T: Vi prøvede os også lidt frem I: Hvad havde I så gjort galt? Er I egentlig med på det? Altså, hvad I havde gjort galt til at starte med? **(taler om script for burger)** E: Ja, det var når … hvis den der burger berørte stjernen så i de sekunder den berørte så skulle den ændre pointene med 1 **(taler om script for stjerne)** og der skulle stjernen ikke have vist sig I så lang tid, fordi at hvis den viste sig i 5 sekunder blev det op i 48.000 eller sådan noget T: Jaehh I: Så hvordan løste I det? E: ja altså vi tænkte os om, at så skulle den nok skulle vises i mindre tid - i 2 sekunder eller sådan noget **(taler om script for burger)** T: ja, og tælle point langsommere så den ikke kom op på 40.000 eller et eller andet vildt. Så vi fandt ud af at den skulle vente i mere end den anden bliver vist.. agtigt… altså mere end 2 sekunder I: Okay…godt.

Det er tydeligt, at Emil og Tobias "tænker med klodserne" selv om de ikke kan se dem. Måske visualiserer de dem på samme måde som Elin. Deres sprog er "farvet" af programmeringssproget,

hvilket kommer tydeligst til udtryk når Emil korrigerer hans eget talesprog: ”Ja det var når… *hvis* den der burger…”. Emil ændrer ”når” til ”hvis”, fordi ”hvis” er en af de ”kognitive stikord” der støtter Emil i at oversætte programmeringssproget til 1.ordenssprog. Han kunne også rent talesprogligt have brugt ”når”, men korrigerer det til ”hvis” for at hente støtte i programmeringssproget til at kunne udtrykke sin tænkning præcist.

Især når eleverne skal forklare deres egen scratchkode, men også når de i glimt fortæller om hvordan de skaber kode ved at ”tænke kode”, så viser undersøgelsen, at de anvender et særligt ”scratchsprog”. Dette tredje sprog er anderledes end deres talesprog eller skriftsprog som vi også ser det hos Emil, Tobias, Valde og Luca. Det består af logiske sætninger, der har ankerpunkter i et Scratch vokabular og som til tider kun bærer reel mening i Scratch konteksten. F.eks. når Valde siger*: ”så går bilen til et bestemt sted..”* så giver kun reel mening, fordi sætningen refererer til scratchklodsen ”gå” (biler ”går” ikke, de kører) eller når Emil korrigerer sig selv og erstatter ”når” med ”hvis”.

På baggrund af disse eksempler, har jeg undersøgt hypotesen om, at eleverne internaliserer programmeringssproget af 2.orden ved at anvende deres 1.ordens sprog til at afkode Scratch redundante mediering af ”kognitve stikord”, former og farver. På den baggrund skaber de et hybrid sprog, et scratchsprog eller et 3.sprog. Her bindes 2.ordenssproget sammen af 1.ordenssprog og skaber i den særlige scratchkontekst meningsbærende, sproglige ytringer(3.sprog), der støtter eleverne i at programmere og tænkes med kode.

13.3 ANALYSETEMA3: TEGN PÅ LÆRING

I analysetema 2, blev der beskrevet, hvordan eleverne afkoder klodsernes tekst, form og farve, og binder disse kognitive stikord sammen i et nyt scratchsprog (3.sprog), der bruges til at tænke og programmere med. Et programmeringssprogs funktioner og begreber og tilegnelsen af dette, er i følge Vygotsky at kategorisere som ”akademiske begreber”. Det betyder, at det er gennem arbejdet med at tilegne sig disse begreber – og den problemløsning der er involveret i processen - at disse bliver tænkeredskaber, der muliggør tænkning på et abstrakt, formelt plan og som gør, at eleverne udvikler kompetencer til at tænke kreativt og problemløsende med programmeringssproget. Det er altså gennem internalisering af programmeringssprogets ”akademiske begreber” og den implicerede problemløsning, at eleverne udvikler metakognitive evner – heriblandt bevidste problemløsningsstrategier (se afsnit 11.2.3). Dette gælder selvfølgelig også arbejdet med de matematiske begreber: Variable, koordinatsystem og kombinatorisk sandsynlighed.

I det følgende vil jeg undersøge hvorvidt et undervisingsforløb bygget på CAS Barefoots didaktiske CT principper indikerer, at eleverne udvikler metakognitive problemlæsningsstategier i form af CT strategier.

