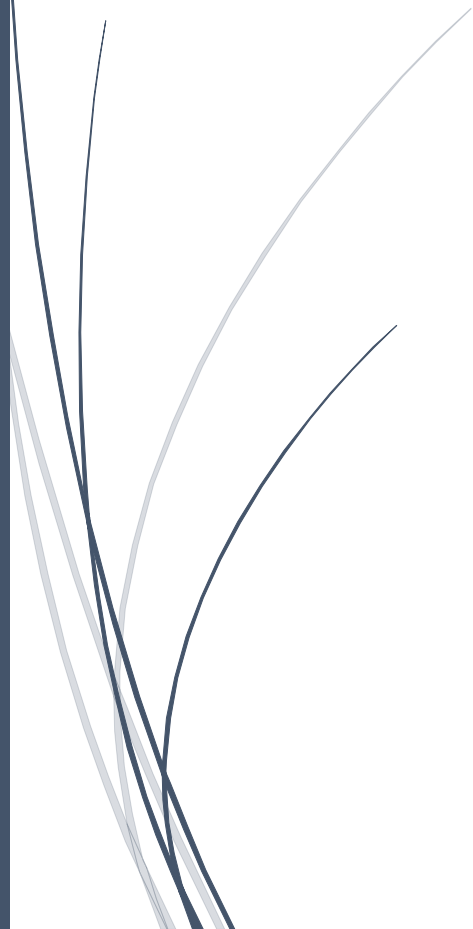


26-05-2019

Kreativ programmering i folkeskolen



Basisprojekt 2

2. semester- forår 2019

Eksamensgruppenr.: S1924791775
Projekt(arbejds)titel: Kreativ programmering i folkeskole
Gruppens medlemmer: Jonas Høst Høvelskov, Emil Just Bluhme, Jonathan Wad Høgsbro, Christian Wintcentsen
Vejleder: Stine Rosenlund Hansen
Hold: A
Dato: 26. maj 2019

Abstract

This project aims to examine the implementation of programming and computing as a part of the curriculum in danish schools, and why different actors think this should be mandatory in danish schools. It appears that many teachers and educational institutions do not feel competent enough to fully implement this kind of teaching. We will through this project try to communicate the benefits that students can acquire by working with programming based teaching in school. We will do this by taking a look at other countries that has already implemented this, and by looking at theorists views on this matter. We aspire to explain the importance of taking this matter seriously in a world that is becoming more and more digitalised. The project also aims to analyse a series of tools that are specially designed for teaching programming to younger students from third grade to high school.

Furthermore this paper will discuss how these programming based educational initiatives have done an actual difference in the world, if the differences correlates with the theory behind, and whether or not these differences have been for the better.

Finally, the paper concludes that, by implementing programming based teaching it seems, that some children acquire skills, that enhances their ability to solve problems - not only in the field of programming.

Resume

Vores projekt omhandler selve implementeringen af programmerings baseret undervisning i folkeskolen som et reelt fag, der tager udgangspunkt i problemløsning. Ved at få programmering og databehandling gjort til en del af pensum i den danske folkeskole, vil det udruste eleverne med nogle kompetencer og redskaber, som kan højne deres generelle problemløsende kompetencer. Vi vil også undersøge og analysere en række værktøjer, der er specielt designet til at undervise programmering til yngre elever fra tredje klasse til gymnasiet. For at kunne forstå om disse initiativer har gjort en forskel, og om de bringer de tiltænkte kompetencer, vil vi diskutere den fremfundne viden, og parallelisere det med teori samt empiri. Til sidst konkluderer vi, på baggrund af realiseret viden, om hvorvidt disse initiativer er givende i uddannelsesregi.

Indholdsfortegnelse

Abstract	2
Resume	3
Indledning	6
Motivation	8
Semesterbindingen	9
Problemfelt	9
Case beskrivelse	10
<i>Oprindelse</i>	10
<i>Folkeskolen</i>	11
<i>Fable robotten</i>	11
Problemformulering	13
Arbejdsspørgsmål	13
Afklaring af målgruppe	14
Afgrænsning	14
Programmering, Blockly, Scratch og Computational thinking	14
<i>Scratch</i> :	16
<i>Blockly</i> :.....	17
<i>Computational thinking</i>	17
<i>Undervisning i Computational thinking</i>	18
Teori	21
<i>Hvorfor er Papert relevant for dette projekt?</i>	22
Metodeafsnit	25
<i>Trin 1: Indre mekanismer og processer</i>	26
<i>Trin 2: Teknologiske artefakter</i>	26
<i>Trin 3: Teknologiers utilsigtede effekter</i>	27
<i>Trin 4: Teknologiske systemer</i>	28
<i>Trin 5: Modeller af teknologier</i>	28
<i>Trin 6: Drivkræfter og barrierer for udbredelse af teknologier</i>	29
<i>Diskussion af TRIN-modellen</i> :	29
<i>Stakeholderanalysen</i>	30
Hvad er det?.....	30

<i>Semistruktureret interview:</i>	33
Analyse	35
<i>Trin-modellen</i>	35
<i>Trin 1: Indre mekanismer og processer</i>	35
<i>Trin 2: Teknologiske artefakter</i>	38
<i>Trin 3: Teknologiers utilsigtede effekter</i>	39
<i>Trin 4: Teknologiske systemer</i>	41
<i>Trin 6: Innovationer, Drivkræfter og barrierer for udbredelse af teknologier</i>	41
<i>Stakeholderanalyse af folkeskolen med fokus på programmering</i>	44
Diskussion	46
Visuel præsentation	48
Konklusion	50
Evaluering	50
Litterateur	52

Indledning

Birgitte Haas siger dette om manglen på IT-specialister ”*Det er en stopklods for Danmarks digitalisering. Det ville gå bedre og hurtigere, hvis vi havde de rette kompetencer*” (IT-specialister er en mangelvare, Hansen, 2018). Den stigende digitalisering kan blive hindret af denne mangel og det er derfor vigtigt at uddanne kompetente IT-specialister. Regeringen har planlagt seks strategier inden for cybersikkerhed, med i alt 68 tiltag. De 68 tiltag fordeler sig på tele, søfart, finans, transport, energi og sundhed (J. Kongstad, 2019). Det er nogle af de udfordringer som der kommer i fremtiden, som kan løses ved at have kendskab og ekspertise inden for programmering og digital forståelse. Problemet er interessant grundet af, at vi lever i en digitaliseret tidsalder. Da digitaliseringen er i stigende kurs, bliver dette felt gradvist vigtigere at besidde kendskab til. Emnet er relevant grundet mangel på IT-specialister. Det står faktisk så slemt til i IT-branchen, at it-virksomheder allerede forsøger at ansætte elever på første semester (IT-B Branchen, 2016).

Hvordan ser digitaliseringen/automatiseringen ud nu, og hvordan kommer det til at se ud i fremtiden?

Martin Ford skriver i sin bog *Rise Of The Robots*, at vi skal prøve at forestille os en person der arbejder på et lager. Denne persons opgave er at skulle identificere forskellige kasser samt løfte og placere dem. Dette fremkommer som værende en let opgave, som personen kan løse nærmest instinktivt. Denne visuelle udfordring er nøjagtigt det som den menneskelige hjerne, har udviklet sig til at kunne varetage. At personen kan forstå og løse denne opgave, virker som værende en komplet selvfølge - personen i dette tilfælde er dog ikke et menneske, men derimod en robot. For at være mere præcis, skriver Ford, er det en slange lignende robot arm, med et hoved som består af en sugekop. Robotten er langsommere til at forstå, end et menneske ville være. Den kigger på kasserne, justerer blikket lidt og tænker lidt mere, for dernæst at bevæge sig frem for, at tage fat i den øverste kasse i stablen. Grunden til at robotten er langsommere til at forstå end mennesket, i dette tilfælde, er, at funktionen kræver, at robotten gennemføre en enormt kompleks computer drevet udregning. Hvis der er en ting vi har lært igennem historien, skriver Ford, så er det, at robotten snart får en opgradering, som givetvis gør det muligt at udføre funktionen hurtigere (Ford, 2015).

Udviklerne der arbejder for Industrial Perception, Inc., Silicon Valley (som har produceret robotten) er overbevist om, at robotten på sigt, vil gøre det muligt at rykke en kasse i sekundet. Det er bemærkelsesværdigt kortere tid end et menneske, som kan rykke en kasse hver ca. Sjette sekund.

Forskellen er, at robotten aldrig bliver træt, skadet eller indgiver et erstatningskrav til arbejdsgiverne (Ford, 2015).

Selvom at robotter i fabriksindustrien ikke er en nyhed, er udviklingen på markedet interessant at kigge på. Ifølge The International Federation of Robotics er markedet for industrielle robotter steget med mere end 60 procent fra år 2000 til 2012, med samlet salg på omkring 28 milliarder dollars i 2012, hvor kina er det hurtigst voksende marked med en vækst på omkring 25 procent årligt fra 2005 til 2012.

Selvom at industrielle robotter leverer en uovertruffen kombination af fart, præcision og styrke, er de stadig, ifølge Ford's egne ord; "blind actors in a tightly choreographed performance" (Ford, 2015).

Robotterne besidder altså visse begrænsninger såsom; at de, såfremt det er robotter med visuel perception, kun kan arbejde under helt særlige og kontrollerede lysforhold, kun har et todimensionelt syn, og i nogle tilfælde kun kan arbejde med objekter på en flad overflade. Det gør det kompliceret for dem at arbejde i et uforudsigeligt miljø.

Resultatet af dette er, at der stadig bliver overladt rutinemæssigt arbejde til mennesker. Det kan for eksempel være at varetage opgaver som, at laste/aflaste lastbiler der levere produkter til og fra fabrikken, eller vælge dele fra en kasse, og fodre dem ind i den næste maskine (Ford, 2015).

Innovationen inden for visuelle perceptions teknologier er dog steget eftersom, at udstyret er blevet billigere. Dette har bidraget til, at for eksempel universitetsstuderende og gør-det-selv innovatorer har hacket sig ind på Kinect (xBox' visuelle perceptions teknologi der kan se i tre dimensioner), hvor de derefter har uploadet videoer til YouTube. Disse videoer viser hvordan de har fået robotter til at se i tre dimensioner. Industrial Perception, som er en virksomheden med robotarme, har ligeledes kreeret deres system med udgangspunkt i Kinect's teknologi. Dette har resulteret i prismæssigt overkommelige robotter, der nu kan arbejde i uforudsigelige miljøer, og som har samme perceptions egenskaber som mennesker (Ford, 2015).

Ford skriver, at en stor del af bogen peger på, hvordan den hastige udvikling inden for teknologien, potentielt kan true en bred vifte af jobs i mange forskellige brancher. Disse jobs varetages af lønmodtagere med en stor uddannelsesmæssig diversitet. Han påpeger, at hvis en sådan trend finder sted, kan det komme til at bringe afgørende konsekvenser for den overordnede verdensøkonomi.

Hvis arbejde og indkomst kontinuerligt bliver automatiseret væk, kan hovedparten af forbrugerne til sidst komme til at mangle den indkomst og købekraft, der er nødvendig for at drive efterspørgslen, der er afgørende for en vedvarende økonomisk vækst. Så hvis automatisering eliminerer en væsentlig del af de jobs, som forbrugerne er afhængige af, eller hvis lønningerne drives så lavt, at meget få mennesker har en betydelig indkomst, så er det svært at se, hvordan en moderne markedsøkonomi fortsat vil kunne trives.

“Automation can be the ally in our prosperity if we will just look ahead, if we will understand what is to come, and if we will set our course wisely after proper planning for the future” (Martin Ford, 2015, s. 32-33).

Automatiseringen finder sted, og det kan, som Ford skriver, være vores allierede såfremt, at vi forstår hvilken omstilling vi står overfor. Er vi i Danmark klar over dette, og bliver der handlet ud fra denne viden?

Børn i folkeskolen er så småt begyndt på at lære programmering. Flere initiativer fra virksomheder i samarbejde med lærere er begyndt at få omstillet almindelige undervisningsplaner med fokus på procesorienteret læring, i stedet for resultatorienteret læring. (Pedersen & Kiellberg, 2019).

Virksomheden Shape Robotic har en vision om at der skal være flere af de her kreative minds. Kreative minds der i princippet lige pludseligt kan opfinde den nye smartphone, eller noget der er bedre. Hack et hverdagsproblem er en problemstilling der stiller rundt omkring på folkeskoler, der har deres produkter. (Pedersen & Kiellberg, 2019). Det siger en del om, det at tænke ud af boksen og være kreativ på måder, børn ikke plejer at være. Det er nemlig de hverdagsproblemer der kan opstå, som kan gøre børn innovative og kreative omkring.

Motivation

Vores motivation er blandt andet kommet ved selv at sidde og arbejde med programmering. Det gjorde vi i faget ”teknologiske systemer og artefakter”. Her blev vi alle meget interesseret i at arbejde med især kodning, og programmering som problemløsende undervisning og mange af os vil også læse videre i den retning når vi efter sommerferien får mulighed for at vælge fag til 3. semester.

Fælles for os, var der en generel under over, at vi ikke var blevet konfronteret med fag, hvor programmering indgik. Kodning og digitalisering er omkring os alle hver dag, det er baggrunden for, eksempelvis, vores sociale medier. Vi har alle smartphones i vores lommer og det er de færreste, som egentlig ved hvordan disse værktøjer fungerer. Vi bevæger os alle på nettet uden rigtigt at kende baggrunden for systemet og hvad det indebærer.

Det har fået os til at tænke over hvordan de yngste, særligt dem i folkeskolen, kan blive mere interesseret i programmering. Derfor er vi meget interesseret i at undersøge, om der er blevet igangsat lignende initiativer, og i så fald hvilket udbytte disse eventuelt kan give.

Semesterbindingen

I forbindelse med 2. semesters projektopgave, skal vi gøre brug af faget teknologiske systemer og artefakter. Vi vil inden for denne dimension vil vi gøre brug af TRIN-modellen, som vi vil bruge til at analysere en given teknologi, ved hjælp af trin 1, 2, 3, 4 og 6, som alle er en del af denne model. Her har vi valgt at tage udgangspunkt i en teknologi, som bliver benyttet på skoler, til at fremme forståelsen af programmering hos folkeskoleelever.

Yderligere bruger vi faget subjektivitet, teknologi og samfund. Inden for denne dimension vil vi især benytte os af stakeholderteorien, som vi vil bruge til at analysere de forskellige interesser og aktører, inden for denne teknologi. Ydermere er det relevant at kigge på samspillet mellem mennesker og teknologi, i dette tilfælde børn og teknologi, da hele denne indlæring kommer igennem interaktion, mellem disse to parter.

