INSTITUTIONEN FÖR
TILLÄMPAD IT

LÅG- OCH MELLANSTADIELÄRARES RESONEMANG OCH TILLÄMPNING AV PROGRAMMERING I MATEMATIKÄMNET

Ksenija Peggar

Arkan Shefram

Uppsats/Examensarbete:	15 hp
Program och/eller kurs:	Lärande, kommunikation och IT
Nivå:	Avancerad nivå
År:	2020
Handledare:	Berner Lindström
Examinator:	Ylva Hård af Segerstad
Rapport nr:	2020:014

Sammanfattning

Skolverket inför programmering och digitala verktyg i kursplanen för matematikämnet på grundskolan 2018 på uppdrag från Regeringen. Anledningen till beslutet är att vi lever i alltmer digitaliserat samhälle som leder till ett ökat behov av digitala kunskaper i de flesta yrken. Elever behöver utrustas med de kunskaper och kompetenser som krävs i det samhälle de ska verka i.

Den här kvalitativa studien syftar till att undersöka hur matematiklärare på låg- och mellanstadiet resonerar kring programmeringskravet i läroplanen. Vi har använt intervju som metod. Det empiriska materialet har bestått av intervjuer med åtta lärare i årskurs 2-6. Resultatet i studien visar att matematiklärarna anser att införandet av programmering i läroplanen är relevant i relation till digitaliseringsutvecklingen i samhället. Alla matematiklärare resonerar om att de har använt programmering någon gång i sin undervisning men få har använt det som hjälpmedel i matematikundervisningen. Ett annat resonemang beskriver att matematiklärarna anser att programmeringen konkretiserar matematiken i elevers lärande. Detta stöds även i tidigare forskning om att programmering utvecklar konkret förmågan att resonera, tänka logiskt och planera.
Matematiklärarna menar också att programmeringen kan undervisas med analoga övningar, i digitala programmeringsmiljöer och som ämnesinnehåll i läromedlen. Analysen av intervjuerna belyser även att lärarna resonerar på olika sätt och kan eventuellt kopplas till deras datavetenskapliga bakgrund.

Den här studien visar att matematiklärarna följer programmeringskravet i läroplanen som står i det centrala innehållet för matematikämnet. Däremot tillämpas programmeringen i begränsad utsträckning som det står i kursplanens syfte. Studien visar också att lärarna resonerar på olika sätt kring programmeringskravet och därmed görs det på olika sätt och i olika utsträckningar.

Nyckelord

Programmering, matematik, grundskola, läroplan, datalogiskt tänkande, resonemang

Title

Teachers' reasoning and application of computing in mathematics: a qualitative study with some primary school teachers.

Abstract

The National Agency for Education introduces programming and digital tools in the syllabus for the mathematics at the primary school 2018 at the request of the Swedish Government. The reason for this decision is that we live in an increasingly digitalized society that leads to an increased need for digital knowledge in most professions. Pupils need to be equipped with the knowledge and skills required in the society in which they live.

This qualitative study aims to investigate how mathematics teachers working within the grades 2-6 reason about the programming requirement in the curriculum and how they apply it in their teaching. The empirical material consists of semi-structured interviews with eight teachers. The study shows that mathematics teachers consider that programming introduction in the curriculum is relevant in relation to digitalization development in society. All mathematics teachers´ reason that they have used programming at least once in their teaching but few have used it a means to teach about mathematics. Another reasoning shows that mathematics teachers consider that the programming concretizes mathematics in student learning. This is also supported in previous research which shows that programming develops concrete ability to reason, think logically and plan. Regarding the practical application of the requirement, the study shows that mathematics teachers teach programming as an analog activity, also they do it in the digital programming environments, and as a subject content in its own matter as presented in the teaching materials. The analysis of the interviews also revealed that the teachers reasoned in many different ways which in this study could possibly be linked to teachers' different backgrounds in computer science.