13.3.1 CT strategi: Dekomposition

Eleverne er blevet stillet overfor udfordringer, hvor de kan have gjort erfaringer med at udvikle strategier i dekomposition på forskellige niveauer. FIRE modellen er en ramme der strukturelt har indeholder dekomposition. I R fasen (Realisering) skulle eleverne *inden* de gik i gang med at programmere, udvikle skitser eller tegninger (prototype), hvor de beskriver enkeltdelene i deres idé. På den måde bryder de deres overordnede idé ned enkelte elementer, der hver skal programmeres og spille sammen med helheden. Interviewene viser tydelige tegn på at begge klasser arbejdede overfladisk med denne fase. Der er ikke noget der indikerer, at det har haft voldsom betydning for deres måde at arbejde eller udvikle strategier på. Dette kan skyldes flere ting. Måske fordi man har været ivrig efter at komme i gang eller fordi denne designproces er uvant for klassen og

underviseren. Det kan også skyldes tidspres. Eleverne havde halvanden uge til at arbejde i designprocessen. Dette ER meget kort tid, og det er et væsentligt kritikpunkt i det didaktiske design og der bør foretages ændringer i forhold til dette i næste iteration af designet. I nedenstående udtalelser fremgår hvad der er generelt i empirien:

Cecile 7.kl:

> *I: Lavede I skitser af jeres ideer inden i gik igang?*
> *C: nej ikke rigtigt, men vi kiggede på andre programmer i Scratch for at finde ud af hvordan vi kunne lave det og så kunne man prøve at sammenligne det lidt så fik man ligesom sit eget spil ud af det*

Elin og Petrine 5.kl

> *I: Hvordan kom I på jeres idé?*
> *E: Øhm ja altså, jeg tænkte noget med jakker og noget med noget tøj..*
> *P: Ja og jeg tænkte sådan lidt på noget med træer*
> *E: Ja.. Så vi lavede "træjak".. så jakkerne hænger i træerne*
> *I: lavede I en skitse af jeres idé inden i gik i gang?*
> *P: Nej.. faktiske ikke. Vi gik bare i gang.*
> *I: Så I lagde ikke en plan først?*
> *P: Nej..*
> *I: Tror I det ville have hjulpet jer at have lavet en plan?*
> *E: Det ville det jo have været så vi ville have fået en lille idé om hvad vi ville lave .. øhhm, men altså, det GIK jo også godt sådan på den den måde at vi bare gik i gang med det samme.*

Selvom eleverne ikke har lagt noget særligt i at lave skitser og planer inden de gik i gang og på den måde bryde deres ide op i delelementer/problemer – så har de alligevel undervejs været nødt til at være opmærksomme på de enkelte delelementer og funktioner i deres kode. Når der er opstået problemer – hvilket der er hos *alle* eleverne i interviewene – har de haft dekompositionsstrategien i spil, for overhovedet at kunne identificere hvilket "område" eller hvilken funktion er deres program der var problemfyldt. Med andre ord: De vil ikke være i stand til at identificere og derved løse problemer, med mindre de betjener sig af en eller anden form for dekomposition. Spørgsmålet er ikke *om* det er sket, blot i hvilken grad og med hvilken kvalitet dette er sket og om eleverne er bevidste om det og kan anvende det som en måde at tilgå andre problemer på. Undersøgelsen må konkludere at på dette overordnede plan, har eleverne gjort erfaring med dekomposition, men empirien fortæller ikke mere om, hvor konsoliderede disse strategier er og om der er transfer til andre sammenhænge.

På et mikroniveau – altså i selve det at kode – er dekomposition en strategi som de fleste elever helt tydeligt har udviklet i forbindelse med at de fejlsøger/debugger deres kode eller løser et problem. Dette kommer til udtryk ved, at eleverne "skiller deres kode ad" i enkeltdele eller "arbejder 1 trin ad gangen", og enten samler koden på ny eller løber koden igennem, for at kunne identificere eller udskille de enkelte områder i deres scripts med der formål at identificerer et eller flere problemer:

Luca og Mikkel, 7.klasse fortæller om, hvordan det kan være en god strategi, at identificere det der virker og skille det fra det der ikke virker:

> *I: Hvis man så afprøver nogle ting der så ikke virker… hvordan gør man så?*
> *L: ..Så er det bare at prøve igen jo..*
> *I: Ja.. hvordan gør man det?*
> *L: Enten så beholder man noget af det der godt virker og så kigger på det andet der ikke gør, og så prøver du at lave om på det*
> *M: altså sådan justere det agtigt og lave om på det..*

Emil og Tobias 5.klasse fortæller her om deres strategi med ”at skille det ad”, for at identificere og udskille problemer:

> *I: Har I haft brug for at skille jeres kode ad for at forstå hvad I laver?*
> *T: Ja*
> *E: Det har vi haft…*
> *I: Kan I prøve at fortælle lidt om det?*
> *E: Nogle gange når vi var gået i stå på en måde, så blev vi nødt til lige at skille det ad og så lige kigge… hvad har vi her?*
> *T: Ja..*

13.3.2 CT strategi: Mønstergenkendelse

Undersøgelsen viser, at alle de interviewede elevpar på forskellige måder, har anvendt noget der minder om mønstergenkendelse som problemløsningsstrategi. Eleverne beskriver, hvordan de genbruger løsninger og scripts som de kender fra tidligere. Her kigger de enten i de udarbejdede vejledninger, eller også kigger de i nogle af de programmer de tidligere har lavet. Enkelte af eleverne finder også programmer og scripts der ”kan det de har brug for” ved at søge i andre scratchbrugeres programmer inde på Scratch hjemmesiden.