Inden for subjektivitet teknologi og samfund, har vi også valgt at benytte os af et interviewmetoden semistruktureret interview, til at interviewe diverse aktører inden for emnet.

Problemfelt

Den kommende generation er en befolkningsgruppe, hvor det er samfundets ansvar at klæde dem rigtigt på til fremtiden. Folkeskolen er et sted hvor læring og dannelse finder sted, og det er derfor et ideelt sted at give redskaber. Det kan dog være svært at skelne mellem hvad der er vigtigt at lære og hvad der ikke er. Skal børn lære for et fremtidigt arbejdsmarked, et fremtidigt studie eller fordi det er

sjovt? Programmering er en vigtig egenskab at dyrke, fordi alt tyder på en langt mere automatiseret arbejdsmarkedet i fremtiden.

”Hastige fremskridt inden for automatiseringsteknologi, såsom kunstig intelligens og avanceret robotteknologi, er begyndt at få mærkbar effekt på markeder og samfund verden over. Også det danske arbejdsmarked står over for omfattende forandring. Vi anslår, at eksisterende teknologi kan automatisere op mod ~40 pct. af de samlede arbejdstimer i Danmark” (McKinsey, 2007, s. 1)

McKinsey anslår at den eksisterende teknologi kan 40 pct. af arbejdsmarkedet automatiseres. Selv om det er gisninger giver det en indikation på hvilket retning arbejdsmarkedet bevæger sig hen mod (McKinsey rapport). Den teknologiske udvikling er stigende, derfor er det vigtigt at forberede sig på dette.

”For otte år siden havde jeg ikke en smartphone, men nu kunne jeg ikke forestille mig min dagligdag uden apps der løser alt muligt i mit liv, og det er otte år. sådanne udviklinger kan bare gå hurtigt” (Pedersen & Kiellberg, Bilag 1, 2019, s. 2)

Dette projekt vil fokusere på programmering i folkeskolen, samt de styrker og svagheder der er forbundet med programmering.

Styrkerne er at programmering stimulerer børn kreativt og fysisk. Hos Shape Robotics benytter de sig af blokprogrammering og robotter. Robotterne bliver benyttet ud fra princippet der hedder ”hack et hverdagsproblem”.

Der er dog problematikker der er forbundet med programmering i folkeskolen. En af dem er undervisere der ikke føler sig klædt på til undervisningen. ”Jeg kan så bare sige at 70 procent af mit lærerværelse har ikke lyst til at gå ind til en time og ikke være klædt på” (Pedersen & Kiellberg, Bilag 1, 2019, s. 5)

Case beskrivelse

Oprindelse

Projektet tager udgangspunkt i Shape Robotics, som er en virksomhed der ligger i Farum.

Shape Robotics er en virksomhed, som arbejder med programmering. De benytter Fable, som er en videreudvikling af Scratch. Fable startede på Fyn, nærmere betegnet Odense. Fyn er kendt som hovedstaden inden for robotteknologi. Fable er udviklet af CEO for Shape Robotics, David Johan Christensen. David Johan Christensen startede på syddansk universitet og senere hen, har han været underviser og lektor på DTU. David Johan Christensen startede sin karriere ud, med at lave et projekt sammen med Lego, og dette projekt var robotten Legoboost. Det er en consumer robot, hvor formålet er at forbrugeren selv, skal bygge noget og så programmere efter. Det lå fjernt fra det ophav i industrirobotter, som David Johan Christensen havde så derfor udviklede han videre på det, da det virkede for meget som legetøj og ikke var tro mod industrien. Det var derfor han gik i gang med Fable robotten (Pedersen & Kiellberg, 2019).

Folkeskolen

”Grundlæggende kan man sige at verdenen har ændret sig og undervisningen forhåbentligt også kan følge med at outputtet førhen var super fint hvis eleverne fik en opgave og løste den til punkt og prikke” (Pedersen & Kiellberg, Bilag 1, 2019, s. 1)

Det danske samfund er i konstant forandring, dette afspejler også folkeskolen. Hos Shape Robotics skriver de at elever fra tredje klasse og helt op til universitetet kan benytte robotten, samt det tilhørende program (Shape Robotics, u.d.). Fable robotten bliver benyttet især i folkeskolen, hvor kreative sind er det mindset, som fable robotten skal være med at til at give. Shape Robotics har deres fokus på problemorienteret løsningsmodeller. Dette kommer til udtryk ved deres ”Hack et hverdagsproblem”-princip. Den slags øvelser træner hjernen til at tænke mere kreativt.

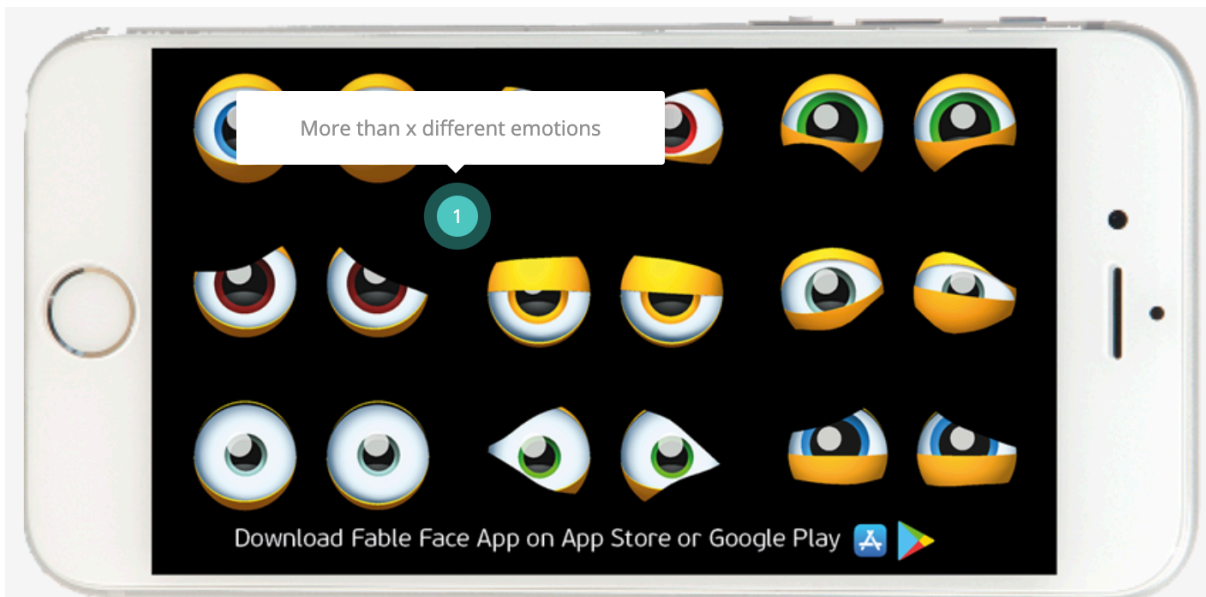
”Når i får det her ”hack et hverdagsproblem” er det ikke fordi i får en perfekt robot, som løser et problem, men i kommer rigtig meget ind i det mindset der hedder ’hvad nu hvis’” (Pedersen & Kiellberg, Bilag 1, 2019, s. 2).

Det er netop dette mindset, som Shape Robotics vil arbejde med, med folkeskoleeleverne.

Fable robotten

Selve robotten er bygget op med forskellige dele, som kan monteres og afmonteres efter det givne problems behov. Delenes funktion kan variere fra at have hjul og kunne køre, til at have arme, der

kan rotere. Derudover kan en smartphone også monteres, dette bliver benyttet til at give robotten øjne, som en form for personificering.



(Shape Robotics, u.d.)

Som det kan ses på billedet (bilag) er der flere forskellige øjne, der er med til at give robotten menneskelige følelser, som for eksempel vrede, tristhed, glæde og så videre.

Alle delene bliver forbundet med en hub, der via bluetooth laver et lukket system, hvor delene reagerer på hvilken kodning der bliver kaldt, i programmet. De forskellige dele har også et unikt nummer, som skal indtastes (Shape Robotics, u.d.).

Programmeringssproget er blokprogrammering, hvor det ikke er kodelinjer, men blokke med kommandoer. Blokkene skal bygges ovenpå hinanden og kodningen foregår ellers på samme måde, som andre programmeringssprog. I højre side er der et vindue, hvor selve kodelinjerne kan ses. Programmeringssproget er python (Shape Robotics, u.d.).

Sct. Jørgens Skole er en helt almindelig folkeskole, med beliggenhed i Roskilde. De har omkring 650 elever på tværs af alle årgange fra 0. til 9. klasse. På Sct. Jørgens Skole er de for nyligt begyndt på at gå meget ind i programmering hos eleverne. Lars Slynghborg der er naturfagslærer på Sct. Jørgens Skole havde i april måned i år, haft projekter i 0. klasse der involverede programmering med iPads og et projekt kaldet SPhero, der også er en robot, som Shape Robotics Fable. De startede dog med et projekt kaldet LEGOMINDSTORM, for omkring 3 år siden, som kører helt fra 3. klasse

til 9. klasse. Shape Robotics robot, er først kommet ind lidt senere, da det er en nystartet virksomhed. Lars Slynborg har haft kontakt til deres salgsmedarbejder Maja, som vi efterfølgende også kom i kontakt med. Lars blev ansat på skolen i maj måned sidste år, og har herefter fået skolen til at investere i Fable robotterne, hos Shape Robotics.

Problemformulering

- *Hvad er drivkræfterne bag øget IT-forståelse i folkeskolen, og hvilke kompetencer kan dette give?*

Arbejdsspørgsmål

Hvad er programmering?

Dette spørgsmål vil vi gerne redegøre for hvad programmering er. Det er et begreb som der er meget bredt og derfor dækker over mange ting. Vi vil besvare på dette ved at undersøge eksisterende viden inden for området.

Hvad er formålet ved at lære børn programmering?

Dette spørgsmål bliver der redegjort for hvorfor vi mener, at det er vigtigt at lære børn at programmere. Dette spørgsmål skal besvares ved at analysere interviews samt eksisterende data inden for området.

Hvilke læringsredskaber bliver benyttet i folkeskolen til at undervise børn i programmering?

Dette spørgsmål lægger op til en analyse af hvordan programmering bliver benyttet i folkeskolen. Dette spørgsmål skal besvares ved et interview med en skolelærer, hvor programmering bliver benyttet.

Hvad er styrkerne og svaghederne ved Fable Blockly?

Vi vil i trin 1 i trin modellen besvare dette spørgsmål.

Dette spørgsmål lægger op til at analysere de eksisterende tiltag som der bliver benyttet i folkeskolen. Spørgsmålet skal besvares ved at bruge TRIN-modellen, på den måde kan tiltagene brydes op og blive mere overskueligt.

Hvilke aktører indgår i folkeskolen og hvad er de forskellige aktørers motiver?

Dette spørgsmål lægger op til en analyse af den valgte case. Her vil der blive benyttet stakeholderanalysen. Stakeholderanalysen vil blive brugt til at finde information og tydeliggøre lærer og Shape Robotics roller som aktører.

Afklaring af målgruppe

Selve de programmer og værktøjer vi beskriver, samt opgavens generelle indhold, henvender sig til implementering af programmering i folkeskolen. Derfor er en oplagt målgruppe for os, folkeskoleelever fra tredje klasse og frem til gymnasiet. Dette er den optimale målgruppe for os, da virksomheden Shape Robotics bruger dette segment til udviklingen af deres produkter. Vi analyserer derfor målgruppen for Shape Robotics, sammen med programmeringsprogrammet Blockly og robotten Fable. En anden målgruppe vi også fokuserer på er lærerne. Lærerne står for undervisningen og vi ønsker at finde metoderne for deres undervisning.

Afgrænsning

Vi har valgt at afgrænse til de danske folkeskoler, da det også er disse, som er målgruppen for den virksomhed vi har undersøgt. Yderligere undlader vi at inddrage privatskoler, da de ikke er underlagt samme regler og vilkår, som den almene folkeskole. Vi har også valgt at undlade interviews med børn, da det mere er undervisernes synsvinkel, vi vil inddrage i rapporten. Det gør vi fordi vores fokus ligger på hvorfor børn skal lære at programmere og hvilke kompetencer det giver fra et teoretisk synspunkt. Vi har også valgt at afgrænse os metodisk, til kun at bruge de mest relevante trin i trin-modellen, samt brugt analysemodeller i STS-sammenhæng.

Programmering, Blockly, Scratch og Computational thinking**Hvad er programmering?**

Tidlige publikationer peger på, at folk indenfor computerverden anvendte "computing".

“Computing is any activity that uses computers. It includes developing hardware and software, and using computers to manage, process, and communicate information for various purposes.”

(Wikipedia, u.d.)

Disse mennesker gav klare definitioner af programmering, uden de nødvendigvis tiltænkte dem, som værende dybdegående analyser af menneskelig aktivitet. Hartree (1950) forklarede

“The process of preparing a calculation for a machine can be broken down into two parts, ‘programming’ and ‘coding’. Programming is the process of drawing up the schedule of the sequence of individual operations required to carry out the calculation” (Blackwell, 2002).

‘Coding’ eller kodning var en ekstremt tidskrævende process, og programmering blev da den primære aktivitet indenfor feltet. Wilkes (1956) beskrev programmering og andre former for computer relaterede beregninger som *“The sequence of orders is known as the programme, and the machine performs it automatically without intervention from the user”* (Blackwell, 2002).

Det at en notation eller et sprog blev udviklet til, at udtrykke programmet førte til metaforer for programmering, som kommunikation; *“Programming ... is basically a process of translating from the language convenient to human beings to the language convenient to the computer”* (Blackwell, 2002). Denne måde at definere programmering på var dog en, allerede dengang, ikke signifikant måde at definere programmering. Faktisk kunne definitionen vendes om;

“The process of organizing a calculation can be divided into two parts – the mathematical formulation and the actual programming ... translating ... into the language of the computing machine” (Blackwell, 2002).

Ved udgangen af det første årti inden for computer forskning, blev begreberne af et program som værende en automatisk sekvens, en matematisk specifikation og en sproglig oversættelse mellem de to, fastgjort; *“This sequence [of basic operations] is called the program and the process of preparing it is called programming”* (Blackwell, 2002).

Programmering er opgaven i at finde en præcis matematisk formulering og metode til løsning, eventuelt i et “convenient problem-oriented language”, hvis symboler er “more closely related to the mathematical problem to be solved” (Blackwell, 2002).

