SKOLPORTENS NUMRERADE ARTIKELSERIE
FÖR UNDERVISNING, LÄRANDE OCH LEDARSKAP

ATT FRÄMJA DATALOGISKT TÄNKANDE I FÖRSKOLEKLASSEN

Elever kan mer än vi tror

FÖRFATTARE:
Åsa Chibás
Jalal Nouri
Eva Norén
Lechen Zhang

SKOLPORTEN
LEDA & LÄRA
3/2022

SAMMANFATTNING

I MÅNGA LÄNDER världen runt har digitalisering och programmering införts i läroplaner för förskola och grundskola. Under senare tid har flera studier om undervisning i programmering i olika sammanhang genomförts. Dessa studier har oftast haft fokus på elever från 7 år och äldre och det finns få studier som rapporterar om införandet av programmering för barn i förskola och yngre elever i förskoleklass. Denna klyfta hanterar vi i denna studie som fokuserar på programmering och utveckling av elevers datalogiska tänkande i förskoleklass. Hösten 2017 startade Ifous (Innovation, forskning och utveckling i skola och förskola) ett forsknings- och utvecklingsprogram med fokus på utveckling av didaktiska modeller för programmering i utbildningen från förskola till och med grundskolan. Programmet avslutades våren 2020. I detta program deltog skolor från fem skolhuvudmän, Tyresö kommun, Åstorps kommun, Simrishamns kommun, Stockholms stad och Freinetskolan Hugin i Norrtälje. Programmet som sådant innebar viss kompetensutveckling och främjade forskning om programmeringsundervisning från förskoleklass till och med årskurs 9 i grundskolan. I denna artikel presenterar vi hur en lärare arbetade med programmering för att främja datalogiskt tänkande hos elever i förskoleklass under treårsperioden och den erfarenhet som hon förvärvat. I artikeln drar vi slutsatsen att förskoleklasselever med systematisk och tankeväckande didaktisk modellering fullt ut kan utveckla ett antal grundläggande datalogiska färdigheter.

Åsa Chibás, lärare i förskoleklass, Freinetskolan Hugin, Norrtälje, asa@freinetskolanhugin.se
Jalal Nouri, docent, Stockholms universitet, jalal.nouri@dsv.su.se
Eva Norén, docent, Stockholms universitet, eva.noren@mnd.su.se
Lechen Zhang, doktorand, Stockholms universitet, chen@dsv.su.se

Denna artikel har den 9 augusti 2021 accepterats för publicering i Skolportens numrerade artikelserie för utvecklingsarbete i skolan. Artikeln har granskats av en forskare som ingår i Skolportens granskargrupp.

Denna artikel är publicerad i Skolportens artikelserie Leda & Lära:
www.skolporten.se/forskning/utveckling/

Aktuella Författaranvisningar & Skrivregler:
www.skolporten.se/forskning/skolutveckling/skolportens-utvecklingsartiklar/

Vill du också skriva en utvecklingsartikel? Mejla till redaktionen@skolporten.se

INNEHÅLL

INTRODUKTION

IDAG LEVER DE flesta barn och ungdomar i en datoriserad, programmerbar värld och många av dem kommer som vuxna att arbeta inom områden som involverar eller påverkas av datavetenskap. Barr och Stephenson (2011) gör gällande att datavetenskap är genomgripande för att främja lösningar av överhängande problem i det digitala samhället. Dagens unga behöver därför förstå datavetenskapens principer och praxis (Seehorn et al., 2013) och programmering kan ses som ett sätt att utveckla dessa färdigheter (Brennan & Resnick, 2012). Ur en didaktisk synvinkel kan programmering uppfattas vara en nyckelkompetens, vilket Fessakis, Gouli och Mavroudi (2013) hävdar. Som sådan kan elever behöva fördjupa sin förståelse inom flera kunskapsområden relaterade till datavetenskap, såsom datalogiskt tänkande (DT).[1] Flera forskare, bland andra Grover och Pea (2013) betonar att det inte är tillräckligt att vänta till gymnasiet eller högskolan för att bli bekant med detta ämne. Det finns således ett behov av att ändra utbildningsmodeller för lärande i programmering genom hela skolsystemet. Intressant att notera är dock att redan på 1970-talet började Papert (1980) att forska om och utveckla metoder för att introducera programmering för yngre elever (se även McNerney, 2004).

I många länder har läroplaner för förskola och grundskola uppdaterats med digitalisering- och programmeringsinnehåll också i förskola och för yngre elever. Den forskning vi refererar till är till stor del internationell. I det svenska sammanhanget går *barn* i förskolan och när de börjar i förskoleklass blir de *elever* eftersom förskoleklassen ingår i skolsystemet. Eftersom åldern för detta varierar och skolsystem i olika länder är olika, så använder vi både barn och elever i vår text. Vilket begrepp vi valt speglar vilken skolform som har studerats. Vi använder även begreppen förskolebarn respektive förskoleklasselever. Noteras bör att förskoleklass är ett svenskt fenomen. Australiska barn ska lära sig hur man löser problem med hjälp av programmering (Falkner, Vivian & Falkner, 2014). Englands nya nationella läroplan för datorer 2014 kräver att elever har upprepad praktisk erfarenhet av att konstruera datorprogram för att lösa problem (Department of Education, 2013). Programmering nämns uttryckligen i läroplanen för grundskolorna i Finland (Heintz, Mannila & Färnqvist, 2016). I den svenska, nyligen reviderade, läroplanen för grundskolan (förskoleklass – årskurs 9) ingår programmering som en del av digital kompetens. Trots denna ökande uppmärksamhet har endast ett fåtal studier undersökt programmering i förskola och förskoleklass. Exempel på studier som genomförts är Rose, Habgood och Jay (2017) som jämförde undervisning i programmering genom ScratchJr. och Lightbot med 6-åringar för att undersöka inverkan av olika programmeringsmiljöer. Fessakis, Gouli och Mavroudi (2013) genomförde en explorativ fallstudie där 5–6 åringar använde programmering för problemlösning i en LOGO-baserad miljö på en interaktiv whiteboard i förskolemiljö. Vi återkommer dock till ytterligare tidigare forskning längre fram i artikeln. Emellertid finns det ingen forskning som fokuserar på de didaktiska strategier och modeller som lärare använder för att introducera programmering i förskoleklass som idag är en del av grundskolan.

Detta är delvis bakgrunden till att Ifousår 2017 startade ett nationellt forsknings- och utvecklingsprogram, *Programmering i ämnesundervisning* med fokus på att undersöka och utveckla didaktiska arbetsmetoder för programmering i undervisningen från förskole-

1 Från engelskans 'computational thinking'. Begreppet omfattar idéer om abstraktion, datarepresentation och logiskt organiserande data. Vi återkommer till begreppet längre fram i texten.

klass till och med årskurs 9. Cirka 135 lärare och rektorer från 15 skolor involverades.

Lärarna som deltog i programmet valde själva vilka programmeringsspråk de skulle använda i sin undervisning, närmare bestämt analogt, blockbaserat eller textbaserat. Programmeringslektionerna planerades inom en rad skolämnen, såsom matematik, teknik, svenska, samhällsorienterade och naturorienterande ämnen, men också ämnesövergripande programmeringslektioner planerades.

I denna artikel redogör vi för hur en lärare under projektperioden arbetade med programmering i tre förskoleklasser för att främja elevernas datalogiska tänkande och de erfarenheter som då förvärvades.

SYFTE OCH FRÅGESTÄLLNINGAR

SYFTET MED ARTIKELN är att ta reda på hur undervisning i programmering som är ämnad att utveckla förskoleklasselevers datalogiska tänkande kan utformas och hur den gestaltades under utvecklingsprogrammets tre år, samt vad eleverna ges möjlighet att lära sig. Vi fokuserar således inte vad eleverna individuellt faktiskt lärde sig.

Vi besvarar följande frågor:

1. Hur utformas programmeringsundervisning med fokus på elevers datalogiska tänkande i förskoleklass?
2. Vilka datalogiska begrepp och metoder är möjliga för eleverna att lära sig genom undervisningen i programmering?
3. Vilka för- och nackdelar framträder i den undervisning om programmering som växer fram i relation till elevernas datalogiska tänkande?

I artikeln har Brennan och Resnicks ramverk för datalogiskt tänkande (2012) och den utökade versionen av Zhang & Nouri (2019) varit utgångspunkt för att utveckla metoder och att kunna utvärdera möjligheter för elever att utveckla färdigheter i datalogiskt tänkande. Detta ramverk består av tre dimensioner: 1) *datalogiska begrepp* som programmerare använder (dvs. sekvenser, loopar, parallellism, händelser, villkor och data), 2) *datalogiska praktiker* (dvs. att arbeta inkrementellt och iterativt, vilket innebär att testa och felsöka, återanvända och remixa kod, samt att abstrahera och modellera) och 3) *datavetenskapliga perspektiv* (dvs. uttrycka sig, arbeta kollaborativt och ifrågasätta). Som Kong (2016) uttrycker är fördelen med detta ramverk att det ”täcker in datalogiskt tänkande med omfattande bredd” (s. 379). Brennan och Resnicks ramverk har diskuterats och använts som grund i många tidigare empiriska och teoretiska studier om datalogiskt tänkande (Lye & Koh, 2014). Vad är då datalogiskt tänkande?