Elin og Petrine 5.klasse og Amalie og Cecile, 7.klasse fortæller her om, hvordan de leder efter ”klodser” i andre programmer de tidligere har lavet, for at finde mønstre de kan bruge til at løse deres deres aktuelle udfordring:

Elin og Petrine, 5.klasse:

> *I: Hvad nu hvis der er noget der ikke virker er I så blevet bedre til at finde ud af hvad der kan være i vejen med det?*
> *P+E: Ja*
> *P: Og ellers så kikker vi på de gamle programmer og sådan noget og så tager vi derfra om vi kan bruge en måde at gøre noget på*

Amalie og Cecile, 7.klasse:

> *I: Var det svært at arbejde med de opgaver i vejledningerne?*
> *A: Det var nemmere, end at vi sådan selv skulle finde ud af det*
> *I: Var opgaverne en hjælp for jer?*
> *A: Ja*
> *C: .. meget*
> *I: Hvordan hjalp det jer?*
> *A: Nogle gange gik vi sådan tilbage i hæftet for at kigge hvordan det nu var, og andre gange kigger vi bare i de programmer vi havde lavet før*

Valde 7.klasse, fortæller at en god strategi også er lede efter mønstre i andre scratchbrugeres programmer:

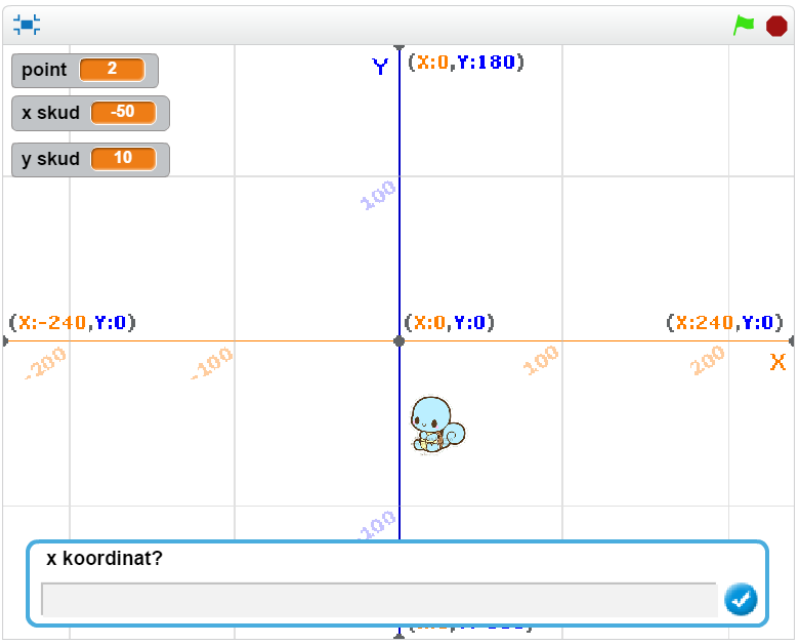
> *V: Og hvis det var at man sådan ikke kunne finde ud af det, så kunne man prøve at finde nogle projekter på scratch der lignede en lille smule og så få nogle ideer fra det*

Empirien giver indikatorer for, at eleverne selv mener, de er blevet bedre til at problemløse ved at finde ”mønstre” til i andre programmer eller ressourcer. Det er vanskeligt at forklare om dette primært skyldes, at de blot har fået større kendskab til ”scratch universets” enorme bibliotek af bugerskabte programmer eller eleverne også vil være mere tilbøjelige til at anvende denne strategi i andre sammenhænge. På baggrund af data kan der ikke konkluderes andet, at der har gjort erfaringer med denne strategi og de har oplevet det som en god måde at problemløse på.

13.3.3 CT strategi: Abstraktion

Abstraktion som problemløsningsstrategi, handler om at kunne fokusere på det væsentlige og udelade uvæsentlige detaljer. At kunne udvælge de hensigtsmæssige foki, elementer og data, og vælge en repræsentationsform, der bedst mediere dette. Den kontekst eleverne opererer i, er at de skal skabe hhv et læringspil om koordinatsystemet og et casinospil til en spilledag på skolen. Grupperne i 5.klassen har alle formået at stille skarpt på de væsentlige elementer i deres læringsspil i deres bestræbelser på, at skulle lære andre om koordinatsystemet. Dette kommer til udtryk i, at spillene alle handler om at lære koordinatpunkter i et koordinatsystem, og at det for de fleste elevers vedkommende er repræsenteret v.h.a et koordinatsystem.