Forsknings definitioner på programmering:

I takt med at programmeringen bevægede sig længere væk fra et matematisk domæne, har de grundlæggende operationer, metoder, symboler og formuleringer til løsning, alle udviklet sig. Et introducerende kapitel i bogen "Psychology of programming" beskriver, at programmering har ændret sig fra måden man beskriver kalkulationer, til det at definere funktioner for dernæst at definere, samt behandle objekter. Disse termer afspejler ændringerne i programmering fra matematiske til mere generelle databehandlings problemer. Inden for almen datalogi (altså uden for den psykologiske definering af programmering) kan disse fremskridt betegnes som værende mere eller mindre intuitive af computer forskere, men kriterierne for at definere disse vilkår er ofte et spørgsmål om en høflig aftale, snarere end en empirisk undersøgelse. Det aftalte grundlag for den menneskelige aktivitet i programmeringen er stadig;

"to write a sequence of coded instructions fed into a computer ... to arrange data in a suitable form so that it can be processed by a computer ... to feed a program into a computer" (Alan F. Blackwell, 2002).

Hvilke læringsredskaber bliver benyttet i folkeskolen til at undervise børn i programmering?

Scratch:

Med det MIT baserede program 'Scratch', kan brugeren selv programmere interaktive historier, spil og tegnefilm samt dele disse med andre i et online fællesskab. Den overordnede ide med 'Scratch' er, at hjælpe børn til at lære at tænke kreativt, tænke systematisk og arbejde sammen. Alle disse kompetencer ser dem der har lavet 'Scratch', som værende nødvendige kompetencer i det 21. Århundrede. 'Scratch' er designet for unge mellem 8 og 16 år, men anvendes af alle aldersgrupper. 'Scratch' projekter bliver lavet, i mange forskellige omgivelser som i hjemmet, i skolen, på biblioteket og kulturcentre. Når brugerne lære at kode i 'Scratch', lære de vigtige kompetencer indenfor problemløsning, design af projekter og kommunikation.

‘Scratch’ bliver brugt i mere end 150 forskellige lande og er ydermere tilgængeligt på mere end 40 sprog. ‘Scratch’ bliver anvendt på tværs af fag såsom matematik, computer science, sprog og sociale studier.

‘Scratch’ anvender byggesten til visuelt at repræsentere programmeringskomponenter såsom handlinger, begivenheder og operatører. Hver blok har en form, der kun gør det muligt at kombinere den med en kompatibel genstand (Scrath, u.d.).

Blockly:

‘Blockly’ er googles og MIT’s videreudvikling af ‘Scratch’ som anvender samme byggestens metafor, men det kan vise kode igennem flere forskellige programmeringssprog som JavaScript, Python, PHP, Lua og Dart (Blockly, u.d.).

Brugeren kan se programkoden live i ‘Blockly’ samt skifte til andre programmeringssprog løbende, så brugeren kan danne sig et overblik, over forskelligheden på tværs af programmeringssprog. Dette er ideelt i forhold til at lære programmering til alle aldersgrupper.

Google arbejder med MIT for at udvikle den næste generation af ‘Scratch’ baseret på ‘Blockly’ platformen.

‘Blockly’ er ikke lige så færdigudviklet som ‘Scratch’ endnu, men det er blevet spået til, at have en fremtid indenfor læringsværktøjer, der har til formål at lære brugere om programmering (Lifewire, 2019).

Computational thinking

I dag er en del lande allerede gået i gang med at forsøge at integrere mere programmering på grundskoleniveau, med formål at øge interessen for IT hos børn i folkeskoler. Yderligere er programmering begyndt at blive set, som et naturligt redskab, som er nødvendigt at besidde i en mere digitaliseret verden. Det er derfor vigtigt at computational thinking bliver integreret som en hovedsag i landenes programmeringsundervisning, mener Eva Petropouleas, som er pædagogisk konsulent for programmering og forfatter af bogen ”programmering i praksis” (Petropouleas, 2018).

Computational thinking er et begreb, som omhandler problemløsning. Det er en form for logisk tankegang, som øger ens kompetencer i forhold til at se på et givent problem, analysere det for da, at frembringe en løsning.

I 2009 definerede professor og datamatiker, Jeannette Wing, begrebet i form af de tankeprocesser, som begrebet indebærer (Petropouleas, 2018).

Computational thinking kan betegnes ud fra nogle forskellige nøglebegreber:

Dekomposition – opdele et problem eller et system i mindre og mere håndgribelige dele.

Mønstergenkendelse – se efter ligheder i og mellem problemer.

Abstraktion – fokusere på relevante informationer og ignorere det irrelevante.

Algoritmer – løse problemet ”step-by-step”.

Hvert begreb er lige vigtigt i denne tankegang. Det kan ses som ben på et bord, hvor hvis der mangler et ben, vil bordet sandsynligvis kollapse (Petropouleas, 2018).

Computational thinking indebærer at tage det komplekse problem og bryde det ned i en række små, mere håndterbare problemer (dekomposition). Hvert af disse mindre problemer kan derefter ses individuelt, idet man overvejer, hvordan lignende problemer er blevet løst tidligere (mønstergenkendelse) og kun fokuserer på de vigtige detaljer, imens irrelevant information ignoreres (abstraktion). Dernæst kan enkle trin eller regler for at løse hvert af de mindre problemer fremfindes (algoritmer).

Siden da er begrebet blevet bearbejdet og yderligere ekspanderet til også at indeholde tankeprocesserne hhv. logisk ræsonnement, samt en mere evaluerende fase (Petropouleas, 2018).

Udtrykket ’computational thinking’ stammer fra 1950’erne, hvor det dengang var kendt som ”algorithmic thinking”. Dengang blev udtrykket betegnet som en mental orientering hvor problemer blev konverteret til løsninger som en slags input/output, altså en form for algoritmisk tankegang. (Denning, 2009).

Undervisning i Computational thinking

I 1980 kom Seymour Papert frem med begrebet ”computational thinking”. Paperts tanke var, at ved at lære børn at programmere, ville de igennem denne læring, også lære at se problemer, emner og selve verden på en ny og anderledes måde. Dette var en del af Paperts læringsteori konstruktionisme (Petropouleas, 2018).

Begrebet computational thinking transcenderer programmeringsfagets grænser. Det er en yderst anvendelig metode at benytte på tværs af flere skolefag, da det kan genfindes i en række faglige kompetencer, samt problemløsningstilgange. Elever kan på den måde tilegne sig flere kompetencer end bare at lære programmeringssprog, ved at integrere dette i skolesystemet (Petropouleas, 2018).

I 2014 lancerede undervisningsministeriet i England en ny national læseplan, som indeholdt programmeringsundervisning, med det formål at give eleverne nogle flere kompetencer ift. analysering og problemløsning. De specifikke mål for læseplanen lød således: (Department of Education, 2013).

- Eleverne kan forstå og anvende de grundlæggende principper og begreber inden for datalogi, herunder abstraktion, logik, algoritmer og datarepræsentation
- Eleverne kan analysere problemer i beregningsmæssige termer og har gentaget praktisk erfaring med at skrive computerprogrammer for at løse sådanne problemer
- Eleverne kan evaluere og anvende informationsteknologi, herunder nye eller ukendte teknologier, analytisk til at løse problemer
- Eleverne er ansvarlige, kompetente, tillidsfulde og kreative brugere af informations- og kommunikationsteknologi.

Kvalitativ undervisning i programmering kan udruste skoleelever med disse tankeprocesser, samt en mere kreativ tankegang. Programmering har yderligere ligheder med både matematik, natur og teknik, samt design og kan give en dybere indsigt, i naturlige og kunstige systemer (Department of Education, 2013).

I Estland har regeringen implementeret programmering som en fast del af undervisningen lige fra skolestart. Dette har de gjort i form af lanceringen af ”ProgeTiiger” (Den programmerende tiger), som er deres initiativ til undervisning af programmering i folkeskolen. Dette initiativ går ud på at alle børn skal lære at kunne programmere en form for computerprogrammer, lige fra de starter i skole, til de er færdige. (Exner, 2012).

Lektor på DTU, Andreas Bærentzen mener også, at det ville være yderst hensigtsmæssigt, hvis eleverne allerede stiftede bekendtskab med noget mere programmeringsbaseret læring i folkeskolen. Han mener, at det ville gøre læringskurven for de videregående uddannelser mindre stejl, hvis man tilegner sig disse kompetencer i en tidligere alder.

"Jeg tror det handler om at få nogle grundlæggende ting ind i elevernes begrebsverden, så de ikke siden hen skal bruge stor mental energi på at kravle hen over en mur af forvirring over helt grundlæggende begreber" (Frank, 2013).

Denne IT – og programmeringsfaglighed kan også kombineres og hjælpe med at give en bedre forståelse for de almene grundskolefag, som matematik, dansk og samfundsfag, mener leder for det nationale videncenter for læremidler, Thomas Illum.

"I et pilotprojekt på nogle skoler kunne man forsøge sig med bl.a. programmering i relation til matematik, dansk og samfundsfag eller i musik. Det ville være interessant at undersøge, hvor det passer ind" (Frank, 2013).

En integration af denne programmeringsbaserede tankegang, kan give eleverne kompetencer i forhold til at kunne se nogle systematiske sammenhæng, samt hvordan disse kan ændres. (Frank, 2013).

Undervisning af programmering og computational thinking handler ikke kun om at lære at bruge systemer og værktøjer, men altså også at arbejde med principper og metoder, inden for feltet. Der fokuseres på nogle grundlæggende videnskabelige begreber inden for databehandling og formidling, som altså kan give nogle kompetencer, inden for en bredere forståelse af mere end bare programmering. (Nardelli, 2019).

Teori

For at forstå baggrunden for Shape Robotics undervisning, går vi her i dybden med Seymour Papert's konstruktionisme.

Seymour Aubrey Papert var en matematiker, datalog og underviser, som primært tilbragte sin karriere på at undervise og forske på MIT (Massachusetts Institute of Technology). Papert ses som værende en af verdens eksperter, når det kommer til at undersøge hvordan teknologi kan bidrage til nye måder, hvorpå man kan lære og undervise i matematik og generel tænkning (Papert.org, u.d.). Papert udviklede en teori til læring (konstruktionisme) baseret på Piaget's konstruktivisme. Han beskriver med egne ord

“Constructionism—the N word as opposed to the V word— shares constructivism's view of learning as “building knowledge structures” through progressive internalization of actions... 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 (Ackermann, 2001, s. 4).

Konstruktionismens fokus på læring gennem skabning hjælper os med at forstå *“how ideas get formed and transformed when expressed through different media, when actualized in particular contexts, when worked out by individual minds”* (Ackermann, 2001, s. 4).

Altså hjælper konstruktionismen os med at forstå, hvordan ideer bliver formet og transformeret, når de bliver udtrykt gennem forskellige medier, samt aktualiseret i bestemte sammenhæng.

For Papert er det, at projektere vores indre følelser og idéer, nøglen til læring. At udtrykke idéer gør dem håndgribelige og delelige, hvilket former og skærper disse idéer, samt hjælper os med at kommunikere med andre gennem vores udtryk. Selvstyret læring er en iterativ proces, hvor den lærende selv opfinder værktøjerne til, at udforske hvad der interesserer dem mest - *“Learners, young and old, are “word- makers,” in Nelson Goodman's sense.”* (Ackermann, 2001, s. 4).

Mange forskere inden for socio-konstruktivistisk tradition, har før Papert undersøgt, hvordan at kulturelle artefakter som værktøjer, sprog og mennesker, har været en resurse til, at udvide

personers kognitive potentialer. Forskellen på disse og Papert, ifølge Ackermann, bunder ud i tre ting:

1. Rollen som disse eksterne midler betyder for personer, på et senere tidspunkt i deres udvikling.
2. Typen af de forskellige eksterne midler eller medier der bliver undersøgt. Papert fokuserede på digitale medier og computer-baserede teknologier.
3. Typen af initiativ, som den lærende tager i designet af hans/hendes egne “objects to think with”. For Papert er viden, selv i blandt voksne eksperter, baseret på kontekst, formet af brug og brugen af eksterne midler, fremkommer som værende essentiel, i forhold til at udvide potentialet af den menneskelige forstand - uanset hvilket udviklingsstadium de befinder sig på (Ackermann, 2001, s. 5).

Papert beskriver hvordan hans interesse for biler, for eksempel, har bidraget til en bedre matematisk forståelse. En forståelse han føler, at han ikke fik fra skolen; *“I believe that working with differentials did more for my mathematical development than anything I was taught in elementary school”* (Papert, 1980, s. 5).

Hvorfor er Papert relevant for dette projekt?

Da den personlige computer blev introduceret i slutningen af 70'erne, var der en generel entusiasme omkring at lære børn at programmere. Tusindvis af skoler lærte millioner af børn at skrive simple programmer i Logo eller Basic. Paperts bog *Mindstorms*, præsenterede i 1980 Logo, som værende en hjørnesteen i forhold til at nytænke tilgange til læring og uddannelse (Resnik, 2009, s. 62). Grundet den tids kompleksitet af programmeringens læringsinitiativer, blev computeren da anvendt til andre aktiviteter i skoleregime. Papert argumenterer dog for, at programmering skulle have et “low floor” (nemt at komme i gang) og et “high ceiling” (muligheder for at kreere mere komplekse programmer over tid). I forlængelse af dette argumenterer Papert for, at programmeringssproget skulle have “wide walls” (altså skulle det kunne understøtte mange forskellige projekter, så personer, med forskellige interesser og læringsstilarter, alle kunne lære at anvende det) (Resnik, 2009, s. 62). Disse principper er nogle som MIT prøver at imødekomme med deres program Scratch. Da Blockly er Google's og MIT's initiativ om den næste generation af Scratch, er det interessant at gå i dybden med den bagvedliggende teori, for at klargøre den grundlæggende tanke bag et sådant program (Lifewire, 2019).

Ideen om "A Visual Programming Language (VPL)" handler om, at kreere programmer, primært ved hjælp af grafisk manipulation. Ideen om et visuelt programmeringssprog er inspireret af måden, hvorpå børn leger med legoklodser;

"Given a box full of them, they immediately start tinkering, snapping together a few bricks, and the emerging structure then gives them new ideas. As they play and build, plans and goals evolve organically, along with the structures and stories. We wanted the process of programming in Scratch to have a similar feel." (Resnik, 2009, s. 63).