BEGREPPET DATA-LOGISKT TÄNKANDE

TERMEN 'COMPUTIONAL THINKING (CT)' – 'datalogiskt tänkande (DT)' användes först på 1980-talet i Seymour Paperts bok "Mindstorms: Children, Computers, and Powerful Ideas". Papert trodde att barn och yngre elevers konstruktioner genom programmering, till exempel med hans skapande programspråk Logo, kunde underlätta deras procedurella tänkande över flera discipliner. Två decennier senare återuppväckte Jeanette Wing 'datalogiskt tänkande' genom att 2006 förnya definitionen, vilket fick stort inflytande. Wing (2006, s.33) beskrev datalogiskt tänkande som ett sätt att lösa problem, designa system och förstå mänskligt beteende genom att använda sig av grundläggande begrepp i datavetenskap. Hon hävdade att "datalogiskt tänkande är en grundläggande färdighet för alla, inte bara för datavetare. Vi bör lägga till datalogiskt tänkande till läsning, skrivning och aritmetik för att utveckla varje elevs analytiska förmåga" (ibid). Wings artikel gav upphov till en ofta kontroversiell diskussion och debatt bland datavetare, kognitiva forskare och lärare angående naturen, definitionen och tillämpning av datalogiskt tänkande (Barr, Harrison, & Conery, 2011, s.20). På grund av de olika uppfattningarna om datalogiskt tänkandes omfattning och karaktär har Wings definition gått igenom många revideringar och förfiningar, vilket lett till en allmänt accepterad definition av 'datalogiskt tänkande' (CT) som hittills saknats (Lye & Koh, 2014).

Trots oenigheten om definitionen av datalogiskt tänkande finns två vanligt förekommande faktorer i den befintliga litteraturen från förskola till grund- och gymnasieutbildning. Först och främst involverar kärnan i DT i den pedagogiska miljön: abstraktion, algoritm, nedbrytning av problem i mindre delar, parallellism, testning, felsökning och kontroll (Zhang & Nouri, 2019). För det andra tror olika forskare och relevanta organisationer att DT kan förvärvas genom undervisning av olika färdigheter relaterade till DT (ibid). Till exempel har datavetarlärarföreningen och International Society for Technology in Education (CSTA & ISTE, 2011) utvecklat en operativ definition. Den säger att DT är en problemlösningsprocess som inkluderar (men inte är begränsad till) följande färdigheter: formulera problem på ett sätt som gör att vi kan använda en dator och andra verktyg för att lösa dem, logiskt organisera och analysera data, representerar data genom abstraktioner med modeller och simuleringar, automatisera lösningar genom algoritmiskt tänkande (en serie bestämda steg), identifiera, analysera och implementera möjliga lösningar med målet att uppnå den mest användbara och effektiva kombinationen av steg och resurser; generalisera och överföra denna problemlösningsprocess till en mängd andra problem av olika sort.

Som nämnts tar vi i denna artikel avstamp i Brennan och Resnicks DT-ramverk (2012). Detta ramverk ser vi som passande eftersom det kan engagera yngre elever, som i förskoleklass, genom principen "lågt golv och högt i tak" (Brennan och Resnick, 2012). Utifrån ramverket kategoriserar vi DT i tre dimensioner:

- DT – **begrepp:** begreppen som eleverna engagerar sig i när de programmerar, dvs. sekvenser, loopar, parallellism, händelser, villkor, operatörer och data.
- DT – **praxis:** den praxis som eleverna utvecklar när de engagerar sig i begreppen, dvs. testar och felsöker, återanvänder och remixar, abstraherar och modellerar.
- DT – **perspektiv:** de perspektiv som eleverna formar om världen omkring sig och om sig själva, dvs. att uttrycka sig, samarbeta och ifrågasätta.

Ramverket har hjälpt läraren att få syn på de olika förmågorna vid planering av programmeringsaktiviteter med fokus på DT. Ramverket har också använts i forskarnas analys av data.

TIDIGARE STUDIER OM PROGRAMMERING I FÖRSKOLAN

PROGRAMMERING OCH FORSKNING om programmering i tidig ålder kan dateras tillbaka till LOGOs tid. Programmeringsspråket LOGO utvecklades av Papert på 1980-talet för att barn skulle få lära sig ett programmeringsspråk genom att styra en sköldpadda på en datorskärm. Papert (1980) var inspirerad av Piaget och var övertygad om att yngre barns tänkande utvecklas genom kommunikation och att kommunikationen via datorn skulle öppna barns ögon för sitt eget tänkande. I Sverige genomförde Hedrén (1990) en studie om mellanstadieelever som programmerade i LOGO. Eleverna som var med i studien blev duktiga problemlösare.

På 1980-talet studerade flera forskare programmeringens effekt på mindre barns kognition, metakognitionsförmåga och prestationer, såsom förmåga att kontrollera sin förståelse (t.ex. Clements & Gullo, 1984; Miller & Emihovich; 1986). Dessa tidiga försök visade att yngre elevers förmåga att upptäcka inbäddade fel i en kommunikationsuppgift var signifikant större efter träning med LOGO än kontrollgruppens med datorassisterade instruktioner. Även om Wing återupplivade datalogiskt tänkande i undervisning för två decennier sedan, är det under de senaste åren som lärare och forskare började introducera datalogiskt tänkande genom programmering för barn och yngre elever. Det finns därför få studier som fokuserat på barns datalogiska tänkande i förskolan (Saez-Lopez, Roman-Gonzaez, & Vazquez-Cano, 2016). Det finns åtminstone två skäl för detta: För det första har det saknats åldersanpassade programmeringsverktyg för de allra yngsta. Traditionellt har man börjar lära sig programmering med textbaserade programmeringsspråk som inte är lämpliga för 6-åringar. Dock har de senaste ansträngningarna inom utbildningsteknologin, såsom ScratchJr, KIBO och Beebots etc., gjort programmering och utveckling av datalogiskt tänkande mer tillgänglig och mindre abstrakt för unga elever (Ching, Hsu, & Baldwin, 2018). För det andra saknas lärare utrustade med adekvata pedagogiska och datalogiska färdighet i förskolor. Förskollärare och lärare i de tidiga skolåren är vanligtvis generalister och har inte någon utbildning i datavetenskap. Utan tvekan kan lärarnas kompetens i ett ämne direkt påverka elevers resultat och deras attityd till ett ämne (Zhang, 2020). Eftersom forskning om utveckling av barns och yngre elevers datalogiska tänkande fortfarande är i sin linda (Bers, Flannery, Kazakoff, & Sullivan, 2014), syftar denna studie till att bidra med kunskap om elevers möjligheter att utveckla datalogiskt tänkande i förskoleklass.

Senare litteratur om programmering i förskolan kan kategoriseras utifrån de programmeringsverktyg som används: pedagogisk robotprogrammering (t.ex. Beebot), programmering med konkreta block (t.ex. KIBO), visuell blockbaserad programmering (t.ex. ScratchJr) och analog programmering. Forskarsamhället har tagit emot pedagogisk robotik med äkta entusiasm som ett tillvägagångssätt för att undervisa i datalogiskt tänkande för förskolebarn (Angeli & Valanides, 2020). En stor fördel med pedagogisk robotik är att barn kan interagera med en robot och observera de omedelbara effekterna av programmeringen när det gäller robotens handlande. Angeli & Valanides (ibid) drog till exempel slutsatsen att Beebot hjälper barn att lära sig bryta ner problem i mindre delar.

Caballero-González, Muñoz och Muñoz-Repiso (2019) lät barn lösa problem genom programmeringsutmaningar med BeeBot. Resultatet visade att de 131 3–6-åringar som deltog fick goda resultat i sekvensering (algoritmer), korrespondens mellan handlingar och felsökning. I Flannery och Bers (2013) studie använde 4–6-åringar 'Creative Hybrid Environment for Robotics Programming' (CHERP) för att lära en robot att dansa hokey pokey. Kazakoff, Sullivan och Bers (2012) engagerade också CHERP för att främja datalogiskt tänkande hos barn och yngre elever.

Samtidigt finns det forskning om virtuell pedagogisk robotik som Lightbot, en virtuell robot som kan följa en serie instruktioner för att utföra uppgifter inom ett bestämt utrymme. Rose, Habgood och Jay (2017) undersökte Lightbots kapacitet att utveckla datalogiskt tänkande, såsom abstraktion och algoritm, hos en grupp 6–7-åringar. Programmering av materiella block, som KIBO, gör det möjligt för barn och yngre elever att uppfinna med hjälp av block som innehåller motorer och sensorer. Dessa block kan programmeras genom att koppla in sekvenser, kontrollstrukturer och modellering. Studier (t.ex. Sullivan, Bers, & Mihm, 2017; Elkin, Sullivan och Bers, 2016) föreslog att även barn i förskolan kunde behärska grundläggande enkla programmering genom konkreta block. Även visuell blockbaserad programmering är ett populärt val i förskolan för barn i förskolan och elever i förskoleklass. Verktyg som ScratchJr exponerar förskolebarn för datavenskapliga begrepp, praxis och perspektiv genom att barnen kan skapa animationer, collage och berättelser (Leidl, Bers, & Mihm, 2017; Papadakis, Kalogiannakis, & Zaranis, 2016).

I Sverige syns sedan några år tillbaka en ökad tonvikt på att integrera programmering i kombination med digitala lärplattor även om programmering inte nämns formellt i den reviderade läroplanen för förskolan. Som exempel visade Otterborn, Schönborn och Hultén (2019) i en enkätstudie att förskollärare satte datalogiskt tänkande som mål för programmeringsaktiviteter i relation till "twenty-first-century skills". Programmeringen var ofta aktivt kopplad till lärande inom andra områden såsom naturvetenskap, matematik, teknik och språk. Tillvägagångssättet visar att traditionella svenska förskoledidaktiska modeller omsattes i undervisning om programmering.