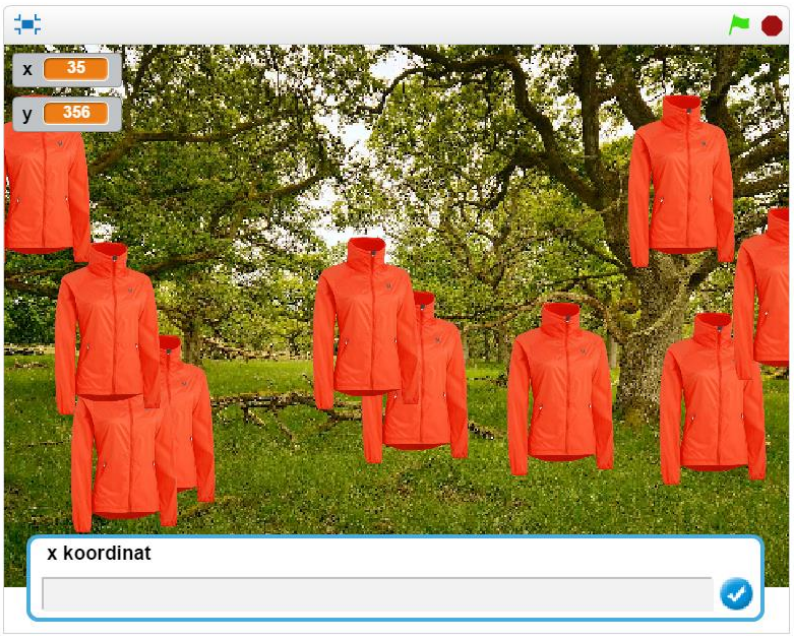
Her er Marillo og Paula, 5.klasse bud på et læringsspil:



Figur 18: Scratch programmet "Pokimon"

Spillet ”Pokimon” går ud, på at kaste en Pokeball hen på en Pokemon figur, der tilfældigt dukker op på et koordinatsystem. Rammer Pokeball’en figuren fanger man Pokemon figuren og man får point. Man kan kun ramme ved siden af 5 gange, så er det Game Over. Man skal skrive x-koordinater og y-koordinater ind for at kaste Pokeball’en. Dette spil er et udtryk for en abstraktion. Spillet skal simulere en Pokemon jagt, hvor brugeren skal opøve sikkerhed i brugen af koordinatpunkter. Marillo og Paula har lavet en vellykket abstraktion af en Pokemon jagt. Alle unødvendige detaljer som Pokemon figurernes navne, det karakterunivers de stammer fra, dialoger som eleverne kender fra tegnefilmene osv. er fravalgt.

Det er slet ikke alle elevernes produktioner der er et udtryk for en vellykket abstraktion. Petrine og Elin har skabt spillet ”træjak”, som handler om at trylle alle jakker væk med en tryllestav, der åbenbart hænger i nogle træer(!):



Figur 19: Scratch programmet "Træjak"

Egentlig minder Petrine og Elins spil en del om ”Pokimon” spillet. Men i forhold til at fokusere på det væsentlige – altså foretage en abstraktion – så er træjak, væsentligt mindre vellykket. Det væsentlige er, at vælge repræsentation og indhold, der modsvarer formålet. Men brugerne af dette spil lærer nok meget lidt om koordinatsystemet, fordi der ikke er noget koordinatsystem. Hvordan skal man vide hvad enhederne er på x og y aksen?

På samme måde som vist i eksemplerne ”Pokimon” og ”Træjak”, er det generelle billede i begge klasser, at der er stor forskel på kvaliteten i elevernes abstraktioner. Ud fra empirien er det vanskeligt sige noget præcist om, om eleverne er bevidste om, at abstraktion er en måde at målrette og løse problemer på. Eleverne italesætter ikke dette. I sidste ende er det uden tvivl særdeles væsentligt, at kvaliteten af elevernes abstraktioner bliver italesat og drøftet løbende i klassefællesskabet. F.eks. synes Elin og Petrine selv, at de havde lavet et fantastisk spil:

> *I: Hvordan gik det med at fremlægge jeres program for resten af klassen undervejs?*
> *E: Det gik jo godt.. folk var rimeligt overraskede over vi havde lavet sådan noget "træjak" (griner).. men de synes også det var underholdende for det her det var sådan meget med at tænke ud af boksen*
> *P: ja.. så folk var jo egentlig ret imponerede*

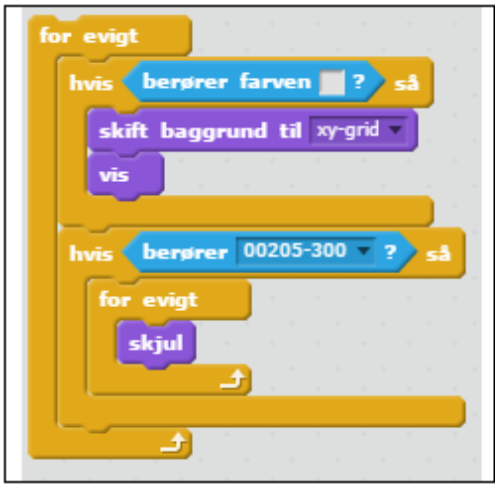
Det at eleverne forholder sig kritisk til deres abstraktioner, vil være et væsentligt fokuspunkt i næste iteration af designet. Empirien fortæller os i hvert fald, at dette ikke i tilstrækkelig grad har været i spil i forløbet og at dette fokus ikke har spillet nogen særlig rolle i forløbets løbende procesevaluering.