Da VPL er en nyere betegnelse, der dækker over et program, som prøver at imødekomme Paperts principper ("low floor", "high ceiling" og "wide walls") er det interessant at kigge på Paperts teori, samt hvad der har influeret den. Paperts konstruktionisme er, som tidligere nævnt, baseret på Jean Piagets konstruktivisme - men hvad er konstruktivisme?

Piagets konstruktivisme giver et indblik i, hvad børn er interesseret i, og hvad de er i stand til at opnå på forskellige stadier af deres udvikling. Teorien beskriver, hvordan børns måder at gøre og tænke ting på, udvikler sig over tid, og under hvilke omstændigheder børn er tilbøjelige til at give slip på deres holdninger. Piaget antyder, at børn har meget gode grunde til ikke at opgive deres verdenssyn bare fordi, at en anden, selv hvis vedkommende er ekspert, fortæller dem, at det er forkert. (Ackermann, 2001, s. 1).

Piaget skriver:

1. At undervisningen altid er indirekte. Børn tager ikke bare det med, som de bliver fortalt, men fortolker det derimod i forbindelse med deres egen viden og erfaring - altså transformere de selv de input, som de får.
2. At viden, ifølge Piaget, ikke bare er information der bliver modtaget, husket og anvendt. I stedet argumenterer hans teori for, at viden er en oplevelse der opnås gennem interaktion med verden, mennesker og objekter.
3. At en teori om læring, der ignorerer modstand mod at lære, er fejlagtig. Piaget argumenterer for, at børn faktisk har gode grunde til ikke at opgive deres syn på verden bare fordi, at de bliver præsenteret for et nyt (Ackermann, 2001, s. 3-4).

Papert skriver, at Piaget argumenterer for hvordan, at de kognitive udviklingstrin er uforanderlige - hvilket et utal af kryds-kulturelle undersøgelser efterfølgende synes at have bekræftet (Papert, Mindstorms, 1980, s.174). Papert hævder dog, at grunden til, at en sådan universel ens kognitiv udvikling hos børn skyldes, at kulturen ikke giver mange muligheder for, at arbejde med elementerne i systematiske procedurer som, for eksempel, kombinatoriske opgaver (Papert, 1980, s.175). Et barn får en samling perler i forskellige farver (grøn, rød, blå og sort), og bliver derefter spurgt, om barnet kan konstruere alle de mulige par

af farver: grøn-blå, grøn-rød, grøn-sort osv. Ligesom at børn ikke, ifølge Piagets teori, besidder evnen til at forstå, at når vand hældes fra et glas i et anderledes formet glas, forbliver mængden af vand den samme. Før de er syv år gamle, forstår børn verden over ikke, at udføre en sådan kombinatorisk opgave. Dette forstår de først, når de er 11 eller 12 år gamle (Papert, 1980, s.174). I computer regi er de mest fremtrædende komponenter af en sådan kombinatorisk opgave relateret til ideen om procedure-systematik og "debugging"

"Debugging is the process of finding and resolving defects or problems within a computer program, that prevent correct operation of computer software or a system" (Wikipedia , u.d.).

En succesfuld løsning består af følgende procedurer:

1. Separer perlerne i to farver.
2. Vælg en farve A til farve 1.
3. Skab alle par der kan blive skabt med farve 1.
4. Vælg farve 2.
5. Skab alle de par der blive skabt med farve 2.
6. Gør dette for hver farve.
7. Gå tilbage og fjern dubletter.

Altså det som opgaven handler om er at skrive og eksekvere et program, der inkluderer det vigtige "debugging" stadie. Paperts observation foreslår en grund til, at børn først opnår disse evner senere på deres udviklingsstadiet, for at rekapitulere, kulturen (på det tidspunkt) gav ikke mange muligheder for, at arbejde med elementerne i systematiske procedurer som, for eksempel, kombinatoriske opgaver. Papert skriver *"In our culture number is richly represented, systematic procedure is poorly represented."* (Papert, 1980, s.175). Hvilket han anvender som argument for, at der er denne forskel indlæringsmæssigt, fra børn er syv, til at de er 11 og henholdsvis 12 år gamle. Papert skriver videre *"I see no reason to doubt that this difference could account for a gap of five years or more between the ages at which conservation of number and combinatorial abilities are acquired."* (Papert, 1980, s.175). Standardmetodikken for at undersøge en sådan hypotese er; at sammenligne børn i forskellige kulturer. Dette er allerede blevet gjort, i forhold til Piagets stadier. Børn på alle udviklingsstadier i over hundrede forskellige kulturer fra alle kontinenter er blevet bedt om, at hælde væske og sortere perler. I alle tilfælde peger det på, at den kombinatoriske evne først

kommer fem eller seks år efter forståelsen for, hvordan, som tidligere nævnt, at når vand hældes fra et glas i et anderledes formet glas, forbliver mængden af vand den samme (Papert, 1980, s.175). Dertil sagt, fortsætter Papert, bringer denne opdagelse ham ingen tvivl om hans hypotese. Det kan godt være, at numerisk viden er mere fremtræden, i en kultur der er før computeren, end kendskab til programmering og computational thinking. Tingene ser måske anderledes ud i en computer-rig kultur i fremtiden. Hvis computere og programmering bliver en del af hverdagslivet blandt børn, vil springet til forståelse af kombinatoriske problemer helt forsvinde, og det kunne tænkes, at udviklingen måske ville komme til at se anderledes ud "*Children may learn to be systematic before they learn to be quantitative!*" (Papert,1980, s.175).

Metodeafsnit

I vores metodeafsnit vil vi især gerne benytte TRIN-modellen fra TSA 1, som vi havde på 1. Semester. TRIN-modellen udmærker sig i, at være en metode med 6 forskellige analysepunkter, som har til formål at beskrive en teknologi, med hovedvægt på de teknisk-videnskabelige aspekter, i en given teknologi. (Slide 20, TSA 1. indre mekanismer). Vi benytter TRIN-modellen på et program og en robot der hedder Fable Blockly. Det bliver produceret af virksomheden Shape Robotics. Vi vil ud fra hvert trin i modellen, beskrive selve trinnet og dens fokuspunkter, samt indføre eksempler om projektets program og teknologi, men også om andre eksempler der kan understøtte tankegangen bag de forskellige trin. Programmet Fable Blockly kom vi frem til, gennem interview med en folkeskolelærer på skolen Sct. Jørgens Skole, der har med denne teknologi at gøre i undervisningen helt fra 3. klasse til 9. klasse. Vi interviewede en naturfagslærer ved navn Lars Slynghborg på Sct. Jørgens Skole, som gav os et godt indblik i hvordan de bruger programmering på deres folkeskole. Han henviste os dernæst til en workshop med Shape Robotics der ligger i Farum. Fable Blockly er et projekt og et program, som Sct. Jørgens Skole bruger i undervisningen. Det kan bruges i dansk, naturfag, fysik og som et selvstændigt valgfag. Det omhandler programmering af en robot, som skal kunne løse forskellige opgaver eller fortælle en historie. Kun fantasien sætter grænser for dette projekt. Vores TRIN-model vil derfor blive omkring dette. I vores metodeafsnit afklarer og redegør vi også, for den interviewmetode, vi har anvendt i forløbet, samt komme ind på vores interviewguide og oplevelser omkring interviewet og workshop. Vi bruger dernæst TRIN-

modellen til at analysere læringsredskaberne og undervisningsmetoderne, med vores viden fra workshop og interview, samt redegørelsen fra hvert trin.

Trin 1: Indre mekanismer og processer

Her vil vi lave en identificering og analyse af programmets indre mekanismer og processer, som helt basalt beskriver, hvordan programmet er opbygget. Det handler også om at forklare hvordan programmet virker og hvordan der navigeres i dette. Indre mekanismer og processer er inspireret af Walter Vincents begreb, kaldet det operationelle princip. Vincenti introducerede begrebet i bogen "What Engineers Know and How They Know it". Det operationelle princip betyder helt enkelt; hvordan teknologien virker (Jørgensen, 2018, s. 29). Et af Vincents eksempler er, hvordan kan en flyvemaskine flyve. Det svarer til at forklare det operationelle princip. I denne rapport tilfælde er det, hvordan Blockly programmet virker og hvordan robotten Fable fra Shape Robotics virker. I dette trin, vil vi gå igennem Blocklys opbygning, men også redegøre og analysere hvordan blokprogrammering kan programmeres til en robot og hvilke mekanismer der skal til, før det sker. Vi bruger dette trin i vores analyse, til at sammenkoble de indre mekanismer og processer, med anvendelsen af blokprogrammering i undervisningen.

Trin 2: Teknologiske artefakter

I TSA bruger vi en metode om teknologiske systemer og artefakter. Den ene kommer fra TRIN-modellen. I TRIN-modellen definerer vi hvordan artefaktet, i dette tilfælde programmet, er påtænkt som en funktion. Hvilke egenskaber programmet og artefaktet har og hvad gør det helt præcist, til et teknologisk artefakt. Det er det vi finder ud af i dette punkt i modellen. Der snakkes om et samlet teknologisk system, som kendetegner, at produkter ofte ikke optræder alene. (Jørgensen og Yoshinaka, 2009) Derfor har vi to forskellige teknologier, som samlet giver produkterne, fra virksomheden Shape Robotics. Artefaktet kan også måles op i hvordan selve computerprogrammet fungerer, og det bliver også målt i selve robotten, som et teknologisk artefakt, da de begge har en grundlæggende teknisk funktion. Robotten og programmet kan passende deles op i materielle og immaterielle teknologier. Vi vil derfor fokusere på begge teknologiske systemer i dette trin. (Jelsø, Teknologiske systemer og artefakter 1 kursusgang 2, 2018) I trinnet teknologiske artefakter, findes

der to begreber: intentionalitet og funktion. De to begreber skal gerne hænge sammen, som producenten har tiltænkt. Intentionaliteten og den forbundne funktion, hænger sammen med de indre mekanismer og processer for vores teknologiske artefakt, for at realisere funktionen. (Jelsøe, Teknologiske systemer og artefakter Kursusgang 4, 2018) Dermed hænger første trin i TRIN-modellen og dette trin sammen. I flere tilfælde opfylder intentionaliteten og funktionen ikke altid kravene fra producenten, idet at det teknologiske artefakt ikke altid får den funktion, den var tiltænkt.

Selve begrebet computational thinking, er noget Lars især har taget til sig. Lars mål med sine elever, er at kunne lære at bruge de 4 nøglebegreber. Dekomposition, mønstergenkendelse, abstraktion og algoritmer. Begrebet computational thinking appalerer også til at kunne tænke anderledes og i andre baner, end i programmeringssproget. I programmeringssproget er det klart en fordel at have disse kompetencer, men det strækker sig også ud til andre skolefag. Der er her vi gennem vores interview med Lars Slynborg og workshop-oplæg hos Shape Robotics, kunne drage paralleller mellem. Hos Shape Robotics, fortalte Maja hvad hele læringsprocessen er bygget op om. ” Det hele bygger på det her konstruktivisme, der er hele det her mindset med eleverne får ikke idéerne de skal skabe dem selv. Viden bliver ikke overført fra lærerne men de skal selv kontrollere den.” (Pedersen & Kiellberg, 2019)

Trin 3: Teknologiers utilsigtede effekter

Det sidste vi skrev om i trin 2, var om den tiltænkte funktion også bliver udlevet for slutbrugeren.

Det hænger en smule sammen med trin 3, om teknologiers utilsigtede effekter. Her drejer det sig om de negative effekter der kan opstå ved en teknologi. Det kan f.eks. være støj fra vindmøller der skyldes de indre mekanismer, samt at de kan ødelægge udsigten.

Utilsigtede effekter vil i dette tilfælde opstå ved dårligt design, få opdateringer og fejl i koden. Vi kender det alle fra arbejde med programmering, at der skal meget små fejl til at hele systemet bryder sammen. Dette kan også være en utilsigtet effekt.

I det store perspektiv vil der også være nogle psykiske aspekter af dette. Børn kan sidde stille for lang tid foran skærmen, kan blive trætte i øjnene osv.

Der kan være flere ting som programmet har som utilsigtede effekter. Det er dog også her vi har behov for eksperterens indsigt, da de har arbejdet med de specifikke programmer og børn som bruger dem.

Trin 4: Teknologiske systemer

Trin 4 om teknologiske systemer samler op på trin 2, om teknologiske artefakter. Definitionen er at teknologiske systemer er sammenhængende systemer af teknologiske artefakter, som samlet besidder en bestemt funktionalitet (Jelsøe, Teknologiske systemer og artefakter Kursusgang 4, 2018). Det betyder at de teknologiske artefakter, kan sammensættes på forskellige måder, så der opstår noget nyt. Vores program opstår ved hjælp af flere forskellige sammensætninger af kode og grafiske værktøjer. I programmeringsverdenen findes der flere typer af programmering. Det er blandt andet interaktiv programmering. Det er flere sammensætninger af kode, der får brugeren til f.eks. at klikke på noget, hvorefter der sker noget nyt. Det er her der opstår en større sammenhæng og læring i programmet. De Vries taler her om skilningen mellem den fysiske beskrivelse af et system, som et samlet sæt af dele, der arbejder sammen. (Jelsøe, Teknologiske systemer og artefakter Kursusgang 4, 2018). I vores program, ser vi i dette trin på, hvordan programmet 'Fable Blockly', kan ses som et teknologisk system, med blandt andet robotten, men også med de forskellige komponenter og muligheder i bloksproget.

Trin 5: Modeller af teknologier

I dette trin bliver der kigget på modeller inden for teknologien. Dette udspringer især fra teknologien, men også generelt om hvad en model er. En model er en abstrakt, visuel eller fysisk repræsentation af et fænomen eller en genstand. Disse skal samtidig være undersøgt og/eller gengivet. Der kan skelnes mellem simple og abstrakte modeller. Under dette findes der så visuelle modeller. De kan bruges til mange tekniske projekter, og er typisk fokuseret på rumlige aspekter i 2D og 3D. Fysiske modeller er lavet fra materialer der skal repræsentere et design, eller et fænomen. De fysiske modeller er også gode til at teste og afprøve materielle prototyper.

Dernæst findes der tekniske modeller. Disse er det kvantitative af et design eller et fænomen. Der er mange måder at lave numeriske modeller i virksomhedsøkonomiske sammenhænge. Et godt eksempel til dette trin kan være billedesign. Der kan laves mange forskellige modeller at disse, da

modellerne skal kunne fastlægge arbejds- og designprocessen. (Jelsøe, Teknologiske systemer og artefakter Kursusgang 4, 2018). I vores projekt, arbejder vi med et software programmeringsprogram. Selve programmet er konstrueret som et blokprogrammeringssystem. Vores program er ikke beregnet til at kunne lave modeller af teknologier, på traditionelt vis, som TRIN-modellen beskriver. Det er det ikke, da det er en robot, programmet skal kunne styre og kontrollere.