METOD

ARTIKEL ÄR BASERAD på arbetet i tre förskoleklasser på Freinetskolan Hugin, under läsåren 2017/18, 2018/19 och 2019/20. De tre grupperna har varierat i storlek från 20–24 elever. Oftast var det två vuxna i klassrummet under alla programmeringsaktiviteter, antingen två lärare eller en lärare och en assistent. Som hårdvara har eleverna använt iPads 2:1 och de har haft tillgång till en Bluebot-robot. Klassrummen var utrustade med en stor skärm (whiteboard) där lärares och elevers iPads har projicerats. Programvaran bestod av applikationer och utbildningswebbplatser som är gratis. Analoga programmeringsövningar har krävt penna, kritor och papper i olika storlekar och färger, samt en del annat material.

Lärarna har planerat lektionerna. Fokus för lektionsplaneringen var: 1) lektionsaktivitet, 2) förväntade elevreaktioner och förväntade lärarrespons och stöd till eleverna, 3) mål och syfte för lektionen, 4) metoder för utvärdering och till sist 5) utvärdering. Utvärderingen har relaterats till mål och syfte med lektionerna men inte särskilt till om och vad varje elev individuellt har lärt sig, eftersom lärarna avsåg att utveckla undervisningen i programmering. Lärarna bedömde dock löpande elevernas utveckling och lärande i de interaktioner som pågick och den dokumentation som samlas in under lektionsaktiviteterna. Utvärderingarna används i första hand som utgångspunkt för fortsatt utveckling av undervisningen under de tre åren. Eftersom ett av syftena med Ifous-programmet också är *”att låta lärares kunskaper och erfarenheter utgöra grund för gemensam kunskapsutveckling, genom att lärare och forskare samarbetar”* har denna artikel författats tillsammans, om än lärares arbete utgör basen.

DATAINSAMLING OCH ANALYS

DATA HAR SAMLATS in i olika former av lärare och forskare. Klassrumsobservationer som genomförts, lektionsplaneringar och lektioner har dokumenterats via video, fotografier och anteckningar. Videorna och fotografierna inkluderar introduktioner av lektioner och diskussioner i klassrummet, elever som arbetar i olika konstellationer och pågående elevprojekt samt elevernas slutliga produktioner. Elevernas självbedömningar och reflektioner har antecknats, fotograferats eller videoinspelats. Läraren har fört anteckningar över innehållet och strukturen i lektionerna. Elevernas uppfattningar om lektionerna har använts för att förbättra lektionerna och har ibland delats med forskare för att diskutera datalogiskt tänkande eller upprätta nya lektionsmål och tillvägagångssätt.

Läraren (första författaren) och en av forskarna (andra författaren) strukturerade det insamlade materialet och utförde individuella analyser. Även gemensamma analyser och diskussioner fördes regelbundet. Ramverket för Brennan och Resnick (2012) användes som en lins för att ge en struktur för analys och presentation av arbetet med DT-begrepp, -praxis och -perspektiv i förskoleklasserna. Det innebär att den begreppsförståelse och den praxis som eleverna ges möjligheter att utveckla och engagera sig i när de arbetar med programmering, samt de perspektiv som de formar om världen omkring sig och om sig själva är i fokus, vilket vi visar i följande avsnitt.

RESULTAT – LÄRANDEPRAKTIKER

I DET FÖLJANDE presenterar vi hur programmeringsundervisning utformades med fokus på elevers datalogiska tänkande i förskoleklass. Vi redogör även för hur progression togs med i planeringen när metoder utvecklades. Presentationen av lärandeaktiviteter som genomfördes följer en kronologisk ordning och inriktar sig på datalogiskt tänkande såsom färdigheter i sekvensering, testning, felsökning, återanvändning, händelser, villkor, loopar samt att arbeta tillsammans och att uttrycka sig. I presentationen sammanfaller de tre förskoleklassernas arbete med programmering. Vi har valt att använda så kallade täta beskrivningar för att levandegöra så mycket som möjligt av aktiviteterna från de tre år Ifous-programmet pågick. Tät beskrivning är en etnografisk term (Hammersley & Atkinson, 1993) som innebär att vi har producerat täta, detaljrika beskrivningar av det som skett i förskoleklassen. Avsnittet avslutas med en presentation av vilka förmågor, relaterade till DT, som är möjliga för eleverna att utveckla.

SEKVENSERING, TESTNING OCH FELSÖKNING: ATT ARBETA MED EN BLUEBOT-ROBOT

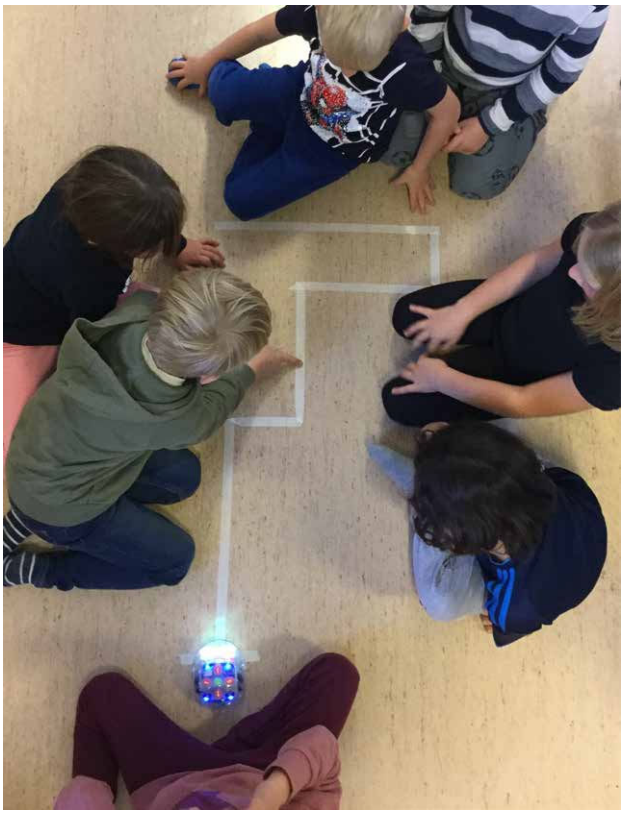
FÖR ATT GE eleverna möjligheter att utveckla de datalogiska förmågorna sekvensering, testning och felsökning introducerades Bluebot-robot i ett första steg till programmering. Med utgångspunkt i en didaktisk strategi, att låta eleverna undersöka och upptäcka genomfördes aktiviteter där fick läraren möjlighet bedöma att vilka elevernas förkunskaper var och vilken förförståelse de hade för enkel sekvensering. Detta skedde i samtal och genom observationer av elevernas handlingar när de arbetade i grupper om 5–6 elever.

Till att börja med undersökte eleverna roboten och delade med sig till varandra vad de visste om robotens olika funktioner eller gissade om de inte hade någon tidigare erfarenhet. Eftersom Bluebotar är gjord av genomskinlig plast, fick eleverna möjlighet att prata om robotens olika komponenter, även om vad som finns under skalet.

Ett tejpat spår på golvet fungerade som elevernas första individuella och gemensamma utforskning av sekvensering och felsökning. En elev programmerade Blueboten med inmatningsinstruktioner från hela gruppen. De programmerade på så sätt intuitivt en hel sekvens på en gång och såg Blueboten förflytta sig igenom banan. När roboten inte uppförde sig som de ville och tog fel väg behövde eleverna felsöka koden. Det visade sig vara ett naturligt sätt att introducera begreppen sekvens och felsökning på. Samtidigt involverade aktiviteten indirekt också arbete med och främjande av olika färdigheter såsom korttidsminne, koncentration, visualisering, fantasi, riktningskänsla, uppskattning av avstånd samt kommunikations- och samarbetsförmåga.

Uppgiften presenterades som en gruppaktivitet men varje elev hade möjlighet att programmera roboten. Aktiviteten inkluderar aspekter av lärande tillsammans där läraren observerade eleverna och uppmärksammade att de engagerade sig i olika former av problemlösning med stöd i hypoteser, test-

Figur 1: Eleverna diskuterar och planerar hur de ska programmera robotens rörelser.



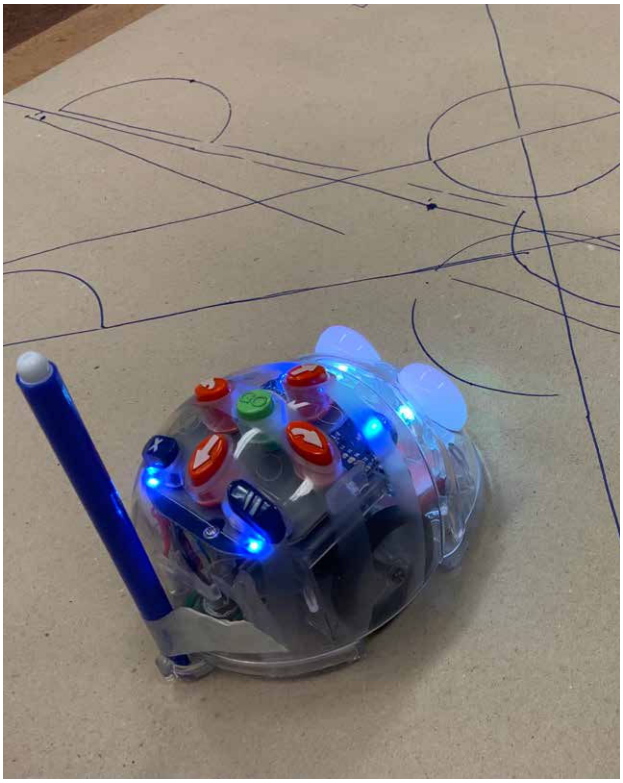
Figur 2: En elev kodar roboten att navigera eleverna roboten att rita.

ning, jämförelser. En nyckelfaktor för att alla elever ska engagera sig i programmeringsprocessen är att låta huvudprogrammeraren ha mandat att ange den information hen väljer, även om elever i gruppen ger andra förslag. Baserat på observationerna pågick ett grupparbete med diskussioner, där eleverna gav varandra förslag på hur många steg framåt roboten skulle gå, i vilken riktning den skulle vändas och var på banan den för närvarande befann sig. Det gav läraren tillfälle att observera elevernas intresse för och förmåga att tänka sekventiellt, att uppskatta avstånd och dra logiska slutsatser om de steg som krävdes av Blueboten för att nå den tejpade vägens ände.