To summarize, the study shows that mathematics teachers follow the programming requirements in the central curriculum for mathematics, although to a limited extent when the purpose of the syllabus is considered. The study also shows that teachers reason in different ways about the programming requirement and that the requirement has been implemented in different ways and to different extents.

Keywords

Computing, mathematics, primary school, curriculum, computational thinking, reasoning

Innehållsförteckning

1 Inledning

Vi lever i en alltmer digitaliserad värld och nya kunskaper krävs för att kunna delta fullt ut. Europaparlamentet (2006/962/EG) definierar digital kompetens som en av de nyckelkompetenser som krävs i den digitaliserade värld. Till följd av den pågående digitalisering har flera länder börjat införa datavetenskap i olika former in i grundskolans läroplan. Datavetenskap kan enkelt definieras som ett vetenskapligt område och ett akademiskt ämne som handlar om praktisk kunskap om informationsteknologi och programutveckling.

I Sverige sattes denna process igång 2012 då den svenska Regeringen konstaterade att IT-politiken behöver breddas och omfatta flera än tidigare områden. Med bakgrund mot detta tillsattes en kommission som skulle arbeta strategiskt med långsiktiga it-politiska frågor - Digitaliseringskommissionen. Enligt Digitaliseringskommissionen (SOU 2014:13) spelar skolan en viktig roll för att förberedda ”barn och ungdomar för framtidens samhälle” (s.129). Efter en utredning om hur det ser ut både i Sverige och i andra länder i Europa slår Digitaliseringskommissionen fast att digital kompetens och programmering bör skrivas in i skolans läroplan. Detta innebär att elever även i de tidigare årskurser ska tillägna sig kunskaper om programmering. I alla yrken i framtiden kommer det behövas kunskap om digitala verktyg och kompetens och förståelse. Programmering kommer därför ses som en viktig faktor i elevernas framtida karriär. Samtidigt är denna utveckling en nödvändighet för att Sverige ska ha tillräckligt med kompetenta experter och ska kunna behålla sin konkurrenskraft i världen (SOU 2014:13).

Programmering som en del av digital kompetens infördes 2018 i grundskolans läroplan på uppdraget från Regeringen (Skolverket, 2017). Detta är ingen ny kunskap inom datavetenskap men nu har den kommit till grundskolan och lärare bör använda programmering i sin undervisning. Ändringarna innebär också att lärare behöver anpassa sin undervisning. Programmering ska bl.a. användas tillsammans med digitala verktyg i matematikämnet. Texten i kursplanens syfte för ämnet matematik lyder följande:

> Genom undervisningen ska eleverna ges förutsättningar att utveckla förtrogenhet med grundläggande matematiska begrepp och metoder och deras användbarhet. Vidare ska eleverna genom undervisningen ges möjligheter att utveckla kunskaper i att använda digitala verktyg och programmering för att kunna undersöka problemställningar och matematiska begrepp, göra beräkningar och för att presentera och tolka data. (Skolverket, 2019, s.54)

Inom forskarkretsar kopplas programmering ofta till datalogiskt tänkande och problemlösning. En av många definitioner av datalogiskt tänkande beskriver det som ett särskilt sätt att tänka på, där den viktigaste komponenten är att tänka som en datorvetare när något problem ska lösas (Sung, Ahn & Black, 2017). Kopplingen mellan matematik, programmering, datalogiskt tänkande och problemlösning är viktigt, men inte kristallklar. Det går dock inte undvika att komma i kontakt med frågan i och med att programmering har införts i matematikundervisning.

Det har gått två år sedan förändringarna gjordes. Skolan och lärarna måste efterleva styrdokumenten, det är därför viktigt att få lära mera om hur ändringarna tas emot och hanteras av lärare. Denna uppsats, tillsammans med andra studier, kan ses som stickprov på matematiklärares perspektiv på det nya programmeringskravet samt på hur det hanteras i praktiken. Studien fokuserar på åtta matematiklärare i årskurser 2-6. Eftersom programmering har tidigare inte varit en del av läroplanen för grundskolan, är det ett helt nytt område för många lärare. Det är också en anledning varför det är särskilt intressant att undersöka just denna grupp.