Trin 6: Drivkræfter og barrierer for udbredelse af teknologier

I det sidste af de mere teoretiske punkter i TRIN-modellen finder vi drivkræfter og barrierer, for udbredelse af teknologier. Der er her meget fokus på innovationsteorier og typer af innovation.

Det er væsentligt, at for at bruge begrebet innovation, skal det i anvendelse. Det kan være nye tekniske løsninger, nye produkter eller rutiner. *"Diffusion is the process by which an innovation is communicated through certain channels over time among the members of a social system."*

(Rogers, 1983). Dette citat kommer fra Everett M. Rogers bog om diffusion. Diffusion omkredser centralt dissemination, som betyder spredning og deling af information til hinanden. Vi vil i dette trin også analysere Rogers 5 faser, også kaldet persuasion-fasens fem karakteristika.

Kommunikationskanalerne er lige så afgørende i dette trin. Dette er afgørende for innovationen og selve forståelsen. Til workshoppen hos Shape Robotics, fik vi et kort indblik i hvordan de bruger begrebet "innovation" i deres virksomhed og i deres produkter. Da de stadig er en relativ ny virksomhed, leder de hele tiden efter forbedringer i deres Fable Blockly program. Lærere og undervisere finder løbende ud af, hvor Fable Blockly har deres styrker og svagheder. Det er her vi i analysen vil gå ind i begrebet diffusion, da hele kernen i virksomheden handler om kommunikation fra elever, til lærere og tilbage til Shape Robotics, så de hele tiden kan forbedre læring, samt deres produkter.

Diskussion af TRIN-modellen:

I metodeafsnittet tager rapporten udgangspunkt i TRIN-modellen og i stakeholderanalysen. TRIN-modellen tager udgangspunkt i TSA1, som er et kursus der er på første semester. Metoden er lavet af egne lektorer på Roskilde Universitet, og er derfor til internt brug. Selve metoden er derfor svær at finde kilder på, da alle trin er lavet ud fra akademiske rapporter fra andre universiteter, og fra

forskere der både er eksterne og interne. De lektorer der har lavet metoden, har derfor også selv defineret de enkelte trins begreber. Det er derfor ikke andre kilder der kan bakke op om beskrivelserne af de enkelte trin i metoden. Metoden er endvidere bygget sådan op, at den skal kunne oplyse og analysere store som små teknologiske systemer og artefakter. I rapporten bruger vi hovedsageligt trin 1, 2, 3 og 6. Vi kommer dog hurtigt ind på trin 4. Grunden til at vi bruger de 3 første trin, er at trinene er velegnede, til at analysere selve processen og argumenterne for blokprogrammering. Trin 1 lægger også op til at beskrive hvordan de indre mekanismer og processer fungerer. Her er metoden bygget meget redegørende op. Her lægger vi vores fokus på, hvordan blokprogrammering opstod og hvilke teorier der ligger bag, samtidig med at vi bruger trinnet til at beskrive, hvordan Fable robotten virker og hvordan programmeringsprogrammet kommunikerer med robotten. Metoden har trin 4 og 5, som vi delvist udelukker. I trin 4 analyserer vi hvordan det teknologiske system fungerer i Blockly og hvilke problemstillinger der opstår ved disse systemer. Trin 5, om modeller af teknologier, udelukker vi i analysen, idet det ikke er relevant for vores rapport. Vi beskriver i de første to trin hvordan block programmeringen fungerer og ser ud som teknologisk model.

Stakeholderanalysen

Dette projekt tager udgangspunkt i programmering i folkeskolen. Dette gør, at der er forskellige aktører involveret. For at give et indblik i, hvilke aktører der er i inden for programmering i folkeskolen, vil denne rapport udføre en stakeholderanalyse. Stakeholderanalysen skal med sine redskaber, overskueliggøre hvilke aktører der kan være, inden for dette felt.

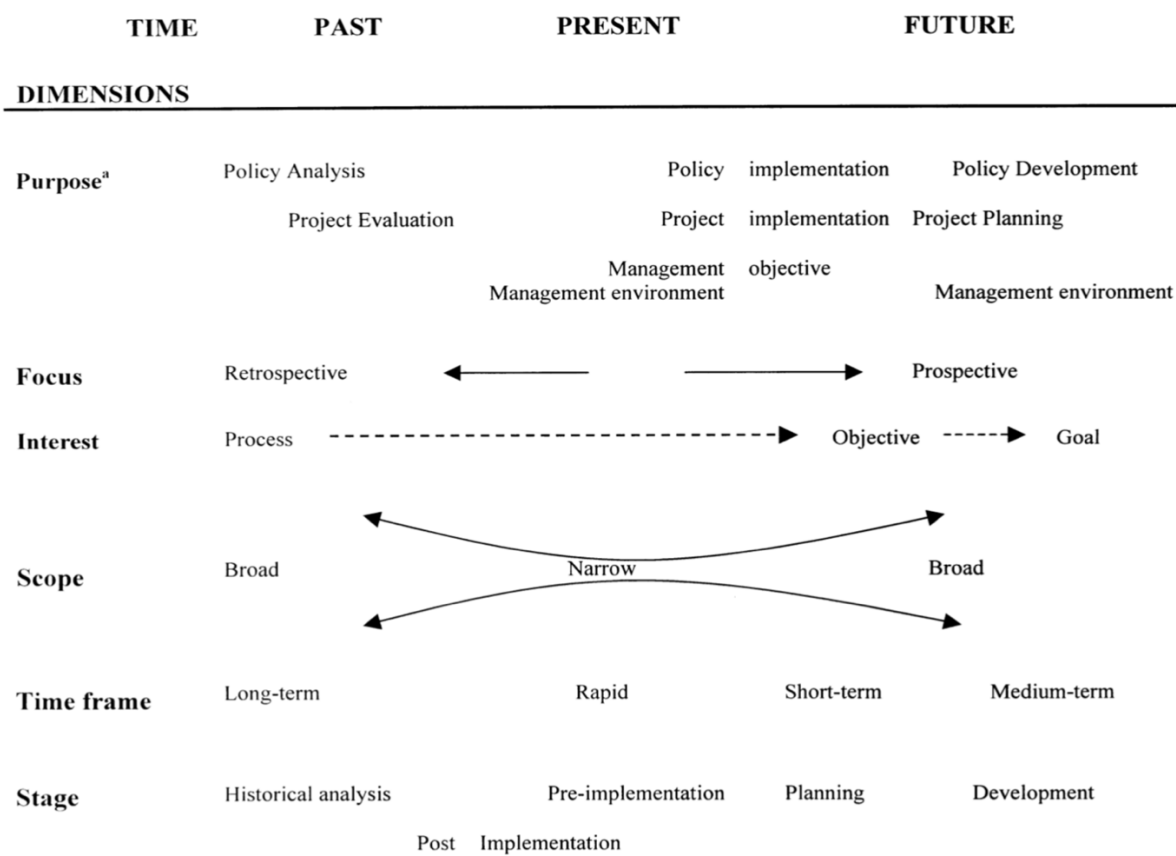
Hvad er det?

Stakeholderanalyse er et værktøj, hvis formål er at få viden omkring forskellige aktører. Disse aktører kan være individer, grupper og organisationer. Hovedformålet med en stakeholderanalyse er at analysere hvilke interesser der er, inden for et givent område og hvorfor den bestemte aktør kan have denne interesse. For at fastslå dette, er det vigtigt at finde ud af hvilken interesse der er, hvad har indflydelse på aktørernes standpunkt, hvilke relationer er der, hvilke netværk og mange andre karaktertræk. Analysen kan bruges i mange forskellige sammenhæng, som blandt andet implementering af nye projekter og indsættelse af ny ledelse (Varvasovsky & Brugha, 2000). Udbyttet af analysen, kan bruges som et hjælpemiddel til at forstå, hvordan implementeringen af et projekt eller en politik foregår. Derudover kan analysen også give et indblik i hvordan strategier, i for eksempel firmaer eller kommuner skal se ud. Det kan være rigtig svært at fastlægge en bestemt strategi. Derfor kan en stakeholderanalyse med de analyseredskaber der er udviklet, hjælpe med

netop dette. Selve analysen tager højde for tid, da mange strategier skal udføres over tid. Derfor tager stakeholderanalysen udgangspunkt i fortid, nutid og fremtid.

Selve analysen kan udføres af enkeltpersoner, et team eller en uvildig person der har den centrale viden, inden for det relevante emne (Varvasovsky & Brugha, 2000).

Stakeholderanalysen er bygget op af mange forskellige faktorer, hvis formål er at udvinde information. Den kan også være på mange forskellige niveauer. Alt fra lokalt, regionalt, nationalt og internationalt.



(Varvasovsky & Brugha, 2000)

Purpose

Formålet er udgangspunktet. Her grundlægges hvad analysen skal bruge til. Som alle analyseredskaberne i stakeholderanalysen, skelnes der mellem fortid, nutid og fremtid. I fortiden

undersøges politisk analyse og projektevaluering. Dette gøres for at grundlægge relevant viden.

Bemærk at politisk analyse er længst til venstre.

I midten ser vi nutiden. Her skal leder miljøet fastlægges, efterfulgt af ledernes mål. Efter dette kan projektimplementering og politisk implementering foregå. Ved dette trin, er det vigtigt at implementere et givent projekt og politik, ud fra et ledelsesperspektiv. I fremtiden er planlægningsfasen for projektet og derefter skal ledelsesmiljøet etableres, derefter kan politikken etableres.

Dette redskab giver en plan for implementering af et projekt. Derudover kan ledelsesstrategi og politisk implementering blive analyseret. Dette felt giver nogle redskaber til at undersøge de forskellige aktører enten fortid, nutid eller fremtid (Varvasovsky & Brugha, 2000).

Focus

I denne dimension bliver det skelnet mellem retrospektiv og prospektiv. Dette er med til at give information om aktørernes interesse.

Interest

I denne dimension bliver aktørernes proces beskrevet ud om de er i process fasen, objektivfasen eller målfasen.

Scope

Denne dimension fortæller om aktørerne har brede eller snævre interesser.

Time frame

Denne dimension beskriver om aktørerne har en langsigtet, hurtig, kortsigtet eller medium plan. For eksempel kan en el-bil producent have et langsigtet ønske om at flere brugere skifter over til el-biler.

Stage

Denne dimension bliver delt op i fire grupper. Dette er en historisk analyse, før implementering, planlægningsfase og udviklingsfase. Som det kan ses på billedet, er det relevant hvor de forskellige stadier er placeret i forhold til fortid, nutid og fremtid.

"Its scope can range from broad with a strong retro- spective dimension, with the aim of understanding the policy context and processes; to working towards a more immediate, often well-defined and focused policy implementation goal; to prospectively outlining more long-term and broadly focused policy directions" (Varvasovsky & Brugha, 2000, s. 6).

Selve analysen giver redskaber til hvordan forskellige dimensioner kan arbejde sammen og på den måde kan analysen udvinde information om aktørerne er langsigtede og prospektive med bred politisk implementering.

Semistruktureret interview:

Interview metode/guide

Interview udført af:	Emil Just Bluhme, Christian Wintcentsen
Personer interviewet:	Lars Slyngborg, 48 år, naturfagsvejleder og underviser
Dato for interview:	25/04/19
Andre relevante oplysninger:	For at sikre et kvalitativt interview, har vi oplyst interviewede om emner vi gerne ville tale, om inden interviewet

(Schødt, 2019)

Vi har valgt at benytte os af et semistruktureret interview som interviewmetode. Vi gik ind til dette interview uden at have den største baggrundsviden omkring emnet, som vi skulle interviewe om. Med denne form for interviewmetode, gav det vores interviewperson, et vis rum til fri udfoldelse og plads til at komme bredt omkring vores emne, uden at være bundet af visse rammer. Vores formål med denne metode var, at personen vi interviewede ville komme mere frit rundt om emnet, for netop også at give os muligheden for at få en bredere forståelse, for hvad vores emne indebærer. Ved at benytte denne interviewmetode, ved vi ud fra vores manglende baggrundsviden selvfølgelig heller ikke, om vi egentlig får de mest væsentlige faktorer med inden for emnet, hvilket er en umiddelbar ulempe ved denne interviewmetode.

Vores valgte interviewmetode tager dele fra både det strukturerede interview, samt det ustruktureret interview. Ved denne interviewform, får man dele fra det ustruktureret interview i form af de lidt mere frie rammer, der som sagt giver plads til bred forklaring. Yderligere får man stadig nogle retningslinjer med, fra det struktureret interview, i form af nogle forholdsvis fastlagte spørgsmål om emnet. Dette gjorde at vi var i stand til at få nogle brede besvarelser, men stadig være i stand til at holde den interviewedes svar, inden for de rigtige rammer.

Man får derfor med denne interviewmetode, et interview med plads til bredere og lidt mere frie svar og forklaringer, men der er dog stadig nogle retningslinjer, som sørger for at selve interviewet holder sig til det, man gerne vil have ud af det (Schødt, 2019).

Som sagt var vores viden omkring emnet begrænset, før vi gik ind til dette interview. Derfor var vores mål og forhåbninger med dette interview udover at få nogle udtalelser og meninger omkring dette emne, også at få en bredere forståelse, for hvad dette emne egentlig omhandler og indebærer.

Vi har valgt at udføre et semistruktureret interview, som havde til formål at give os bedre indblik i elevernes og ikke mindst lærernes oplevelser med implementering af programmering i folkeskolen. Vi har i den sammenhæng fundet en skole, som netop har forsøgt sig med denne implementering. Her har vi været i kontakt med en lærer, som vi efterfølgende har gennemført et interview med. Første del af interviewet var meget basalt, om hvorfor de på denne skole har valgt at prøve at implementere programmering i undervisningen. Her ledes der senere over til, hvilke kompetencer som menes at blive givet til folkeskoleeleverne, gennem denne undervisning. Derefter spørger vi ind til, samt får set en demonstration af, hvilke værktøjer der bruges til denne undervisning af programmering, på de forskellige klassetrin. Ydermere spørger vi ind til, hvilke metoder der bruges, når sådan en type undervisning skal praktiseres, samt hvordan det har været som lærer, at skulle begive sig ud i at undervise om programmering. I anden del af interviewet, spørger vi ind til, hvordan elevernes reaktion har været på at blive introduceret til denne form, for lidt utraditionel undervisning. Interviewet fandt sted i et lokale på den givne skole. Vi blev dernæst inviteret til at deltage, i et event om produkter til læring om programmering, som finder sted i Farum i starten af Maj.