Baserat på observationerna hade eleverna med tidigare erfarenhet av just Bluebots, eller liknande robotar, ett försprång eftersom de kände till knapparnas funktioner och hur roboten kunde röra sig. I slutet av programmeringsaktiviteten fanns det däremot inga skillnader mellan eleverna med eller utan tidigare erfarenhet. Istället var det förståelse för riktning och avstånd, arbetsminne samt kommunikations de viktigaste förmågorna som påverkade slutresultatet.

I efterföljande aktiviteter fick eleverna instruktioner att utforska på egen hand, i par eller i mindre grupper.

Figur 3: Med en fasttejpad penna på roboten kodar genom den för hand ritade banan.



Figur 4: Ett gemensamt projekt där eleverna bygger en bana av kartong och andra material de kan hitta i klassrummet.

Eleverna byggde bland annat banor med olika material och ritade banor eller kartor på stora papper. Sedan kodade de roboten för att ta sig igenom banorna. Eleverna utforskade också att fästa en penna på baksidan av roboten och låta roboten rita på stora pappersark. Detta var en öppen uppgift som möjliggjorde den fria utforskningen i programmering. Dessutom användes aktivitetskort som gav eleverna specifika uppgifter när de behövde inspiration, hjälp med fokus, vara redo för en ny utmaning eller gå vidare till en annan nivå.

Ett arbetssätt där 'trial and error', att pröva och misslyckas och att därefter pröva igen, uppmuntras var infogat i alla dessa aktiviteter, och baserat på våra observationer verkade det vara fördelaktigt för eleverna på många sätt. Först och främst verkade eleverna uppleva trial and error som lekfullt och när roboten inte tog sig fram så som planerat uppfattades det som en naturlig del av processen snarare än som ett misslyckande. Det hindrade inte eleverna från att fortsätta, snarare tvärtom. För det andra, det faktum att eleverna arbetade i grupper och turades om att programmera roboten, gjorde det möjligt för dem att observera och lära av varandra innan de försökte lösa problemen på egen hand.

Sammanfattningsvis visar den didaktiska strategin att låta eleverna undersöka och pröva att programmera en Bluebot tillsammans, att de erbjöds möjlighet att utveckla datalogiskt tänkande vad gäller sekvensering, testning och felsökning, dvs att utveckla såväl begrepp som praxis, men också att uttrycka sig, samarbeta och ifrågasätta. Detta arbetssätt användes i alla tre förskoleklasserna. Detta kan vi relatera till Brennan och Resnicks DT-ramverk (2012) i tre dimensioner. Genom principen "lågt golv och högt i tak" engageras de yngre eleverna i förskoleklass i begreppen, de utvecklar en praxis och de uttrycker sig, samarbetar och ifrågasätter.

SEKVENSERING, TESTNING OCH FELSÖKNING: ATT ARBETA MED BLUEBOT APP(LIKATIONEN)

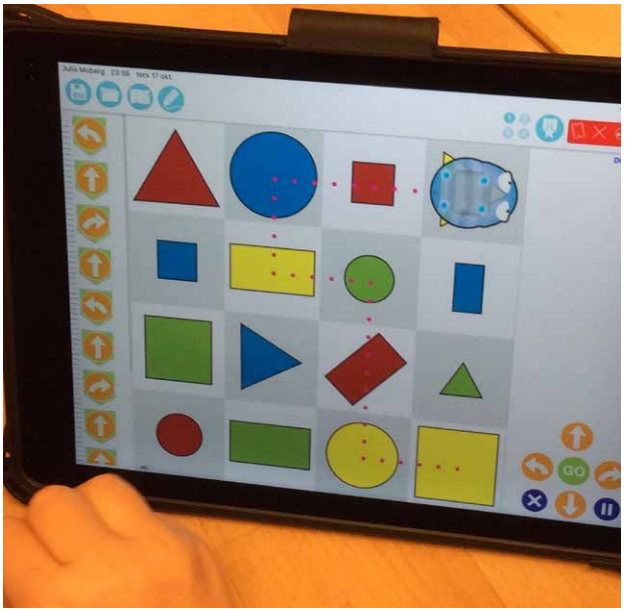
EFTER ETT ANTAL planerade aktiviteter med den fysiska Bluebot-roboten introducerade läraren lärandeaktiviteter med hjälp av Bluebot-appen. De visuella likheterna mellan Bluebot-roboten och Bluebot-appen ger en bra övergång från det ena till det andra. I applikationen finns en 2D-version av Bluebot-roboten och samma symboler som indikerar kommandona (representerade som knappar på den fysiska roboten). Appen och de lärandeaktiviteter som är kopplade till appen introducerades av läraren på den stora skärmen i klassrummet innan eleverna började utforska appen i par till att börja med och sedan individuellt.

Introduktionen av nya arbetssätt, appar, DT-begrepp, -praxis eller -perspektiv började alltid med ett gemensamt utforskande, som en gruppaktivitet, där läraren ställde frågor som uppmuntrade eleverna att undersöka och problemlösa tillsammans på den stora skärmen. Detta är en situation där en öppen miljö och ett undersökande arbetssätt gör det möjligt för alla att prata fritt, det har stor betydelse att kunna prata utan att vara säker på om man har rätt svar eller inte. När introduktionen är slutförd fick eleverna en specifik uppgift som uppmuntrade till fortsatt gemensamt utforskande i par. När de flesta hade uppnått en grundnivå av kunskap fick de uppgifter som de arbetade med individuellt.

Exempel på uppdrag är mer specifika problemlösningsinstruktioner: gå till den röda triangeln med så få steg som möjligt; kan du få roboten att rita en kvadrat, kan du få roboten att gå på varje kvadrat på nätet bara en gång, eller öppna instruktioner som ”utforska en ritning på vit bakgrund”. Analoga aktiviteter presenterades också, som den som visas i figur 7.

Att introducera Bluebot-appen visade sig vara ett naturligt steg föra att introducera programmering som en digital aktivitet, dvs. gå från att programmera en fysisk robot genom att trycka på fysiska knappar, till att välja visuella kommandon i ett gränssnitt för att skapa en sekvens av kommandon. Den begränsade mängden block var fördelaktig i den inledande fasen av programmeringen, eftersom eleverna gavs möjlighet att utveckla grundläggande DT-färdigheter utan att bli överväldigade av för många kommandon och funktioner. Bluebot-appen erbjuder inbyggda utma-

Figur 5: Ett resultat från en utmaning: Starta på den gula kvadraten, gå på varje gul figur och stanna på den gula triangeln. Vilket är det minsta antal steg du behöver koda för att lösa uppgiften?



Figur 6: En skattkarta som från början konstruerades för att använda med den fysiska Blueboten fotograferades och lades därefter till som bakgrund i appen.

Figur 7: En analog aktivitet där två elever bygger en bana och en annan grupp elever letar sig igenom den samtidigt som de lägger ut pilar för att representera den väg de har gått. ”Banbyggarna” kontrollerar sedan att pilarna stämmer. Elevernas samtal ger läraren möjlighet att bedöma elevernas förståelse och se eventuella luckor i förståelsen. Denna aktivitet introducerades efter att arbetet med Bluebot-appen hade börjat.

ningar som eleverna kan försöka lösa i sin egen takt samtidigt som lärare har möjlighet att strukturera lärandeaktiviteter som tar hänsyn till olika tematiska arbeten. Detta kan göras genom att använda olika bakgrunder som ingår i appen eller lägga till egna digitala bilder som bakgrunder.

Observationer och bedömningar som gjordes indikerar att eleverna förstod sekvensering och de grundläggande principerna för felsökning väl efter att ha programmerat den fysiska roboten och i BlueBot-appen. Några svårigheter identifierades också, till exempel att en del elever kunde distraheras av lekfullheten och var mer motiverade att ”köra runt roboten” på ett friare och mer undersökande sätt snarare än att fokusera på specifika uppgifter och problemlösande uppgifter. Det visade sig stödjande att ge dem extra tid att utforska appen fritt och sedan gå vidare till specifika uppgifter där läraren kunde fortsätta att stödja dem efter behov. Vi har också observerat att några elever ännu inte har utvecklat den kommunikations- och samarbetsförmåga som behövs för att lösa uppgifter tillsammans i en grupp. Sådana företeelser ingår i en förskoleklasslärares dagliga arbete, dvs att planera lärandeaktiviteter som passar alla elever, vare sig de har väl utvecklad kommunikations- och samarbetsförmåga eller inte. Även här kan vi relatera till Brennan och Resnicks DT-ramverk (2012) i de tre dimensionerna. Det är lätt att trycka på knapparna på roboten fysiskt. Med grunden i programmering av roboten går det lättare att använda appen där det finns flera funktioner. I appen utvecklar eleverna förmågan att sekvensera och felsöka på en högre nivå. Eleverna utvecklas olika och uppgifterna kan individanpassas.