13.3.4 CT strategi: Algoritmer

I en programmeringskontekst handler algoritmer om, at udvikle ”trin-for-trin” scripts, der kan udføre en bestemt handling. Min kodning af interviewdata viser, at eleverne selv har udviklet algoritmer og genbrugt, tilpasset og justeret andres algoritmer, når de leder efter mønstre. Algoritmisk tænkning som problemløsningsstrategi, handler om, at kunne tænke ”trin-for-trin” for at kunne lave en generelt opskrift, der kan bruges i specifikke sammenhænge. Som tidligere vist, er denne ”trin-for-trin” tænkning helt fundamental, når man programmerer og i processen med at udvikle et hybrid sprog mellem 1.orden og 2.ordens sprog som beskrevet i afsnit 13.2.2. Eleverne fortæller gentagne

gange om strategien med at starte fra toppen og gennemgå "opskriften". Nogle af disse generelle algoritmer, har "spredt" sig i klassefællesskabet og bliver brugt til at løse ens udfordringer.

Her beskriver Valde og Luca, hvordan et script for "kollision mellem flere sprites" fungerer som en algoritme for flere af eleverne i klassefælleskabet:



I: Hvordan fik I løst udfordringen med at få husene til at forsvinde? Hvordan fandt I ud af at det var "for evigt" klodsen der manglede.
V: Det var fordi på de andres scratch projekter, så hvis der stod sådan en "hvis" klods så skulle man sætte en "for evigt" uden om, for ellers så virker det jo kun 1 gang.
I: Nu fandt i ud af det ved at se på hvordan de andre havde gjort, tog I så bare klodsen til jer eller forstod I hvad klodsen gjorde og hvorfor jeres program ikke fungerede uden?
L: Ja, vi forstår det godt…
V: Den gjorde sådan så huset det skjulte sig HELE tiden
..
L:... og at det ikke bare poppede op igen

Denne elevudviklede opskrift eller algoritme, har været i spil i flere af elevernes projekter og vidner om at eleverne både udfører algoritmisk tænkning som problemløsningsstrategi, og selv skaber algortimer, som de kan bruge på tværs af programmer.

Lignende algoritmer kan ses på tværs af projekter i klasserne. F.eks. scriptet der spørger om x og y koordinater i 5.klassernes projekter som vi så det i spillene "Pokimon" og "Træjak".

14 ANBEFALINGER PÅ BAGGRUND AF LÆRERINTERVIEWS

Lærerne havde få konkrete anbefalinger til forbedringer af det didaktiske design. De udtrykte helt overordnet, at de enkelte elementer i modellen havde fungeret efter hensigten og oplevede det desuden, som relevant at arbejde med designprocesser i matematik. Det er ikke formålet, at beskrive det her. De udfordringer de havde oplevet, var ikke i særlig høj grad knyttet til designet som sådan, men til andre faktorer som læringskulturen, vaner for elev og lærerroller, elevernes opfattelse af matematik og andre faktorer.

Dog havde lærerne ikke oplevet de fastlagte evalueringsloops som tilstrækkelige, i forhold til at sprede viden og ideer i klassefællesskabet. At evalueringsloopsne var fastlagt på bestemte tidspunkter, fungerede ikke som en tilstrækkelig effektiv stilladsering for eleverne:

> *De kunne simpelthen ikke vente. Vi prøver at tage 10 minutter og det havde de svært ved at vente på. Vi vil bare gerne videre nu. Jeg kan ikke sidde og vente på at jeg får hjælp af min anden gruppe. Nogen gange overholdt vi de der deadlines og andre gange blev vi simpelthen nødt til at skyde dem. Så skrev jeg præcis på tavlen, hvornår vi skulle gøre det. Vi gør det her kl. 11.10 og det her kl. 11.30. Men de ville gøre det her og nu. For hvad skulle de lave i 20 minutter.*

Lærernes egne forslag til forbedringer på dette område går på, at foretage evalueringsloops ad-hoc, når lærerne kunne "fornemme" at der var et behov for enten:

1) at løse et fælles oplevet problem

2) at klassefællesskabet kunne hjælpe med en gruppes konkrete udfordring, når læreren ikke kunne hjælpe, eller
3) der var en gruppe, der havde lavet noget læreren vurderede som værdifuldt, at en gruppe delte med resten af klassefællesskabet.

Anbefalingen går også på, at det ikke var nødvendigt, at *alle* elevgrupperne nødvendigvis skulle formidle deres work-in-progress. Her var anbefaling også en ad-hoc-tilgang. Det blev oplevet som tidskrævende i forhold til en alt for stram tidsplan. Mere tid er også en helt tydelig anbefaling til næste iteration af den åbne designsekvens.

> *Man stiller det op med at sige, at vi har en uge, så lyder det også i starten rigtig lang tid for dem "en hel uge på det her" og da vi så næsten var halvvejs, sagde de "vi har jo ikke nået noget!". Så hvad er en hel uge?*

Ovenstående anbefalinger, bør overvejes i næste iteration af interventionsdesignet.