Vores udbytte af interviewet var rigtig godt. Disse frie rammer, som kom igennem vores interviewmetode gjorde, at vi fik nogle brede besvarelser, som gav os et rigtig godt indblik i hvad emnet drejer sig om og berører. Det var tydeligt at vores interviewperson brændte for emnet, samt at fortælle omkring hvordan han og hans institution har oplevet denne implementering.

Vi har derefter besøgt Shape Robotics, hvor vi deltog i et møde med virksomheden, samt nogle forskellige repræsentanter og lærere fra diverse uddannelsesinstitutioner. Vi fik igennem dette møde et bedre indblik i, hvordan lærerne fra skolerne havde det i forhold til denne implementering af programmerings baseret undervisning. Yderligere fik vi snakket med Shape Robotics omkring deres motiver og budskab. Vi fik også selv lov til at afprøve de værktøjer, som de kan tilbyde skolerne. Til dette møde befandt vi os mere i en observerende position, hvor vi kunne høre omkring hvilke spørgsmål selve uddannelsesinstitutionerne havde omkring denne implementering. Vi fik lov til at optage diverse samtaler der var i mødelokalet, til brug af citater mm.

Analyse

Trin-modellen

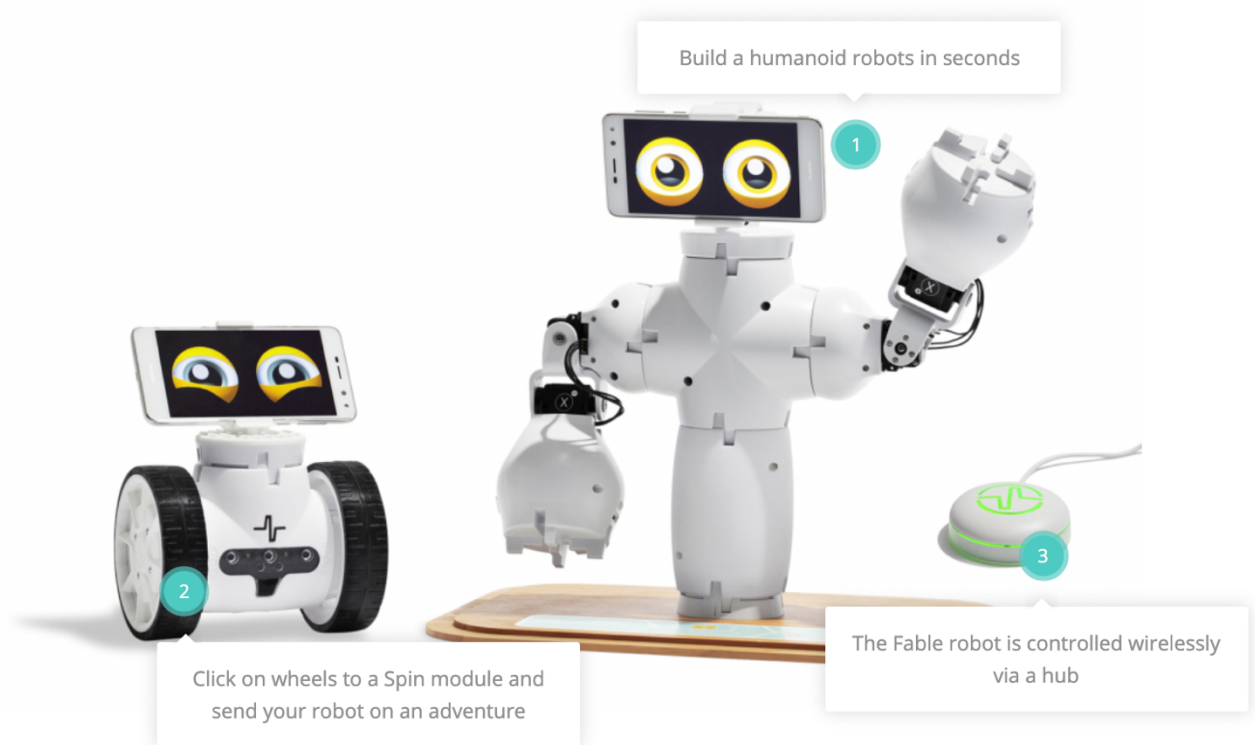
Trin 1: Indre mekanismer og processer

I forhold til Shape Robotics, som vores case drejer sig om, er der flere komponenter i dens indre system. For det første er der noget hardware, som vi vil redegøre for. Der findes 3 moduler på robotten. Det ene er et funktionelt modul, det andet er det passive modul og det tredje er noget de kalder styling editor. Det funktionelle modul er generelt opbygget, af to dele, som er deres X og Y-akser. X og Y delene, kan dreje i forskellige retninger, og fungere som "armen" på robotten. Det funktionelle modul kan sammenlignes med overarmen og underarmen, hvor midten og der hvor X og Y skiller, er albuen. Det passive modul er kroppen, man sætter X og Y armen på. De kan også dreje, lidt ligesom hvis man forestiller sig skulderen på et menneske. Det passive modul findes både med 3 og 4 indgange til armene. Modulet med 3 indgange er deres nyeste modul, hvor der kan placeres hjul. Ved begge typer passive moduler, kan der placeres en holder til telefon, hvorpå der kan designes øjne mm. Dette gør det mere menneskeligt at kigge på, og kan bruges som øjne til robotten. Energien til modulerne kommer gennem et batteri. I både det funktionelle modul og det

passive modul, er det placeret en trådløs router, der kommunikerer med det sidste modul, kaldet styling editor. Computeren skal igennem dens softwareprogram, sende data fra styling editoren til robotten, som så skal reagere herefter. Styling editoren er en lille rund anordning, der bliver forbundet til computeren ved hjælp af et almindeligt 2.0 USB-stik. Det interessante ved det hele er, at styling editoren og de andre moduler, skal have samme farve, før de er forbundet og parret med hinanden. Dvs. at når det funktionelle modul bliver tændt, skal den have samme farve som styling editoren, og det er det samme med det passive modul. Der findes her 6 forskellige farver. (Shape Robotics, u.d.).

Softwareprogrammet Fable Blockly, er deres foretrukne programmeringsprogram. Programmet hedder bare Brockly, og er udviklet af Google og udspringer fra programmet Scratch (Pedersen & Kiellberg, 2019).

Fable Blockly åbner med en tom side, hvor vi ser flere valgmuligheder i venstre side. Oppe til højre på siden står der "Program: Stoppet" der viser om programmet er startet eller ej. Under det står der "Hub: ikke forbundet", som betyder at style editoren ikke er tilsluttet og har valgt en farve. Under igen står der "Moduler: 0 fundet" og viser hvor mange moduler der er tilkoblet. Valgmulighederne i venstre side er "Mest anvendte, logik, handlinger, løkker, farver, funktioner, sanser, kamera, data, variable, lister og matematik." Alle mulighederne er forskellige blokke, med kommandoer som kan trækkes ind, i det tomme felt midt på siden. De kan herefter kobles sammen med flere og kæde flere bevægelser og kommandoer sammen. Som sagt udspringer hele programmet sig fra Scratch, og derfor har du i højre side hele tiden mulighed for at se den kode det bliver til.



(Shape Robotics, u.d.).

Nu har vi fortalt lidt om selve programmet og hvordan det ser ud. Fable Blockly er kun et navn Shape Robotics har givet til programmet, der oprindeligt bare hedder Blockly. Blockly er som sagt et blokprogrammeringsprogram, som går ud på at sammensætte blokke, som kan give forskellige outputs, alt efter hvad brugeren har lyst til. De tidlige programmeringsinitiativer til børn, var ofte introduceret og sat op af aktiviteter. Det kunne være en liste af primtal, for at lave simple linjer/streger. Dette vakte ikke betydelig gejst for børnene, da det var for svært og der kom ikke nok ud, af de hårde anstrengelser. Derudover var programmeringen ofte introduceret på en sådan måde, at der ikke var nogle lærere til at støtte og guide børnene, når noget gik galt, eller til at tilskynde nysgerrighed og prøve igen, hvis koden ikke virkede i første omgang (Resnik, 2009, s. 63). Seymour Papert argumenterer herfra, at programmet børnene skulle arbejde i, skulle have "low floor", som betyder at det skulle være nemt at starte op for nye. Det skulle ifølge Papert også have "high ceiling", der betyder at mulighederne for projekter, skal kunne være komplekse over tid. Til sidst skulle programmet ifølge Papert have "wide walls". Her betyder det at der skal være plads til projekter med forskellige interesser og læringsredskaber. De her tre begreber brugte forskere og undervisere til at grundlægge Scratch.

”To achieve these goals, we established three core design principles for Scratch: Make it more tinkerable, more meaningful, and more social than other programming environments.” (Resnik, 2009, s. 63)

Google tog efterfølgende samme råd fra Papert, og designede programmet Blockly.

Blokprogrammering fungerer meget som at lege med Lego. Når børn får lego i hånden, begynder de hurtigt at kunne sammensætte klodser, med forskellige farver og former. Når børn bygger lego, giver det dem løbende nye idéer om nye sammensætninger og metoder at gengive en historie. Når børn leger med Lego, kan begrebet ”low floor” især bruges.

Trin 2: Teknologiske artefakter

Vi vil i dette punkt definere om programmet får den tiltænkte funktion, som evt. lærere og udviklere havde tiltænkt. Det sker ofte at anvendelsen viser sig at være forskellig fra det intendede. Det kan nemlig tænkes at nogle elever og lærere har en forskellig måde at opfatte hvordan indlæringen af programmering skal køre. Dette kaldes for fortolkningsmæssig fleksibilitet.

Shape Robotics robot har flere sammensætninger af artefakter. Som sagt er der tre dele der kan sammensættes. Funktionelt modul, passive modul og styling editor. Det funktionelle modul er et meget centralt artefakt i robotten. Den bruges til at dreje og rotere X og Y, som vi snakkede om i første trin af TRIN-metoden. Det funktionelle modul kan placeres flere steder på det passive modul og derfor kan udseende og funktion ændres en del, i forhold til hvor den placeres. Ud fra en problemorienteret opgave, skal eleverne kunne bygge og placere modulerne efter hvor de mener det kan løse problemet. I workshoppen i Farum d. 6. maj fik vi opgaven at kunne løse et hverdagsproblem. ”Hack et hverdagsproblem” gik ud på at brainstorme helt almindelige hverdagsoplevelser/hverdagsproblemer på små post-it papirer. Derfra skulle vi udvælge det bedste problem, som ikke behøvede at være realistisk, men som kunne løses. Vi valgte at robotten skulle kunne vande planter. Vi skulle så bygge robotten, der kunne løse vores problem rent fysisk. Vi valgte det passive modul, med to hjul, da robotten skulle kunne køre hen over gulvet og over til planten. Det passive modul fik to hjul på i hver sin side og et lille hjul bagerst. De to hjul i højre og venstre side var relativt store, så der kunne opretholdes en balance i modulet, samtidig med at det lille hjul skulle holde vægten som blev placeret bagerst da robotten ikke skal vælte når den kører. Vi koblede endnu et passivt modul på, som skulle gøre det lettere for det funktionelle modul at

dreje rundt og så ville højden på robotten også blive markant større. Derefter satte vi to funktionelle moduler i hver sin side af det andet passive modul. Begge funktionelle moduler fik hver en "hånd" på, som gjorde vi kunne få en vandflaske på. Meningen er så at robotten skal programmeres til at køre over til en given plante, når den skal have vand. Herefter skal den kunne hælde vandet ned i planten og køre videre til den næste. Koden skal se sådan ud, at når robotten er kommet over til planten, skal den kunne dreje "hånden" med vandflasken, og komme tilbage til udgangspunktet efter.

Selve programmet Fable Blockly er programmeringsprogrammet, som er skabt ud fra Scratch. Shape Robotics ville lave programmet med fokus på computational thinking. Det er hele visionen og idéen bag Fable Blockly og robotten. Lars Slyngborg fortæller at det ikke nytter noget at lære programmering på den gammeldags måde. Dvs. at fasit og "rigtige" svar skal sorteres væk fra undervisningen.

Det teknologiske artefakt er ofte brugt på andre måder end hvad hensigten er. Som eksempel kan man tage en lighter, man kan bruge til at åbne flasker med. Derfor spurgte vi Lars Slyngborg om dette kunne forekomme i folkeskolen, altså om programmet kan bruges anderledes end hvad hensigten er fra producenter og lærere. Spørgsmålet blev stillet til Shape Robotics og Lego Mindstorm, som de også brugte på skolen. Lars Slyngborg afviste rimelig hurtigt at det kunne bruges til andre hensigter. Han siger at hvis det kan bruges på andre måder, har man fundet en kreativ måde at bruge det anderledes (Slyngborg, 2019).

Når man snakker om anvendelsen, findes der begrebet intentionalitet og funktion. Intentionaliteten i et artefakt, hænger sammen med den påtænkte funktion. Ved dette artefakt, er den påtænkte funktion at alt kan lade sig gøre. Det er kun fantasien der sætter grænser her. For at komme med et eksempel på en helt anden måde at bruge teknologien på, sagde Lars Slyngborg at man i princippet kunne programmere robotten som en slags rulle-Marie og altså klare opgaver med bombe ryddere og militæret. Det er nok ikke hvad udviklerne havde tænkt den skulle bruges til, da det er leg og læring det er hovedformålet med Shape Robotics produkter.

Trin 3: Teknologiers utilsigtede effekter

Lars Slyngborg nævner omkring de utilsigtede effekter, at elever kan give op hvis det er for nemt eller for svært. Det er ofte kulturen i folkeskolen der gør det, men også de opgaver der bliver stillet

til robotterne. Programmet Blockly bliver også hele tiden opdateret, men har nogle begrænsninger idet at det er blockprogrammering. Til workshopen forklarede arrangørerne, at undervisere ofte skriver til dem, hvis de finder punkter hvor der skal opdateres i Blockly programmet. Lars Slyngborg fortæller også efter vi har stillet spørgsmålet om hvorvidt lærere føler sig klædt nok på til at undervise i programmering. Dette er en utilsigtet effekt, idet at Blockly programmet tager udgangspunkt i at løse opgaver, uden nogen videre vejledning eller opgaver. Selve begrebet computational thinking og Paperts teori om læring, lægger op til, at være kreativt og problemløsende. De problemstillinger er svære at løse, hvis børnene ikke ved hvordan systemet virker. Lærere skal derfor kunne gå ind og hjælpe børnene med mindre tekniske problemer hvis de opstår. Lars Slyngborg argumenterer dog imod dette. Han mener at lærere skal prøve at lære, sammen med eleverne.