SEKVENSERING, FELSÖKNING, ÅTERANVÄNDNING, ATT UTTRYCKA SIG OCH SAMARBETA I CODE.ORG

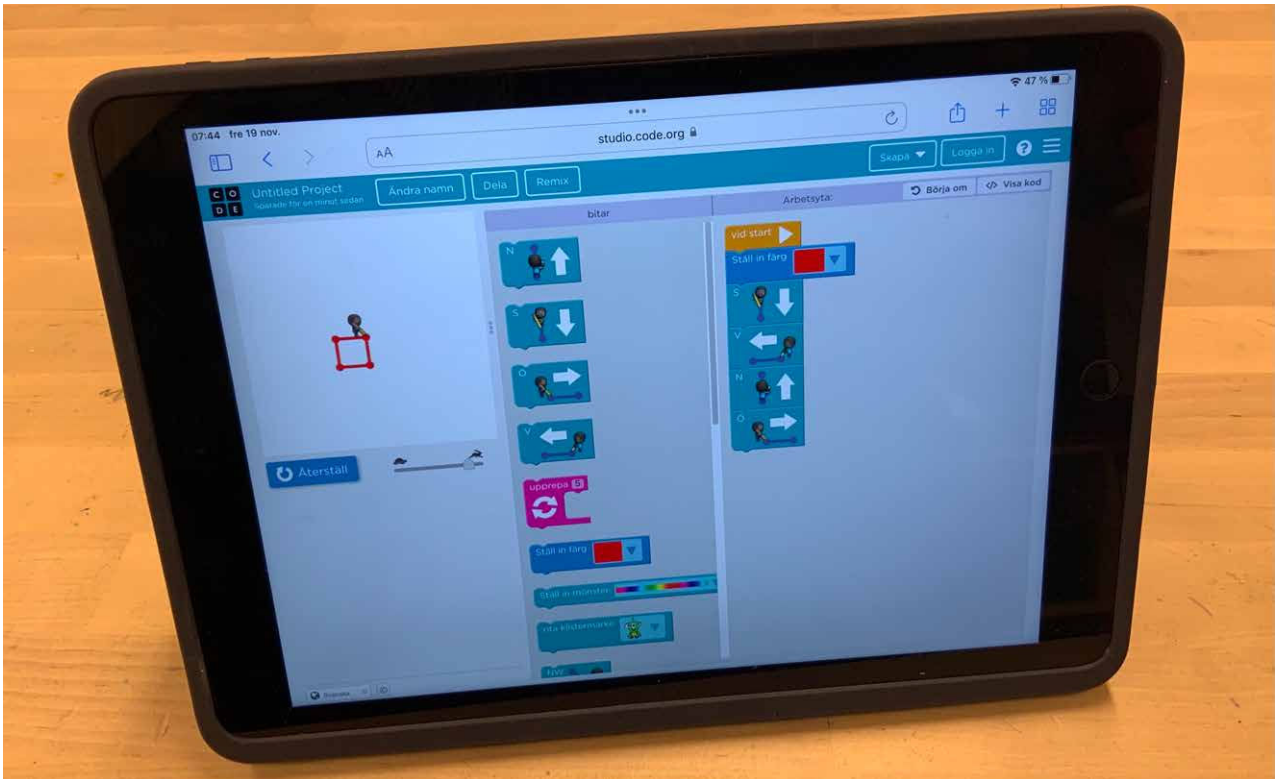
EFTER EN PERIOD av arbete med Bluebot-roboten och appen, och när lektionerna hade utvärderats, fick ett nytt arbetsområde utrymme. Det här arbetet genomfördes i två av grupperna eftersom läraren bedömde att eleverna i dessa två grupper skulle kunna fortsätta att utveckla sitt DT med en arbetsform som var mer styrd, även om ett undersökande arbetssätt fortsatte att prioriteras. Läraren introducerade code.org, som är en miljö online för lärande av programmering.

Webbplatsen code.org erbjuder projekt och kurser för elever på olika nivåer och åldrar. Den är både didaktiskt öppen, i den meningen att lärare själva skapar en didaktisk design för lärandeaktiviteter, men också har en inbyggd didaktisk struktur som vägleder eleverna steg för steg genom fördefinierade projekt, uppgifter och handledning. Det finns även existerande projekt som skapats där det går att ändra på den existerande koden. Förskoleklasseleverna fick nu använda sådana tidigare skapade projekt som utgångspunkt i lärandeaktiviteterna.

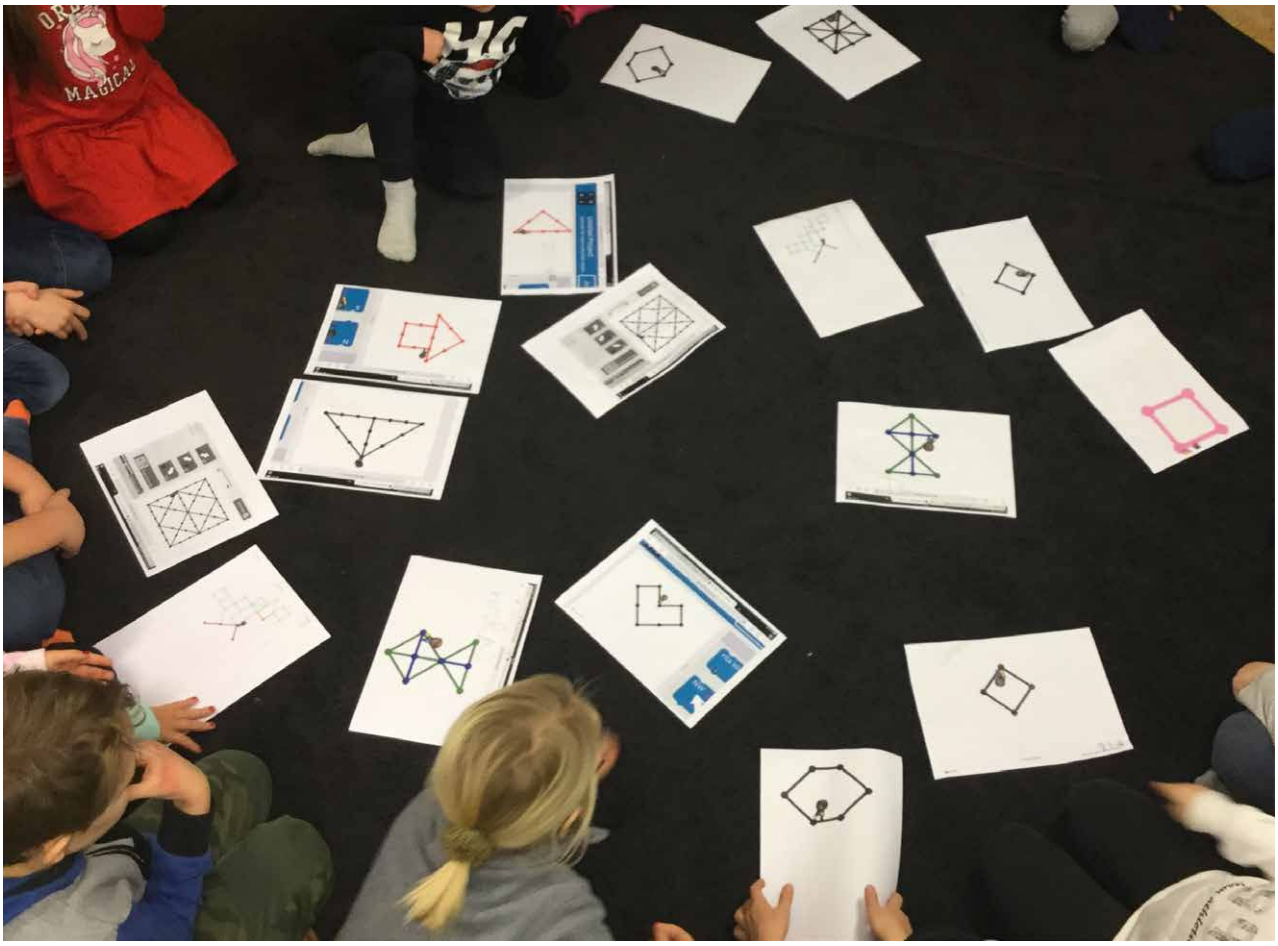
Aktiviteterna baserades på programmering av geometriska mönster och former. Aktiviteten inkluderades för att få in ett annat perspektiv och låta eleverna arbeta på ett annat sätt med mönster, än att utforska på egen hand, genom att variera och alternera färg, form och storlek med till exempel pärlor, rutat papper och liknande metoder som används i andra sammanhang i förskoleklassen. En annan aspekt var att knyta an till design och DT-perspektivet genom att skapa ett digitalt tvådimensionellt mönster som låg till grund för skapandet i 3D med fysiskt material.

Introduktionen gjordes genom att vi tillsammans tittade på ett förprogrammerat mönster ritat på webbplatsen. Hela klassen såg detta på den stora skärmen och efteråt diskuterade de hur de trodde att det var uppbyggt. Diskussionen som följde var en kombination av vilda gissningar och kloka observationer. Det blev en gemensam undersökning och gemensamt problemlösande. Efter diskussionen studerade eleverna det förprogrammerade mönstret som ännu en

Figur 8: Koden och resultatet av att konstruera en blå kvadrat med block som är avsedda för elever som ännu inte lärt sig läsa.