2 Syfte och frågeställningar

Syftet med denna studie är att undersöka hur matematiklärare i grundskolans årskurser 2-6 resonerar kring och arbetar med programmeringskravet i läroplanen för ämnet matematik i skolan. De mer specifika forskningsfrågorna är följande:

- Hur resonerar lärare kring det programmeringskrav som införts i matematikämnet?
- I vilken utsträckning ser lärare att matematikundervisningen kan stödjas av programmering?
- Hur har lärare infört programmering i matematikundervisningen?

Denna studie syftar till att göra varken skillnad eller en jämförelse mellan låg- och mellanstadiet och inte heller mellan lärarna själva. Studien fokuserar först och främst på att identifiera och beskriva vilka resonemang som finns gällande införandet av programmering i matematikämnet.

3 Bakgrund

3.1 Programmering i läroplanen

I Sverige kan kopplingen mellan matematik och programmering spåras tillbaka till 1975 då Björk m.fl. (1975, i Rolandsson och Skoghs, 2014) i sin artikel kom fram till att de elever som fick interagera med programmering hade mera positivt förhållningssätt till matematik. Det var på den tiden viktigt att höja motivationen för att lära sig matematik eftersom antalet gymnasieelever som valde matematik och naturvetenskap hade sjunkit. Dessutom, ansågs programmeringen vara fördelaktig för att lära sig problemlösning i matematik, eftersom att programmera innebär att skapa algoritmer som kan användas för att lösa problem. Allt detta skedde samtidigt som vikten av att kunna lösa problem blev allt viktigare för att Sverige skulle kunna behålla ledande positioner inom industriområdet (Rolandsson & Skogh, 2014).

Rolandsson och Skogh (2014) visar att denna koppling mellan programmering och matematik i den svenska läroplanen har funnits länge. Arbetsgrupper som jobbade med läroplanen för ämnet datavetenskap på 70-talet hade åsikten att det är viktigt att se datalära i ett bredare perspektiv och som ett konceptuellt ramverk för andra ämnen, exempelvis för matematik-, natur- och samhällsämnen (Rolandsson & Skogh, 2014). Programmering ansågs kunna erbjuda ett praktisk och *hands-on* sätt att lära matematiska begrepp. När vi tittar på förändringar i läroplanen som trädde i kraft i 2018, blir det tydligt att införandet av programmering i grundskolan är mest synligt i ämnena matematik och teknik. Inom matematik som är mest intressant för denna uppsats, ses programmering som ett redskap att hjälpa lära sig matematik (Ahmed, Nouri, Zhang & Norén, 2020). Det går därför att säga att den förändrade läroplanen i Sverige följer den svenska traditionen hur man ser på datalära.

Sverige är dock inte det enda landet som har infört programmering. Kjällander, Åkerfeldt och Petersens (2016) rapport visar att det finns en stor variation i hur olika länder hanterar frågan om programmering och hur ämnet införs: Från vilken ålder? Som ett eget ämne eller som en del av redan befintliga ämnen, främst matematik? England, exempelvis, har sen flera år tillbaka haft IKT som ett eget ämne i grundskolan, men 2014 ändrades läroplanen och stort fokus lades på kodning och datalogiskt tänkande (Brown, Sentance, Crick & Humphreys, 2014; Wong, Cheung, Ching & Huen, 2016), Australien lägger också stort fokus på datalogiskt tänkande medan Finland liksom Sverige fokuserar på digitala kompetenser och inför programmering i de redan befintliga ämnen (Nouri, Zhang, Mannila & Norén, 2020).