”Noget af det der er vigtigt at der er rigtig mange lærere der ikke har noget viden om det her. At man faktisk går ind og så siger jamen vi ved sgu ikke så meget om det, men vi kan godt gå ind og undervise for vi lærer sammen med eleverne.” (Slyngborg, 2019, s. 3).

Lars Slyngborg har dog her en anden opfattelse end Majbritt fra teknisk skole, som vi snakkede en smule med til workshopen hos Shape Robotics i Farum. Hun siger, at 70% af de lærere der underviser sammen med hende, ikke kan gå ind i et lokale og ikke være klædt på til undervisning.

”Nu siger du jo det her med at lærerne ikke behøver at være klædt på, jeg kan så bare sige at 70 procent af mit lærerværelse har ikke lyst til at gå ind til en time og ikke være klædt på. Så det kan man ikke bare smide dem i hovedet, der bliver vi nødt til at gå ind som støtte. Enten at starte det op, for du får dem ikke til at gå ind og sige jeg har ikke styr på det.” (Pedersen & Kiellberg, 2019, s. 4)

Maibritt argumenterer her for, at hun ikke kan få lærere til at gå ind og sige de ikke har styr på det. Det som Lars argumenterer for at man godt kan. Under hele workshopen var der flere undervisere til stede, som gav os et indblik i hvordan deres skole/uddannelsessted, behandler programmering og problemorienteret undervisning. De fleste var klar over hvordan deres nuværende undervisningsmodel ser ud. Her nævner Maibritt at metoden gerne skulle skifte fra resultatorienteret undervisning til procesorienteret undervisning.

”I matematik altså der er meget korrekt eller ikke korrekt især nede i indskoling. Når det kommer på et højere niveau, så bliver det lidt anderledes. Det tror jeg bare er en del af vores skolekultur og det har været det i lang tid. Det har været resultatorienteret i stedet for processororienteret og det tager bare rigtig lang tid at ændre og det er jo især lærerne der skal ændre sig og det er jo en svær kultur at ændre.” (Pedersen & Kiellberg, 2019, s. 4)

Hun bruger lærerne som en stor del af det der skal ændre sig, før programmeringsundervisningen kan give de resultater, som Paperts argumenter for.

Trin 4: Teknologiske systemer

I trin 4 vi vil analysere Shape Robotics programmeringsprogram Fable Blockly og robotten som et teknologisk system. Der kan argumenteres for, at robotten og Fable Blockly har to helt forskellige teknologiske systemer. Shape Robotics vil dog gerne have, at man ser robotten og Blockly som et samlet system. Vores case drejer sig generelt om programmering i folkeskolen, og ikke om robotter. Derfor kommer vi til at fokusere mere på programmet Blockly. Det teknologiske system opstår, idet at der er flere komponenter der skal hænge sammen, før der opstår en visuel præsentation af hvad koden/blokkene, består af. Blockly består af nogle afgrænsningsproblemer, der er lodret. Det skyldes at programmet har en hierarkisk struktur. Dvs. at systemet kommer oppefra, og kan studeres på mange forskellige kompleksitetsniveauer. (Buhl, 2005, s. 10)

Generelt er programmering og kodning opbygget sådan, at hvis der i koden har en lille kommafejl, kan programmet ikke køre. Sådan er det teknologiske system bygget op omkring Blockly og kodning. Teknologiske systemer har ikke den store relevans i forhold til vores opgave. De ting rapporten kommer ind på, er ikke sammenhængende i forhold til teknologiske systemer. Derfor udelukker vi flere dele af dette trin.

Trin 6: Innovationer, Drivkræfter og barrierer for udbredelse af teknologier.

Vi vil i dette trin kigge på persuasion-fasens fem karakteristika i forhold til Fable Blockly.

“Everett Rogers har udviklet en teoretisk diffusionsmodel til, at forstå hvordan teknologier optages og spredes i et socialt system. En del af diffusionsmodellen er persuasion fasen, hvor Rogers mener, at fem karakteristika ved en teknologi er betydende for, om brugerne overtales til, at anvende den pågældende teknologi” (Jelsøe, Teknologiske systemer og artefakter 1 kursusgang 2, 2018).

Diffusion er processen hvor innovation er kommunikeret ud gennem bestemte kanaler over tid mellem borgere i et socialsystem (Resnik, 2009, s. 5). Altså er diffusion en speciel form for kommunikation hvori, at budskabet omhandler den nye ide (Resnik, 2009, s. 6). Som tidligere nævnt vil vi kigge på persuasion-fasen, der er en del af diffusionsmodellen - om persuasion-fasen skriver Rogers "The characteristics of innovations, as perceived by individuals, help to explain their different rate of adoption." (Resnik, 2009, s. 15)

De fem faser, som vi vil analysere Fable Blockly ud fra er:

1. Relative advantage: teknologiens relative fordel for den enkelte beslutningstager.
 2. Compatibility: er (opleves) teknologien som forenelig med det eksisterende?
 3. Complexity: er (opleves) teknologien som kompleks? (kræves oplæring/..).
 4. Trialability: er (opleves) det som nemt at afprøve teknologien?
 5. Observability: er det nemt at se resultaterne?
- (Jelsøe, Teknologiske systemer og artefakter Kursusgang 4, 2018).

1. Relative advantage:

Selve programmet Fable Blockly er programmeringsprogrammet, som er skabt ud fra Scratch. Shape Robotics ville lave programmet med fokus på computational thinking. Det er hele visionen og idéen bag Fable Blockly og robotten.

Hvordan er Blockly innovativt set i forhold til Scratch? En af funktionerne der er værd at nævne, er for eksempel det her med at kunne se koden på flere forskellige programmeringssprog. Dette gør det muligt at få et bedre kendskab til, hvordan at flere programmeringssprog ser ud, på en overskuelig måde.

2. Compatibility:

Er et program som Fable Blockly kompatibelt med folkeskolen? Fable Blockly kræver, at man har en computer og en robot - halvdelen af hvad det kræver for at kunne benytte sig af Fable Blockly,

har eleverne allerede (computeren), det er derfor kun robotten, som skolen skal sørge for. Computeren bliver i forvejen anvendt af elever i folkeskolen, hvorfor Fable Blockly fremstår som yderst kompatibelt, i forhold til en implementering.

3. Complexity:

Kompleksiteten af programmet er designet ud fra Paperts principper ("low floor" (nemt at komme i gang) "high ceiling" (muligheder for at kreere mere komplekse programmer over tid). Ideen bag Blockly er at skalere et komplekst felt som programmering ned til et visuelt programmeringssprog, der tager abstrakte koncepter, og gør dem til objekter børn kan forstå. Denne bagvedliggende design ide + en bred vifte af undervisningsmateriale på Shape Robotics egen hjemmeside gør, at eleverne står med et program, der er let at lære.

4. Trialability:

Shape Robotics tilbyder en gratis prøveperiode på 30 dage, hvor skolerne får tilsendt flere forskellige moduler til at samle robotten. Robotten kan herefter samles på flere forskellige måder, så eleverne kan udfolde sig på alle tænkelige måder, inden for robotens funktioner. Shape Robotics sender samtidig lektionsplaner, i klassetrin fra 3. klasse til 3.G. Fable Blockly kan gratis hentes gennem deres hjemmeside, og behøver ikke en robot til at afprøve. I programmet kan eleverne hurtigt få et overblik over mulighederne af de forskellige blokke og hvordan de kan sammensættes.

5. Observability:

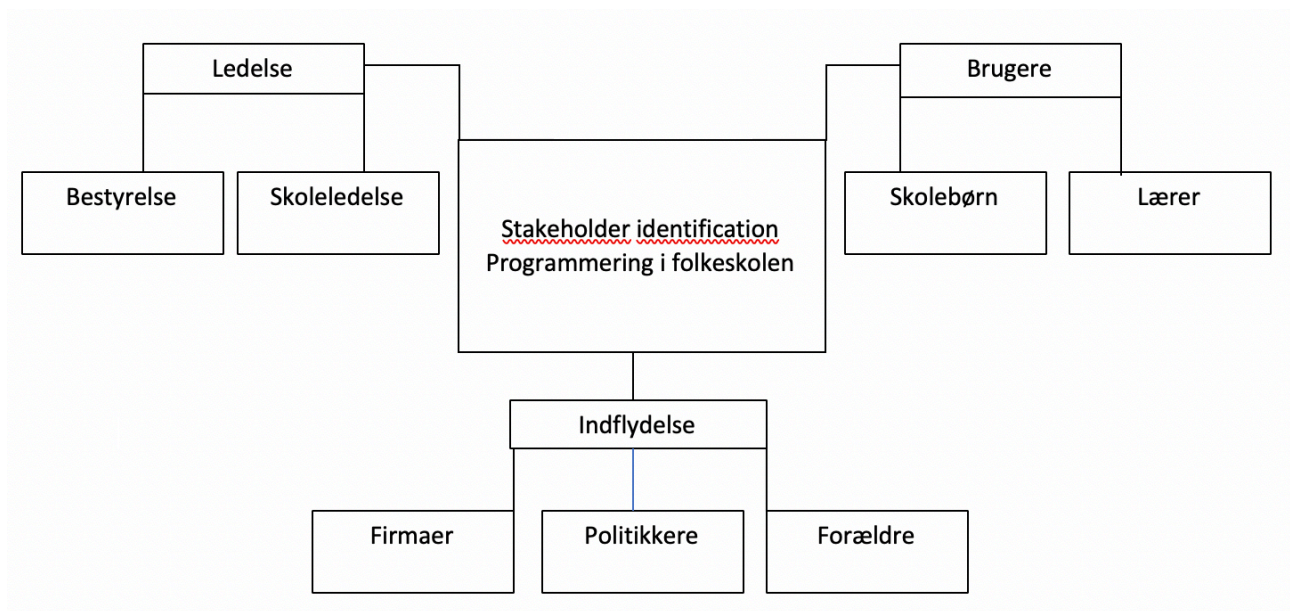
Lærere der har testet Fable Blockly af i undervisningen siger, at de ser en klar øgning af elevernes evner i forbindelse med problemløsning. Disse resultater er nogle, som er tydelige at se for andre, hvorfor det vil være inspirerende for andre uddannelsesinstitutioner at inddrage programmet i deres undervisning.

Stakeholderanalyse af folkeskolen med fokus på programmering

Aktører

I folkeskolen er der en lang række aktører. Dette kan blandt andet være forældre, børn, lærer, ledelse men også firmaer der vil have implementeret deres teknologi. Her kan Shape Robotics nævnes, da det er et firma der i høj grad er interesseret i folkeskolemarkedet. Det skyldes at robotten Fable henvender sig blandt andet til børn i skolealderen.

Nedenfor er forskellige aktører i folkeskolen delt op efter om aktørerne er brugere, ledelse og indflydelses aktører.



I denne stakeholderanalyse vil der blive fokuseret på lærere og Shape Robotics.

Dette skyldes at ud fra rapporten er det netop disse aktører der er interessante at analysere. Det bygges på den teori der er opsamlet gennem rapporten.

En af de firmaer, der har interesse i folkeskolen, er Shape Robotics. Shape Robotics er interessant at analysere som aktør, fordi de har en teknologi de gerne vil have implementeret i folkeskolen. Som nævnt tidligere i rapporten, er Shape Robotics et firma der er bygget op af blockprogrammering der

sammen med en fable robot giver et værktøj til at lære og programmere (Pedersen & Kiellberg, 2019).

Shape Robotics står overfor projekt evaluering. De er blevet implementeret i skoler som Sct. Jørgen skolen og har dannet samarbejde med coding pirates, som er det programmeringstilbud rettet mod børn i folkeskolealderen. Deres fokus er i prospektiv i og med at de har et færdigt produkt der skal udbredes. Deres scope er bredt fordi der egentlig ikke er én målgruppe, deres robot henvender sig til mange aldersgrupper. På workshop med Shape Robotics var der private interessenter, folkeskolelærere og lærere fra teknisk skole. Dette giver et godt indblik over hvor bredt deres produkt når ud. Fable robotten og Blockly, der er det tilhørende programmeringssprog (Pedersen & Kiellberg, 2019).

Shape Robotics har en langsigtet plan om, at få programmering i folkeskolen mere udbredt, end det allerede er (App academy, u.d.). Derudover er der også et problem med implementering fordi at lærerne ikke føler sig godt nok klædt på til at undervise med Fable robotten og andre programmeringsredskaber (Pedersen & Kiellberg, 2019).

Som firma er shape robotics med til at påvirke blandt andet folkeskolen fordi, de gerne vil implementere deres produkt i netop folkeskolen. Denne implementering vil i høj grad påvirke lærerne i folkeskolen.

Som undervisere er lærere en vigtig aktør. Dette skyldes at de ud fra deres faglige viden og læringsmål nedsat af regeringen, bestemmer hvilket pensum eleverne skal op i. Som nævnt er der nogle lærere der ikke kan lide at undervise i noget som de ikke har styr på (Pedersen & Kiellberg, 2019). Dette er et problem for implementering af programmering i et bredere perspektiv, da det kan variere fra folkeskole til folkeskole alt afhængig af lærerne der er ansat på skolen. (Pedersen & Kiellberg, 2019).

”Rigtig mange snakker om programmering som et mål, vi skal lære og programmere vi skal programmere vi skal være brugere vi skal være en hel masse ting” (Slyngborg, 2019, s. 1).

I et interview med en folkeskolelærer beskrives det, at programmering ikke skal ses som et mål, men som en læringsmetode som lærerne kan benytte.

Der findes mange forskellige måder at lære og programmere og der findes et hav af programmeringssprog. Det kan være for eksempel være java, python, javascript og mange andre

(Tiobe, u.d.). Lærerne som aktører er interesseret i at vide hvad de skal undervise i. Om det er H.C. Andersen eller programmering med en Fable robot, så er visheden om emnet vigtigt for mange af lærerne (Pedersen & Kiellberg, 2019). Derudover er der også stor utilfredshed over manglende forberedelsestid (Mainz, 2006).

Lærernes formål kan være politisk ændring både lokalt, kommunalt, regionalt og nationalt. Dette formål kan nemlig variere fra skole til skole, fra kommune til kommune eller at gøre det ens nationalt.