Figur 9: Eleverna visar varandra sina mönster.



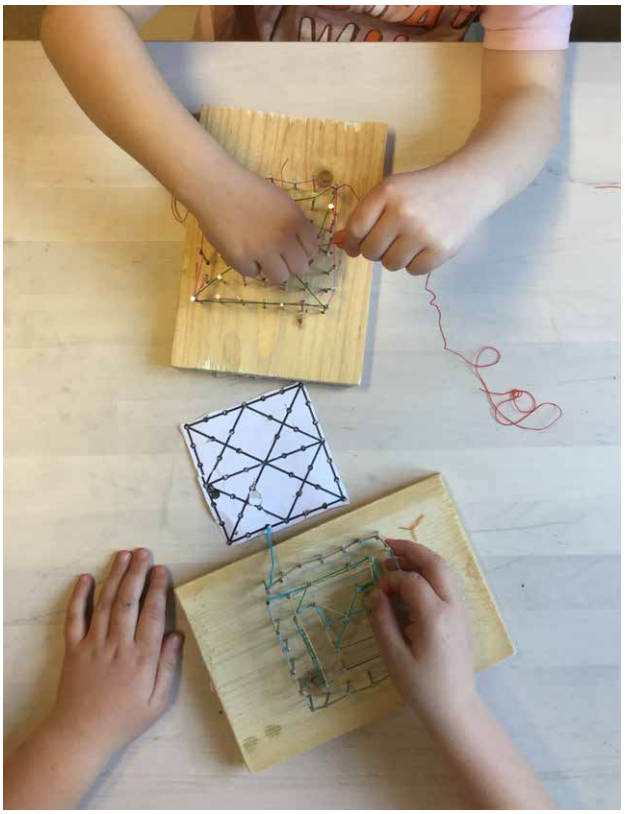
gång ritades, nu i slow motion. Diskussionen fortsatte och eleverna gjorde fler kopplingar och drog slutsatser om hur det skulle kunna fungera. Lärare och elever undersökte också blocken tillsammans.

I detta specifika projekt innehöll blocken information som riktning, avstånd, färg och mönster. Blocken är baserade på symboler snarare än ord, vilket visade sig vara fördelaktigt för denna åldersgrupp eftersom det är få som kan läsa. Det tog dock lite tid för några av eleverna att vänja sig vid symbolerna eftersom de såg annorlunda ut än Bluebot-symbolerna. Övergången från en programmeringsmiljö till en annan, och från en syntax till en annan, visade sig vara svårt för en del av eleverna.

På grund av de nya symbolerna genomfördes därför några av de första stegen tillsammans, med läraren på den stora skärmen och eleverna som arbetade parvis på iPads. Eleverna startade ett nytt projekt och visade sedan hur blocken skulle sättas ihop. Sedan instruerades eleverna att skapa en liten kvadrat och sedan en större kvadrat, för att se att de hade uppfattat grunderna (se bilden ovan). Efter utvärdering av denna aktivitet och planering av fortsatt arbete, fick eleverna i uppdrag att skapa en unik form eller mönster, individuellt eller parvis. I planeringen ingick att kombinera digitalt och analogt arbete, mest för att fånga upp elever och inte gå för fort fram, men också för att inkludera mer traditionella arbetsformer i förskoleklass.

Mönstren trycktes sedan på papper och blev utgångspunkten för att skapa en 3D-version av detsamma. Pappret tejpades på en platt träbit och lämnades där när eleverna spikade i spikar för varje prick i sitt mönster, i träbiten. När de hade skapat en kontur av mönstret med spikar tog de tråd i olika färger och fäste mellan spikarna för att göra mönstret komplett. De flesta av eleverna uppskattade denna kombination av digitalt och analogt arbete, eftersom många av dem älskar att använda hammare och spik. En del elever tog bort papperet innan de lade till tråden och andra lämnade det kvar som en del av det slutliga mönstret.

Figur 10: Efter att ha spikat lade eleverna till olikfärgade trådar för att få mönstren att framträda.



Figur 11: De slutliga skapelserna visades upp på en plats i skolan där andra elever kunde se dem.

De flesta eleverna i förskoleklasserna kunde koda mönster på egen hand och endast ett fåtal behövde vägledning av en vuxen för att slutföra projektet. Många av eleverna valde att arbeta i par och även om de programmerade och använde samma form eller mönster blev deras slutliga 3D-former väldigt olika. De elever som ville skapa fler mönster fick möjlighet att göra det i efterföljande aktiviteter. Sammantaget visade sig aktiviteten vara ett sätt att använda programmering och analoga medel för att låta eleverna uttrycka sina kreativa och estetiska intressen. De flesta av eleverna var mycket engagerade och alla kände sig stolta över sina skapelser. Även detta moment kan knytas till Brennan och Resnicks DT-ramverk (2012). Även här är det tydligast att de utvecklar en praxis och att de samarbetar, ifrågasätter, och engagerar sig i återanvändning av kod.

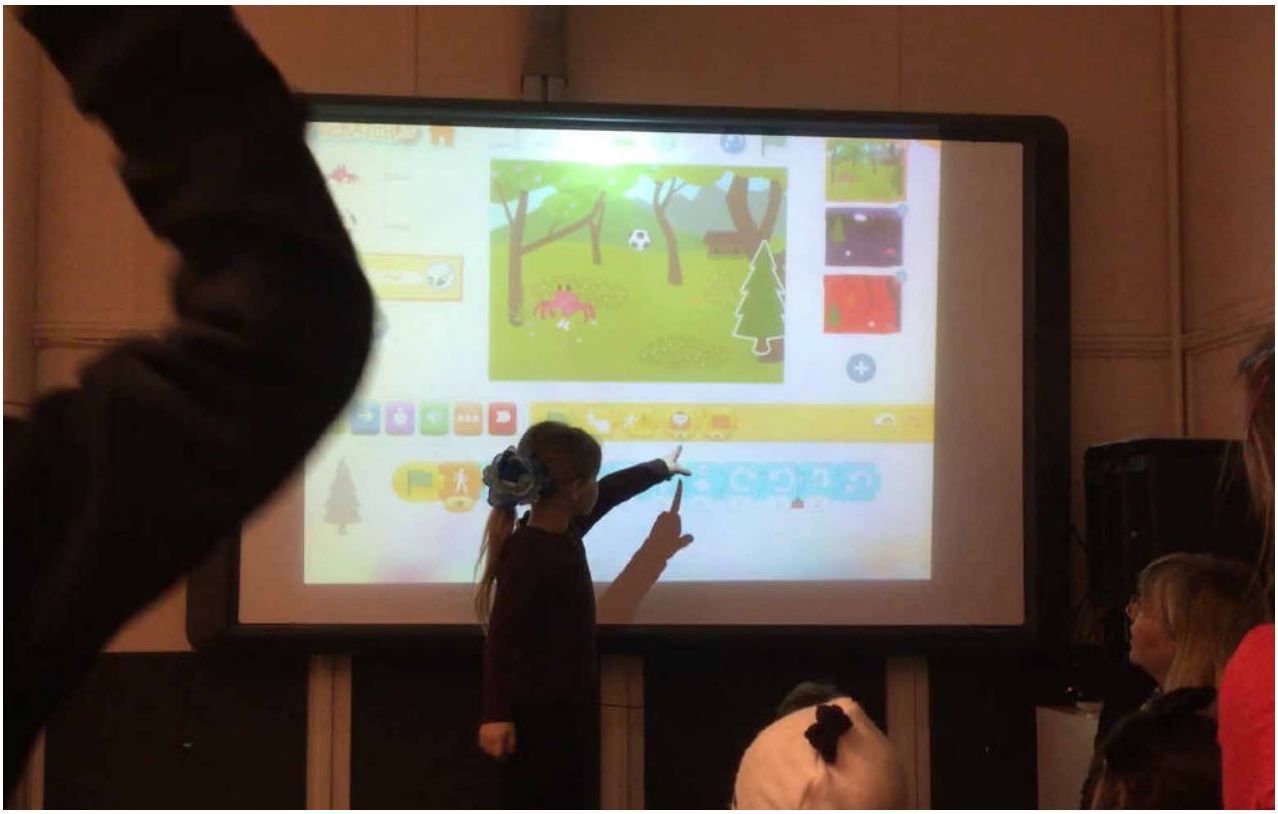
SEKVENSERING, FELSÖKNING, HÄNDELSER, VILLKOR, LOOPAR, ATT UTTRYCKA SIG OCH ATT SAMARBETA I SCRATCHJR

EFTERSOM ELEVERNA HADE olika erfarenheter av programmering även utanför skolan planerades aktiviteter som drog nytta av det.

Inför en rad lärandeaktiviteter som gick ut på att utforska ScratchJr fick några elever som hade tidigare erfarenhet av ScratchJr leda en introduktion för andra elever. Även om många lärare börjar med denna miljö med lite äldre elever, visade det sig vara en användbar öppen programmeringsmiljö även för förskoleklasselever. Med ScratchJr fick de möjlighet att växa i sin egen takt och det gjorde att de kunde öva på flera DT-begrepp och -metoder, såsom sekvensering, testning och felsökning, händelser, villkor och loopar. I senare utvärdering och analys av observationer visade sig förskoleklasselever kunna skapa ganska komplexa sekvenser.