15 KONKLUSION

Hvordan kan man anvende et didaktisk design med visuel programmering i folkeskolens matematikundervisning med henblik på at støtte elevernes ejerskab til læreprocesser, deres forståelse af programmering samt udvikle deres problemløsningsstrategier?

Gennem en DBR metodologi, er der udviklet et didaktisk design for visuel programmering i matematik. Det didaktiske design bygger på et didaktisk rammedesign med lukkede og åbne sekvenser. De lukkede sekvenser kobler centrale matematikfaglige begreber med programmeringssproget Scratch gennem opgaver der bygger på en progression fra worked-examples mod åbne udfordringsopgaver. Målet var at ruste eleverne til at lykkes med deres eget designprojekt. Undersøgelsen viste, at denne del af designet fungerede efter hensigten.

Den åbne sekvens bygger på FIRE modellen som designframework, der guider designprocessen. Det var forskelligt, hvor godt designprocessen forløb i de to klasser. 7.klassen oplevede designprocessen mere frustrerende end 5.klassen. Det kan forklares ud fra fravær af deres matematiklærer og for kort tid til at arbejde i denne fase. Intentionen bag FIRE modellens idegenereringsfase fungerede ikke optimalt for 7.klassen, da undersøgelsen viser, eleverne ikke tillagde den nogen særlig betydning. Forklaringen kan være tidspres, fravær eller manglende erfaring i at arbejde med designprocesser. Dette bør – samme med det oplevede tidspres - medtages som fokuspunkt i næste iteration.

Den åbne designsekvens inddrog også løbende evaluering i form af læringsloops. Særligt evalueringsloopsne fungerede ikke tilfredsstillende som stilladsering for elevernes læreproces, og næste iteration bør overveje at designe for en ad-hoc tilgang til disse.

Underviserne, der ikke før forløbet havde programmeret i undervisningen, udtrykker koblingen af designprocesser, programmering og matematik som værdifuld. Designinterventionen har på den måde formået, at knytte matematik og programmering meningsfuldt sammen.

Undersøgelsen bekræfter beskrivelser af, at programmering pr definition er udfordrende. I forhold til elevernes oplevelse af ejerskab til læreprocesser, er der indikatorer for, at selve det at overkomme programmeringstekniske udfordringer, er med til at udvikle ejerskab til læring. Undersøgelsen viser tegn på, at eleverne støtter sig til Scratch redundante multimediering af stilladserende symboler, og at selve programmeringsinterfacet i Scratch er medvirkende til at eleverne overkommer

udfordringer og derfor udvikler ejerskab. Der er endvidere tegn på, at det er betydningsfuldt for elevernes ejerskab til læring, at det de skaber, skal fremvises for andre og at elevernes ønske om at blive anerkendt for det man har lavet i klassefællesskabet, spiller en rolle for udvikling af ejerskab.

I forhold til DBR metodologiens ønske kun at forbedre praksis, men også at generere domænespecifik forståelsesviden, har forskningsprojektet været optaget af, at forstå, hvordan eleverne tænker og lærer med Scratch og hvordan det kommer til udtryk. Forskningsprojektet har med empirisk forankring fremsat hypotesen om, at eleverne udvikler et stærkt kontekst afhængigt, logisk opbygget 3. sprog. Dette sprog fungerer som et hybridsprog mellem programmeringssproget og talesproget og støtter eleverne i at programmere og tænke med kode. Yderligere iterationer er påkrævet for at styrke robustheden af hypotesen og for en dybere forståelse af dette sprogs rolle for kognitionen.

Specialet har belyst sammenhængen mellem undervisning med CT designprincipper og udvikling af CT strategier. Undersøgelsen viser, at eleverne anvender forskellige former for dekomposition, mønstergenkendelse, abstraktion og algoritmer når de programmerer, men analysen kan ikke svare på, om eleverne har *udviklet* deres problemløsningsstrategier. Undersøgelsen viser, at eleverne oplever at være blevet bedre til at problemløse i Scratch, men det er ikke ensbetydende med en generelt forbedret problemløsningskompetence i anden matematikundervisning eller andre kontekster.

En del af forklaringen på disse resultater, kan måske hænge sammen, at interventionen kun forløb over 3-4 uger og at man kan formode, at observérbare ændringer i problembehandlingskompetencer, tager længere tid at udvikle. Derfor anbefales uddybende forskning. Følgende iterationer af interventionsdesignet kan i forhold til DBRs ønske om teoriproduktion af designmetodik, lade sig inspirere af den kvantitative forskningstilgang i forskningsprojektet ”Developing Mathematical Thinking With Scratch” (Calao, Moreno-Leon, Correa, & Robles, 2015) som beskrevet i forskningsreviewet i afsnit 5, for muligvis at komme tættere på en kortlægning af elevernes udvikling af problembehandlingskompetence.