Implementeringen kan være en læringsplan, der er udarbejdet ud fra bestemte programmeringsredskaber. Dette skal dog også tilpasses efter klassetrin da der læringsmæssigt er forskellige krav i henholdsvis indskoling, mellemtrin og udskoling (App academy, u.d.).

Shape Robotics har et bredt fokus med implementeringen af deres produkt. De bevæger sig også retrospektivt, dette skyldes at på deres hjemmeside kan en lang række undervisningsforslag ses. Undervisningen bygger på eksisterende undervisning som for eksempel pythagoras, geometri og musik (Shape Robotics, u.d.).

Diskussion

Denne rapport har belyst programmering i folkeskolen. Efter analyse samt inddragelse af primær- og sekundær kilder, er der blevet skabt et billede af, hvordan programmering i folkeskolen kan implementeres.

I interviews med Shape Robotics og folkeskolelærer Lars Slyngborg, blev der indsamlet viden omkring folkeskolen. Dog er det også blevet belyst i rapporten at folkeskolelærerne ikke har lyst til at undervise i programmering, da ikke alle lærer føler sig udrustet sig til at undervise i det (Pedersen & Kiellberg, 2019). En af de deltagere fra interview med Shape Robotics sagde følgende:

”Jeg kan så bare sige at 70 procent af mit lærerværelse ikke har lyst til at gå ind til en time og ikke være klædt på. Så det kan man ikke bare smide dem i hovedet, der bliver vi nødt til at gå ind som støtte. Enten at starte det op, for du får dem ikke til at gå ind og sige jeg har ikke styr på det”

(Pedersen & Kiellberg, 2019, s. 5).

Det blev også berørt i stakeholderanalysen at lærere som aktører har interesse i at blive klædt bedre på til programmeringsundervisning. Dette gælder om det er Blockly eller linje kodning. Især i et felt som bevæger sig så hurtigt som programmering gør.

Der er dog delte meninger om hvordan det skal ske. I interview med Lars Slyngborg siger han følgende:

“Man er ikke nødt til at være et geni for at kunne programmere i folkeskolen eller i at lære og bruge robotter. Det som vi voksne kan er, at vi kan gå ind og fejlfinde. Eleverne kan hurtigt komme ind og sige ”så lavede vi det her program” men så når det til et eller andet sted, hvor de sidder fast i deres process, og der er det tit, at vi som voksne går ind og siger: har du tænkt på eller har i overvejet at gøre sådan og sådanE”

(Slyngborg, 2019, s. 3).

Dette citat giver en indikation om er at der er en stor forskel på, hvad lærerne egentlig ønsker. Det indikerer også at der findes lærere, der gerne vil springe ud i at lære noget nyt som programmering og andre som ikke har lyst eller tid til det.

Fra politisk side er der også blevet skåret i mængden af forberedelsestid, så lærere kun har 15 minutters forberedelse til hver lektion hvilket er 40 timer ugentligt (Mainz, 2016). Dette lyder umiddelbart af meget tid, dog skal denne tid også bruges til diverse interne møder samt forberedelse til undervisning. Så ekstra tid på at sætte sig ind i enten Fable robotten eller Blockly vil gå ud over denne tid. Dette kan også være et problem, idet Shape Robotics metoder går ud fra at lærere springer ud i programmerings undervisning. Computational thinking er nøgleord hos dem, idet deres produkter bliver meget mere relevante hvis læringsmetoderne ændrer sig til den problemløsende tankegang. I TRIN-modellen finder vi det væsentligt hvordan lærerne arbejder med denne tankegang, sammen med Blockly. Styrkerne ligger her hos Seymour Paperts teorie om lige præcis denne måde at undervise kan hjælpe eleverne på tværs af alle fag.

Visuel præsentation

Nedenfor er vores poster i forhold til den visuelle præsentation af projektet illustreret. Denne poster indeholder nogle forskellige keywords inden for vores projekt, samt vigtige begreber inden for programmering i folkeskolen. formålet med denne poster, er at vække interesse og give information, om vores projektrapports indhold.

PROGRAMMERING I FOLKESKOLEN

FABLE/BLOCKLY

UNDERVISNINGSVÆRKTØJ

Er et program, som er designet til undervisning af programmering i folkeskolen. Dette program giver eleverne en mulighed for at udfolde sig kreativt, igennem et mere overskueligt værktøj, som samtidig kan hjælpe med at give en forståelse for programmering .

COMPUTATIONAL THINKING

Computational thinking er en af de mange kompetencer, som eleverne kan tillegne sig ved at arbejde med programmering. At lære at arbejde med denne tankegang giver eleverne et form for værktøj til at se på et problem og komme med løsninger .

PÅKLÆDNING TIL FREMTIDEN

Ved at udruste eleverne med denne viden, vil de også være bedre forberedt til at tackle fremtidens problemer på arbejdsmarkedet.

SEYMOR PAPERT

Seymour Aubrey Papert var en matematiker, datalog og underviser, som primært tilbragte sin karriere på at undervise og forske på MIT. Papert ses som værende en af verdens eksperter, når det kommer til, at undersøge hvordan teknologi kan bidrage til nye måder hvorpå man kan lære og undervise i matematik og generel tænkning.

KONSTRUKTIONISME

Konstruktionismens fokus på læring gennem skabning hjælper os med at forstå "how ideas get formed and transformed when expressed through different media, when actualized in particular contexts, when worked out by individual minds"

AUTOMATISERING

Verden bevæger sig længere og længere hen imod en mere og mere automatiseret hverdag. Ifølge The International Federation of Robotics er markedet for industrielle robotter steget med mere end 60 procent fra år 2000 til 2012, med samlet salg på omkring 28 milliarder dollars i 2012, hvor kina er det hurtigst voksende marked med en vækst på omkring 25 procent årligt fra 2005 til 2012.

Konklusion

Hovedemnet i denne rapport er at undersøge vigtigheden ved, at fremme læring indenfor programmering. Den problemstilling er belyst med en række arbejdsspørgsmål, der ved hjælp af andre rapporter samt artikler til at analysere vigtigheden i at lærer og programmere.

Rapporten belyser ud fra interview med Lars Slynghborg samt en lydoptagelse fra en workshop med Shape Robotics, at programmering med en problemorienteret læringstilgang har en gavnlig effekt.

Ud fra denne rapport kan det konkluderes, at der er klare fordele ved implementering af programmering i folkeskolen. Denne rapport har taget sit udgangspunkt ud fra casen Shape Robotics "hack et hverdagsproblem". "Hack et hverdagsproblem" er en problemorienteret læringsmetode, der giver gode kompetencer, og disse kompetencer er gode at tilegne sig, i en stigende automatiseret hverdag. Som påvist i denne rapport bliver computational thinking i stigende grad udbredt. Derfor kan det konkluderes, at folkeskoleelevers kompetencer kan blive øget ved, at udføre sådanne øvelser.

Det kan konkluderes ud fra rapportens stakeholderanalyse at lærerne og firmaer som Shape Robotics er vigtig for at fremme programmering i folkeskolen. Lærernes ansvar i folkeskolen er at udbrede læring til børn, og derfor er deres andel i dette altafgørende.

Et kritikpunkt i rapporten er, at stakeholderanalysen kun analysere et begrænset antal af de aktører, der er til stede. Grundet den komplekse natur der er omkring folkeskolen, er det svært at analysere alle aktøreres bevæggrunde.

Evaluering

Ser man tilbage på hele vores projektperiode, er der selvfølgelig ting vi kunne have gjort bedre. Vi var alle sammen forholdsvis sikre på emnet, men det at vi ikke havde en synderlig stor baggrundsviden om dette, gjorde at vi var nødsaget til at undersøge en masse ting, uden helt at vide hvor vi skulle starte. Vores plan var at få et interview i hus, som noget af det første, også for at få en dybere indsigt i, hvad emnet egentlig indeholdt. Idet at vores interview blev lidt forsinket, blev selve vores projekt også en anelse udskudt. Vi har igennem hele skriveperioden været gode til at snakke sammen om, hvad vi hver især var i gang med at udarbejde, men set tilbage skulle vi nok

havde sat os sammen og udarbejdet noget mere i fællesskab, i stedet for bare at uddele opgaver, som folk satte sig til at lave alene. Vi har igennem hele forløbet været gode til at holde struktur på vores arbejde og haft styr på hvem, som havde gang i hvad. Vi har dog også stået i den situation, at eksamener har endt med at tage en del tid fra selve projektskrivningen, så denne er endt med indimellem at blive lidt tilsidesat.

Alt i alt føler vi at det har været en lidt stresset, men lærerig projektskrivning med fin struktur, opgavefordeling og samarbejde. Da dette projekt kun er vores andet projekt i vores uddannelsesforløb, var der stadig nogle elementer, som vi stod tvivlende overfor - i den forbindelse har vores vejleder været yderst behjælpelig. Disse elementer var for eksempel; konkretisering af problem, metodikker til at fremfinde relevant læsestof og generel sparring i forhold til projekt.

Som motivation for projektet var, blandt andet, det her med, at vi selv har været i berøring med programmering på første og henholdsvis andet semester. I disse forløb har vi kunne mærke, at vi har tilegnet os en anderledes tilgang til problemløsning. Projektet har bidraget til et større kendskab til, hvad der egentlig har foregået, hvilket har gjort det muligt, at sætte ord på hvad vi egentlig har lært ved programmering. Computational thinking.

Afslutningsvis vil vi tilføje, at hvis man ser bort fra en sideløbende eksamensperiode med deadlines, har projektet været både interessant og lærerigt.

Litterateur

- Ackermann, E. (2001). *Piaget's Constructivism, Papert's Constructionism: What's the difference?*.
App academy. (u.d.). Hentet fra <https://appacademy.dk/forborn/hvorfor-skal-man-laere-om-programering-i-folkeskolen/>
- Blackwell, A. F. (2002). *What is programming*.
- Blockly. (u.d.). Hentet fra <https://developers.google.com/blockly/>
- Brugha, R., & Varvasovsky, Z. (2000). *Stakeholder analysis: a review*. Oxford University Press.
- Buhl, H. (2005). Busenderen. Valdermar Poulsens radiosystem. I H. Buhl. Aarhus Universitetsforlag.
- Caeli, E. N. (2018). *At lære programmering i skolen er ikke nok – og det vidste vi også for 50 år siden*.
- Den store danske. (u.d.). Hentet fra http://denstoredanske.dk/Samfund,_jura_og_politik/%C3%98konomi/Produktion,_investering_og_%C3%B8konomisk_v%C3%A6kst/innovation, lokaliseret d. 25.04.19
- Den store danske . (u.d.). Hentet fra http://denstoredanske.dk/Samfund,_jura_og_politik/%C3%98konomi/Produktion,_investering_og_%C3%B8konomisk_v%C3%A6kst/innovation, lokaliseret d. 25.04.19
- Denning, P. J. (2009). *Beyond Computational Thinking*.
- Department of Education. (2013). *Computing programmes of study: key stages 1 and 2 National curriculum in England*.
- Exner, M. (2012). *Folkeskolen.dk*. Hentet fra <https://www.folkeskolen.dk/519757/i-estland-laerere-at-programmere-fra-1-klasse>
- Ford, M. (2015). *The Rise of The Robots*. Oneworld Publications.
- Frank, L. (2013). *Folkeskolen.dk*. Hentet fra <https://www.folkeskolen.dk/530564/enighed-i-uddannelsesverdenen-elever-skal-laere-at-programmere-i-skolen>
- Hansen, S. (20. August 2018). *TV 2 nyheder*. Hentet fra <http://nyheder.tv2.dk/samfund/2018-08-20-it-specialister-er-en-mangelvare-studerende-rekrutteres-kort-efter-studiestart>, lokaliseret d. 22.04.2019

- IT-Branchen. (2016). Hentet fra <https://itb.dk/maerkesager/fremtidens-kompetencer/hvad-er-coding-class/>, lokaliseret d. 20.4.2019
- IT-Branchen. (2016). Hentet fra <https://itb.dk/maerkesager/fremtidens-kompetencer/coding-class/>, lokaliseret d. 20.4.2019
- Jørgensen, N. (2018). *Teknologiers indre mekanismer og processer Eksemplificeret med digital signatur.* .
- Jelsøe, E. (2018). Teknologiske systemer og artefakter 1 kursusgang 2.
- Jelsøe, E. (2018). Teknologiske systemer og artefakter Kursusgang 4.
- Kongstad, J. (2019). *Jyllandsposten*. Hentet fra <https://jyllands-posten.dk/premium/indland/ECE11108789/stor-mangel-paa-itspecialister-kan-bremse-cyberoprustning/>,
- Lifewire. (2019). Hentet fra <https://www.lifewire.com/kids-programming-languages-4125938>
- Mainz, P. (2016). Lærer har 14 minutters forberedelse til hver time. *Politikken*.
- McKinsey. (2017). *Automatiseringens effekt på det danske arbejdsmarked*. Disruptionsrådet.
- Nardelli, E. (2019). *Communications of the ACM*. Hentet fra <https://cacm.acm.org/magazines/2019/2/234348-do-we-really-need-computational-thinking/fulltext>
- Papert, S. (1980). *Mindstorms. Children, Computers and Powerful Ideas*. New York: Basic books. Basic Books, Inc., Publishers .
- Papert.org*. (u.d.). Hentet fra <http://www.papert.org>
- Pedersen, M., & Kiellberg, J. (2019). Bilag 1.
- Petropouleas, E. (2018). *Vitec*. Hentet fra <https://www.mv-nordic.com/dk/bloggen/artikel/computational-thinking/>
- Resnik, M. (2009). *Scratch: Programming for All* .
- Rogers, E. (1983). *Diffusion of innovation third edition*.
- Schødt, U. (2019). Hentet fra <https://metodeguiden.au.dk/interviews/>
- Scrath*. (u.d.). Hentet fra <https://scratch.mit.edu/>
- Shape Robotics*. (u.d.). Hentet fra <https://www.shaperobotics.com>
- Shape Robotics*. (u.d.). Hentet fra <https://www.shaperobotics.com/da/fable-systemet/>
- Slyngborg, L. (2019). Bilag 2.

Tiobe. (u.d.). Hentet fra <https://www.tiobe.com/tiobe-index//>

Wikipedia . (u.d.). *Debugging*. Hentet fra <https://en.wikipedia.org/wiki/Debugging>), lokaliseret d. 13.4.2019

Wikipedia. (u.d.). *Wikipedia*. Hentet fra <https://en.wikipedia.org/wiki/Computing>, lokaliseret d. 13.4.2019

Yoshinaka, Y., & Jørgensen, U. (2009). *Teknologi som genstand og vision*.