Under elevernas introduktion visades en av elevernas lärplatteskärm på den stora skärmen. Eleverna visade sina klasskamrater hur man ritar sina egna sprajts, dvs karaktärer, i det här fallet emojis. Eleverna prövade därefter parvis och sedan individuellt innan nästa del av introduktionen genomfördes. Eleverna introducerade då rörelser och visade hur man flyttar emojin/sprajten i olika riktningar. Rörelseblocken i

Figur 12: En elev förklarar ett blocks funktion för resten av klassen.



ScratchJr är de som mest liknar de tidigare erfarenheterna av programmering och skapar en övergång till att ta sig an fler block i ScratchJr miljön.

Under den elevledda introduktionen stöttade läraren de elever som lyssnade genom att ställa frågor till dem och ge förtydliganden. När eleverna var färdiga med introduktionen gav läraren instruktionen att alla skulle starta en sekvens med samma block, nämligen startblocket (händelse). Sedan fick de utforska blocken relaterade till rörelse. I slutet av denna utforskande fas verkade alla elever känna sig hemma i den nya programmeringsmiljön eftersom de visade att de kunde rita och koda i den. Jämfört med code.org verkade eleverna uppfatta ScratchJr som mer tydligt, och steget till den nya programmeringsmiljön föreföll vara lättare.

BERÄTTELSER I SCRATCHJR

I DEN FORTSATTA planeringen av lärandeaktiviteter fick eleverna instruktioner att skapa sin egen berättelse i ScratchJr, för att inspirera och utmana eleverna att utforska och lära sig fler aspekter av programmering. Idén bakom detta var att låta eleverna uttrycka sig genom programmering, engagera sig i problemlösning och arbeta vidare med DT-metoder (Brennan & Resnick, 2012) såsom testning och felsökning samt introducera ny DT-begrepp, händelser, villkor och loopar. Läraren introducerade nya moment vid olika tidpunkter för att hjälpa eleverna att förstå och utforska nya aspekter av ScratchJr stegvis. Berättelsen fungerade som stöttning och hjälpte eleverna att relatera till ny information.

Läraren ledde sedan en gemensam introduktion i syfte att få igång elevernas fantasi. Eleverna fick instruktioner att välja två eller tre befintliga karaktärer (sprajts) och bakgrunder i ScratchJr. Dessa skulle

Figur 13: Eleverna letar efter lämpliga bakgrunder och sprajter att använda i sina berättelser och ritar dem i deras storyboards.

ligga till grund för deras berättelse. Eftersom eleverna var runt sex år varierade deras förmåga att skapa och strukturera en berättelse ganska mycket. Därför hölls en gemensam brainstorming-diskussion i början av aktiviteten där läraren diskuterade med eleverna om hur berättelser kan skapas och vilket innehåll de kan ha. Därefter introducerade läraren tekniken att använda en storyboard genom att tillhandahålla en mall på papper med tre rutor som representerar en scen vardera i ScratchJr. Eleverna fick då uppgiften att skapa sin berättelse genom att rita de tre scenerna. Elever som ännu inte kunde skriva fick möjlighet att muntligt berätta för läraren som skrev ner berättelsen under bilderna. Storyboard blev ett mycket användbart verktyg eftersom det gav en visuell riktlinje och referens samt en struktur som hjälpte eleverna att fokusera och hålla sig till den röda tråden i berättelsen. Storyboard fungerade också som ett verktyg som stödde kommunikationen mellan eleverna och läraren om berättelsen samt programmeringen.

När eleverna var färdiga med att rita och skriva sina berättelser kunde de börja att koda berättelsen i ScratchJr. Innan den individuella kodningen ägde rum samlade läraren eleverna framför den stora skärmen i klassrummet och gav en gemensam undersökande genomgång som visade hur man skapar en ram för berättelsen genom att först lägga till tre scener och deras sprajter. Eleverna ombads sedan att göra detsamma på sina lärplattor. Eftersom de redan hade utforskat några funktioner i den föregående aktiviteten började vissa koda direkt. Med tanke på att eleverna redan var bekväma med de grundläggande rörelseblocken, ombads de att utforska alla rörelse- och riktningsblock, dvs. flytta åt vänster och höger, samt upp, ner, hoppa och så vidare för att se vilka som fungerade bäst för deras berättelse. Därefter utforskades kontrollblocken, som att förstora, krympa och göra osynligt. Eleverna verkade tycka att både rörelse- och kontrollblocken var lätta att förstå, eftersom de kunde skapa en visuell koppling till deras funktion. De fick fria händer att använda alla block de behövde för att skapa sina berättelser. Läraren fanns till hands för att stötta eleverna när de kodar. Dialogen med elever individuellt skapar tillfällen för läraren att bedöma elevernas förståelse, upptäcka eventuella kunskapsluckor och för att bestämma och planera nästa steg.

När elevernas berättelser utvecklades och deras förståelse och bekvämlighet växte introducerades fler block individuellt eller för hela gruppen genom utforskning på den stora skärmen. När de flesta eleverna hade kodat sin berättelse, och hade lagt till lite kod i varje scen, samlade läraren återigen hela klassen framför den stora skärmen. En elevs arbete visades och hela klassen tittade på den kodade berättelsen. Emellanåt var berättelserna mycket korta och avslutades på några sekunder. En gemensam diskussion om hur man kan utöka berättelsen gav eleverna möjlighet att lära av varandra samt upptäcka fler block och deras funktioner. I lärarens planering ingick att introducera nya DT-begrepp, mer specifikt villkorssatser (om-uttalanden) genom att visa eleverna hur en sprajt i taget kunde röra sig. Villkor är ett begrepp som kan framstå som abstrakt och obegripligt för barn och yngre elever i förskoleklass, men när det införs vid rätt tidpunkt och sätts i ett sammanhang som i exemplet ovan blir det mer konkret. Villkor infördes med hjälp av kuvertblocken, dvs. om en sprajt får ett kuvert, ska en specifik händelse sättas igång. Om det behövdes, förklarades och praktiserades detta också på analoga sätt. Läraren förberedde till exempel färgade pappers-kuvert som innehåller kod (kommandon) och agerade tillsammans med eleverna de givna kommandona.

Efter utvärdering av dessa aktiviteter och att eleverna 'hängde med' introducerade läraren loopar. Baserat på våra observationer verkade eleverna lätt förstå kommandot som gör att deras berättelse gör en kontinuerlig loop, medan gruppering av några block i ett loopblock var svårare att förstå. Många av eleverna såg inte syftet med att använda loopkommandot när de skapade sina berättelser och det var en utmaning för läraren att hitta specifika exempel som eleverna kunde relatera till. Vi kunde dra slutsatsen att det verkade lättare för eleverna att förstå en längre sekvens av block/kod, som representerar en linjär tidslinje. En kortare version som innehöll samma kod men med färre block var för många krångligare att konstruera och följa. För några elever var det inte alls svårt.

Förutom ovan nämnda svårigheter kan vi identifiera ett antal fördelar med att använda ScratchJr för att främja yngre elevers DT. En sådan fördel är att ScratchJr är öppet och har låg tröskel och högt i tak (Brennan & Bresnick, 2021), vilket stödjer elever på olika nivåer och med olika förmågor. Förskoleklasseleverna kan göra små introduktionsprojekt liksom mer komplexa projekt och därigenom stöds progression. En annan identifierad fördel är att de kan arbeta på ett multimodalt och kreativt sätt när de engagerar sig i programmering, vilket enligt våra observationer verkar vara mycket engagerande. Dessutom omfattar kombinationen av berättande och programmering till exempel språkkunskaper, fantasi, sekvensering, problemlösning, trial and error och ett antal grundläggande färdigheter för DT. Erfarenheterna med ScratchJr betonar dock ett stort behov av struktur och tydliga instruktioner, liksom stegvis stöttning och introduktion av begrepp.

DISKUSSION

I DENNA ARTIKEL rapporterar vi om erfarenheter från ett nationellt Ifous-program som genomfördes under tre år. Mer specifikt presenterar vi hur programmering med fokus på elevers datalogiskt tänkande introducerades i tre förskoleklasser under dessa år. Vi visar också vilka datalogiska begrepp och metoder som är möjliga för elever att lära sig genom undervisningen. Vi redogör även för vilka för- och nackdelarna är i relation till DT i den undervisning som växt fram. I diskussionen hänvisar vi till de olika program och verktyg som använts i förskoleklasserna.

Genom att utgå från Brennan och Resnicks (2012) datalogiska ramverk drar vi slutsatsen att förskoleklasselever klarar mer än vad både lärare och forskare förväntade sig i början av det treåriga programmet. För oss är det är uppenbart möjligt att introducera och få yngre elever att utveckla kunskaper och färdigheter relaterade till flera grundläggande DT-begrepp, -praxis och -perspektiv (se tabell 1), redan i förskoleklassen, vilket bekräftar tidigare forskning (Papadakis, Kalogiannakis , & Zaranis, 2016; Caballero-González, Muñoz, & Muñoz-Repiso, 2019; Angeli & Valanides, 2020). Vi kan också dra slutsatsen att processen att utveckla DT-färdigheter också utvecklade dessa elevers digitala kompetens, i linje med forskning utförd av Otterborn, Schönborn & Hultén (2019), särskilt när

Tabell 1: TDT-begrepp, -praxis och -perspektiv som elever kan utveckla i förskoleklassen.

DT-BEGREPP	DT-PRAXIS	DT-PERSPEKTIV
Sekvensering	Testning	Uttrycka sig
Villkorssatser	Felsökning	Samarbete
Loopar	Återanvända	Ifrågasätta
Händelser	Remixning	

man använder verktyg som ScratchJr som möjliggör multimodala skapelser.