16 DISKUSSION OG PERSPEKTIVERING

DBR metodologien ønske er, at udvikle robuste designs. Dette forskningsprojekt har med en pragmatisk og fortolkende tilgang undersøgt enkelttilfældet og gået detektivisk til værks i analysen. Der er foretaget én iteration. Dette bærer undersøgelsens resultater også præg af. Der er ikke fundet mange endegyldige sandheder, men derimod observeret en række tegn og hypoteser om sammenhænge, der kunne være interessante at udfordre. Mange af disse er beskrevet i konklusionens forslag til fokuspunkter til næste iteration. Disse tegn og hypoteser, er udsprunget af det kvalitative interview som metode. Det har haft den ulempe i dette projekt, at data om hvordan man tænker og løser problemer, er blevet indhentet på retrospektivt grundlag. Informanterne har skulle tænke tilbage på hvordan de tænkte. Dette er vanskeligt for voksne – men nok endnu mere vanskeligt for børn. Andre kvalitative metoder der bedre kunne opfange tænkning og problemløsning ”in-situ”, ville uden tvivl have været berigende for dette projekt, og have leveret data, der ville danne mere robuste designs og hypoteser. Her tænker jeg specielt på observationsstudiet som metode.

I forhold til Misfeldts forestillede læringsvej, har det været en udfordring at holde mit domænekendskab og personlig erfaring med visuel programmering ude af projektet – uanset om disse antagelser var underbygget af teori eller ej. Dette har vist sig i især interviewene i form af

ledende spørgsmål, men også i kodningen af data. Den pragmatiske induktive ”bottom – up” tilgang til videnskabelse, har til tider været udfordret.

Et interessant perspektiv er, hvad programmering betyder for elevernes måde at se sig selv som lærende på? Et perspektiv der ikke er inkluderet i dette speciale, men som alligevel fremgår af empirien, er elevernes vedholdenhed når de løser udfordringer i Scratch. Har dette betydning for en øget tiltro til egne evner og forventninger til sig selv? Og om en sådan ændring i opfattelsen af sig selv som lærende har transfer ud over programmeringsaktiviteter?

17 REFERENCER

Ackermann, E. (1. September 2001). *Piaget's Constructivism, Papert's Constructionism: What's the difference?* Hentet 20. April 2017 fra CiteSeer: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.4253

Adams Becker, S, Cummins, M., Freeman, A., & Rose, K. (2017). *2017 NMC Technology Outlook for Nordic Schools.* Austin, Texas: The New Media Consortium.

Andersen, F. Ø. (2006). *Flow og fordybelse : virkelystens og det gode livs psykologi.* København: Hans Reitzel.

Bang, J., & Dalsgaard, C. (2005). Samarbejde – kooperation eller kollaboration? *Tidsskrift for universiteternes efter- og videreuddannelse*(2. årgang, nr. 5).

Barnes, R. (Red.). (2017). #Insights: Learning through making. (1), s. 18-21. Hentet fra Hello World: https://helloworld.raspberrypi.org/helloworld

Basawapatna, A. R., Repenning, A., Koh, K. H., & Nickerson, H. (2013). The zones of proximal flow: guiding students through a space of computational thinking skills and challenges. *Proceeding: ICER '13 Proceedings of the ninth annual international ACM conference on International computing education research*, 67-74.

BBC Bitesize. (2017). *What is Computational Thinking*. Hentet 2. April 2017 fra BBC Bitesize: http://www.bbc.co.uk/education/guides/zp92mp3/revision/1

Braun, V., & Clarke, V. (6. Juni 2006). Using thematic analysis in psychology. *Qualitative Research in Psychology* , s. 77-101.

Brennan, K. A. (2013). *Best of both worlds: Issues of structure and agency in computational creation, in and out of school.* Hentet fra http://web.media.mit.edu/~kbrennan/files/dissertation/Brennan_Dissertation.pdf

Brinkkjær, U., & Høyen, M. (2011). *Videnskabsteroi for de pædagogisk professionsuddannelser.* København: Hans Reitzels Forlag.

Brinkmann, S. (2006). *John Dewey : en introduktion.* København: Hans Reitzels Forlag.

Brinkmann, S., & Tangaard, L. (2015). *Kvalitative metoder* (2. udg.). Hans Reitzels Forlag.

Bråten, I. (2006). *Vygotsky i pædagogikken.* København: Frydenlund.

Calao, L. A., Moreno-Leon, J., Correa, H. E., & Robles, G. (2015). *Developing Mathematical Thinking with Scratch: An Experiment with 6th Grade Students.* Hentet fra http://jemole.me/replication/2015ectel/CodeMath_Draft.pdf

CAS Barefoot - CT Approaches. (2014). Hentet 23. Marts 2017 fra http://barefootcas.org.uk/barefoot-primary-computing-resources/computational-thinking-approaches/

CAS Barefoot. (2014). Hentet 23. Marts 2017 fra http://barefootcas.org.uk/

Christensen, O., Gynther, K., & Petersen, T. B. (2012). Design Based Research - introduktion til en forskningsmetode i udvikling af E-læringskoncepter og didaktisk design medieret af digitale

teknologier. *Læring & Medier (LOM)*(9). Hentet fra http://ojs.statsbiblioteket.dk/index.php/lom/issue/view/540

Dale, E. L. (2006). Læring og udvikling - i leg og undervisning. I I. Bråten, *Vygotsky i pædagogikken* (s. 47-82). København: Frydenlund.

Dysthe, O. (2003). *Dialog, samspil og læring.* Århus: Forlaget Klim.

Egholm, L. (2014). *Videnskabsteori : Perspektiver på organisationer og samfund.* Hans Reizels Forlag.

EMU Danmarks læringsportal (2). (2017). Hentet 3. Marts 2017 fra http://www.emu.dk/omraade/gsk-l%C3%A6rer/ffm/matematik/7-9-klasse/matematiske-kompetencer

Geisnæs, D., & Kirkegaard, P. O. (2017). *Metakognition, selvregulering og konsolidering.* Frederikshavn: Dafolo.

Goldkuhl, G. (2012). Pragmatism vs interpretivism in qualitative information systems research. *European Journal of Information Systems*(2), s. 135-146. Hentet fra https://pdfs.semanticscholar.org/6d3c/20168fcf96de21a9b0bb424a31d3d66d3f24.pdf

Grynberg, S. (26. 5 2016). *Pris til læremiddel til et ikke-eksisterende fag*. Hentet 13. Marts 2017 fra Folkeskolen.dk: https://www.folkeskolen.dk/588256/pris-til-laeremiddel-til-et-ikke-eksisterende-fag

Gynther, K. (2010). *Didaktik 2.0 Læremiddelkultur mellem tradition og innovation.* København: Akademisk Forlag.

Hansen, T. I., & Bundsgaard, J. (2016). *Effektmåling af demonstrationsskoleforsøg.* Læremiddel.dk.

Hermansen, M. (2005). *Læringens univers.* Århus: Forlaget Klim.

Hiim, H., & Hippe, E. (1997). *Læring gennem oplevelse forståelse og handling - en studiebog i didaktik.* Gyldendal.

Howland, K., & Good, J. (2015). *Learning to communicate computationally with Flip: A bi-modal programming language for game creation.* Hentet fra Learning to communicate computationally with Flip: A bi-modal programming language for game creation

Høines, M. J. (2004). *Begynner-opplæring - fagdidaktik for barnetrinnets matematikundervisning* (4. udg.). Landås: Caspar Forlag AS.

IT-Branchen. (19. Marts 2017). *Coding Class*. Hentet 19. Marts 2017 fra IT-Branchen: https://itb.dk/articles/fremtidens-kompetencer/coding-class

Jeffries, R. (16. Marts 2011). *What is Extreme Programming?* Hentet 2. Maj 2017 fra Ron Jeffries: http://ronjeffries.com/xprog/what-is-extreme-programming/

Kafai, Y., & Resnick, M. (1996). *Constructionism in Practice.* New Jersey: Lawrence Erlbaum Associates.

Knoop, H. H. (2004). Om kunsten at finde flow i en verden, der ofte forhindrer det. *Kognition og Pædagogik*(52).

Kvale, S., & Brinkmann, S. (2009). *Interview: Introduktion til et håndværk.* København: Hans Reitzels Forlag.

Misfeldt, M. (December 2010). "Forestillet læringsvej" i it-baserede pædagogiske udviklingsprojekter. *Dansk Pædagogisk Tidsskrift*(4), s. 45-51.

Papert, S. (1983 [1980]). *Den totale skildpaddetur. Børn. datamaskiner og kreative tanker. Mindstorms.* Gylling: G.E.C. Gads Forlag.

Papert, S. (2. June 1998). *Child Power: Keys to the New Learning of the Digital Century.* Hentet 17. April 2017 fra Papert.org: http://www.papert.org/articles/Childpower.html

Papert, S., & Harel, I. (1991). *Constructionism.* Ablex Publishing Corporation.

Pind, P. (2010). *Gode grublere og sikre strategier.* Skødstrup: Forlaget Pind og Bjerre.

Qvortrup, L. (30. Juli 2016). Statusrapport: Glade børn og kedelige timer. *Politiken*.

Repenning, A., Basawapatna, A., & Escherle, N. (2016). CT Tools. I *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (s. 218-222). Cambridge: IEEE.

Resnick, M. (13. Juni 2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *Proceeding C&C '07 Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*, s. 1-6. Hentet fra ACM Digital Library.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*(52), 60-67.

Rohde, L., & Olsen, A. L. (2013). *Innovative elever - undervisning i FIRE faser.* København: Akademisk Forlag.

Sørensen, B. H., & Levinsen, K. (2014). *Didaktisk design digitale læreprocesser.* København: Akademisk Forlag.

Technopedia. (2017). Hentet 6. Marts 2017 fra https://www.techopedia.com/definition/22855/visual-programming-language-vpl

Vygotskij, L. S. (1971). *Tænkning og sprog.* København: Hans Reitzels Forlag.

Wing, J. (Marts 2006). *Computational Thinking.* Hentet 4. Marts 2017 fra Communications of the ACM: http://dl.acm.org/citation.cfm?id=1118215

Wing, J. (2011). Research Notebook: Computational Thinking - What and Why? *The Link*(6), 20.