För att utveckla färdigheter i DT drar vi dock slutsatsen att vissa betingelser och krav är nödvändiga. För det första var det fördelaktigt att ta utgångspunkt i ett ramverk för datalogiskt tänkande och följaktligen redan från början ha ett uttryckligt fokus på att utveckla DT-färdigheter. Detta bör ses som en kontrast till tillvägagångssätt där ett programmeringsverktyg är utgångspunkten (att införa ett verktyg som yngre elever kan utforska fritt) eller att införa programmering som ett verktyg för att lära sig ämnesinnehåll som matematik.

För det andra drar vi slutsatsen att lärarens roll är betydelsefull. Det är avgörande att lärare tar utgångspunkt i en förståelse för vilka fördelar olika programmeringsspråk, miljöer och verktyg har för att främja DT-färdigheter och i vilken ordning dessa bör införas. Baserat på erfarenheterna från detta treåriga program drar vi slutsatsen att programmeringsverktyg kan stödja olika DT-färdigheter, -praktik och -perspektiv, i olika utsträckning. Således betonar vi vikten av att sekvensera användningen av programmeringsverktyg på ett systematiskt och medvetet sätt för att ta hänsyn till förskoleklasselevers kapacitet och utveckling. Dessutom drar vi slutsatsen att förskoleklasselever har förmågan att lära sig blockprogrammering och att gå utöver analoga eller robotorienterade aktiviteter som har dominerat programmeringsaktiviteter i förskolan (Otterborn, Schönborn & Hultén, 2019). I förhållande till lärarens erfarenheter och ett verktygsperspektiv för programmering i förskoleklassen konstaterar vi följande:

Robotar som Bluebot kan vara ett utmärkt verktyg för att introducera programmering i förskoleklassen, eftersom de är påtagligt fysiska och konkreta. De stöttar meningsfullt samarbete kring problemlösning, är mycket engagerande för yngre elever och möjliggör utveckling av grundläggande DT-färdigheter som sekvensering, testning och felsökning.

Efter roboten är Bluebot-appen en fördelaktig övergång till blockprogrammering, eftersom det finns en begränsad mängd block och de liknar Bluebot-roboten. Appen gör det möjligt för lärare att skapa egna lektioner som kan vara skräddarsydda för olika tematiska arbeten. Det finns också inbyggda utmaningar i appen. I denna programmiljö kan elevernas grundläggande DT-färdigheter stabiliseras.

Code.org-miljön tillåter öppna och strukturerade lektioner. Om lärare föredrar förutbestämda lektionsplaneringar ger den här miljön en struktur som stöttar elevernas möjligheter att lära sig grundläggande DT-begrepp stegvis.

En introduktion till blockprogrammeringsspråk med Bluebot-appen och eventuellt code.org verkar underlätta övergången till ScratchJr, som är mer öppen och tillåter eleverna att arbeta med programmering på ett kreativt sätt baserat på eget intresse och utifrån egen drivkraft. Vi har dock observerat att om man börjar i en öppen miljö som ScratchJr kan det leda till att elever blir distraherade och spenderar för mycket tid på att designa sprajts och ägna sig åt mer estetiska aspekter, på bekostnad av kodning och utveckling av DT-färdigheter. Vi ser att det var betydelsefull att utforskningen av ScratchJr gjordes på ett kontrollerat sätt. Det vill säga att läraren didaktiskt planerar stöttning och sekvensering men samtidigt låter eleverna arbeta relativt fritt och utforskande. Läraren behöver sätta tydliga mål, systematisk stegvis introducera DT-begrepp och -praxis, samt planera stöttning och ha gemensamma genomgångar och repetitioner.

Generellt, oavsett vilket programmeringsverktyg och vilken miljö som används, drar vi slutsatsen att utveckling av förskoleklasselevers DT kräver systematisk didaktisk planering som inkluderar att: 1) presentera och fokusera på begrepp och praxis på ett reflekterande och stöttandet sätt, 2) ha en utvecklingsplan för progression och aktiviteter, 3) vid behov kombinera analoga och digitala metoder, 4) och viktigast av allt, att använda elevaktiva arbetssätt, dvs att lärandeaktiviteter äger rum i en miljö som eleverna uppfattar som tillförlitlig och som möjliggör kreativ utforskning och experimenterande genom trial and error. Dessutom bör läraren sträva efter att låta de yngre elevernas personliga mål och intressen få utrymme. Framtida forskning behöver dock djupare studera hur specifika DT-begrepp och praxis didaktiskt kan arbetas med i förskoleklassen.

REFERENSLISTA

- Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. ***Computers in Human Behavior, 105***, 105954.

- Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. ***Learning & Leading with Technology, 38***(6), 20–23.

- Wing, J. M. (2006). Computational thinking. ***Communications of the ACM, 49***(3), 33–35

- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. ***Acm Inroads, 2***(1), 48–54.

- Caballero-González, Y. A., Muñoz, L., & Muñoz-Repiso, A. G. V. (2019, October). Pilot Experience: Play and Program with Bee-Bot to Foster Computational Thinking Learning in Young Children. In ***2019 7th International Engineering, Sciences and Technology Conference (IESTEC)*** (pp. 601–606). IEEE.

- Ching, Y. H., Hsu, Y. C., & Baldwin, S. (2018). Developing computational thinking with educational technologies for young learners. ***TechTrends, 62***(6), 563–573.

- Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. ***Journal of Educational psychology, 76***(6), 1051.

- Department for Education, ***"National curriculum in England: Computing programmes of study,"*** 2013. https://www.gov.uk/government/publications/national-curriculumin-england-computing-programmes-of- study

- Elkin, M., Sullivan, A., & Bers, M. U. (2016). Programming with the KIBO robotics kit in preschool classrooms. ***Computers in the Schools, 33***(3), 169–186.

- Falkner, K., Vivian, R., & Falkner, N. (2014, January). The Australian digital technologies curriculum: challenge and opportunity. In ***Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148*** (pp. 3–12). Australian Computer Society, Inc.

- Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. ***Computers & Education, 63***, 87–97.

- Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. ***Educational Researcher, 42***(1), 38–43.

- Hammersley. M, & Atkinson, P. (1995). ***Ethnography: principles in practice***. Biddles Ltd. Guildford and King's Lynn: London.

- Hedrén, R. (1990). ***LOGO-programmering på mellanstadiet. En studie av fördelar och nackdelar med användning av LOGO i matematikundervisningen under årskurserna 5 och 6 i grundskolan***. Linköping: Department of Education (diss).

- Heintz, F., Mannila, L., & Färnqvist, T. (2016, October). A review of models for introducing computational thinking, computer science and computing in K-12 education. In ***Frontiers in Education Conference (FIE)***, 2016 IEEE (pp. 1–9). IEEE.

- Horn, M. S., AlSulaiman, S., & Koh, J. (2013, June). Translating Roberto to Omar: Computational literacy, stickerbooks, and cultural forms. In ***Proceedings of the 12th International Conference on Interaction Design and Children*** (pp. 120–127). New York, NY: ACM.

- Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. ***Early Childhood Education Journal, 41***(4), 245–255.

- Kong, S. C. (2016). A framework of curriculum design for computational thinking development in K-12 education. ***Journal of Computers in Education, 3***(4), 377–394.

- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. ***Computers in Human Behavior, 41***, 51–61.

- McNerney, T. S. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. ***Personal and Ubiquitous Computing, 8***(5), 326–337. DOI 10.1007/s00779-004-0295-6

- Miller, G. E., & Emihovich, C. (1986). The effects of mediated programming instruction on preschool children's self-monitoring. ***Journal of Educational Computing Research, 2***(3), 283–297.

- Otterborn, A., Schönborn, K. J., & Hultén, M. (2019). Investigating Preschool Educators' Implementation of Computer Programming in Their Teaching Practice. ***Early Childhood Education Journal, 1–10***.

- Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study. ***International Journal of Mobile Learning and Organisation, 10***(3), 187–202.

- Papert, S. (1980). ***Mindstorms: Computers, children, and powerful ideas***. NY: Basic Books, 255.

- Rose, S., Habgood, J., & Jay, T. (2017). An exploration of the role of visual programming tools in the development of young children's computational thinking. ***Electronic journal of e-learning, 15***(4), 297–309.

- Ryokai, K., Lee, M. J., & Breitbart, J. M. (2009, October). Children's storytelling and programming with robotic characters. In ***Proceedings of the seventh ACM conference on Creativity and cognition*** (pp. 19–28).

- Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. ***Computers & Education, 97***, 129–141.

- Seehorn, D. W., Stephenson, C., Pirmann, T. R., & Powers, K. (2013, March). CSTA CS K-12 instructional standards and CS curriculum. In ***Proceeding of the 44th ACM technical symposium on Computer science education*** (pp. 746–746). ACM.

- Sullivan, A. A., Bers, M. U., & Mihm, C. (2017). Imagining, playing, and coding with KIBO: using robotics to foster computational thinking in young children. ***Siu-cheung KONG The Education University of Hong Kong, Hong Kong, 110***.

- Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks [Electronic version]. ***The Journal of the Learning Sciences, 17***(4), 517–550.

- Yoshida, M. (2012). Mathematics lesson study in the United States: current status and ideas for conducting high quality and effective lesson study ***International Journal for Lesson and Learning Studies, Vol. 1*** No. 2, pp. 140–152.

- Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. ***Computers & Education, 141***, 103607.

SKOLPORTEN