Förutom att det finns en viss tradition att se på datalära i bredare sammanhang, urskiljer Heintz, Mannila, Nordén, Parnes och Regnell (2017) tre konkreta anledningar till varför programmering i Sverige inte blev ett eget ämne utan är en del av de redan befintliga ämnen, främst av matematik, teknik och samhällsvetenskapliga ämnen: 1) brist på utrymme för ett nytt ämne; 2) önskan att nå ut till flera elever som annars aldrig skulle få möjlighet att prova på programmering; 3) målet att introducera datalogisk tänkande, vilket anses vara allt viktigare i dagens samhälle på grund av att datalogiskt tänkande erbjuder ett sätt att lösa

komplexa problem med hjälp av en dator. (Heintz, Mannila & Färnqvist, 2016). Det sista har dock inte fått lika stort genomslag som forskarna hade önskat. Istället för att lägga fokus på datavetenskap i sin helhet, valde regeringen att fokusera på två avgränsade aspekter och uttryckligen angav två syften med läroplanens reviderings: 1) stärka elevernas digitala kompetens och 2) införa programmering i grundskolan (Heintz m.fl., 2017). Forskarna däremot menade att programmering är bara en del av datavetenskap och om det bara är den delen som införts i läroplanen, så mycket mer lämnas utanför:

> If you only focus on programming and code, you risk missing out on general and useful skills such as dividing problems in smaller parts, solving problems in creative ways, finding patterns, thinking logically, designing algorithms, working in a structured manner, making generalisations and finding models. (Heintz, Mannila, Nygårds, Parnes & Regnell 2015, s.118)

Dessa praktiker och färdigheter refererar forskarna till som datalogiskt tänkande. Sverige är dock inte ensam om att utelämna explicita formuleringar om datalogiskt tänkande i läroplanen, detta är ganska vanligt konstaterar Heintz m.fl. (2016) i sin översikt över vilket uttryck tar datavetenskap i tio länder runt om i världen. De nämner också att även om datalogiskt tänkande inte står utskrivet svart på vit så finns det ändå med i någon annan form.

Programmering följaktligen är infört i läroplanen först i form av stegvisa instruktioner som utgör basis för programmering i årskurs (vidare å.k.) 1-3. Sedan flyttas fokus till skapandet av algoritmer och deras användning i programmering i å.k. 4-6 och slutligen övergår det till användandet av algoritmer i den matematisk problemlösningen i å.k. 7-9 (Heintz m.fl., 2016).

Här följer ett utdrag ur kursplanen för ämnet matematik och årskurserna 2-6 (Skolverket, 2019). Årskurserna 7-9 tas inte med eftersom de ligger utanför ramen för denna uppsats.

Centralt innehåll

Årskurs 1-3:
Algebra

- Hur entydiga stegvisa instruktioner kan konstrueras, beskrivas och följas som grund för programmering. Symbolers användning vid stegvisa instruktioner.

Årskurs 4-6:
Algebra

- Hur algoritmer kan skapas och användas vid programmering. Programmering i visuella programmeringsmiljöer.

Syftet med att använda programmering och digitala verktyg beskrivs under syftet i kursplanen för ämnet matematik (Skolverket, 2019, s.54):

> Vidare ska eleverna genom undervisningen ges möjligheter att utveckla kunskaper i att använda digitala verktyg och programmering för att kunna undersöka problemställningar och matematiska begrepp, göra beräkningar och för att presentera och tolka data.

De ovan presenterade skrivningarna gällande införande av programmering kan förstås som breda riktlinjer som handlar om programmering, algoritmer och problemlösning och som lärare själva behöver tolka och förstå. I början kan det vara positivt att inte avgränsa för mycket och inte heller skriva explicita kunskapskrav (Heintz m.fl., 2017). Vagheten i kursplanerna ger lärare den frihet de behöver att närma sig ett nytt område. Istället för att fokusera på detaljerade krav kan de börja med att göra så mycket de själva är i stånd med. I takt med att lärare blir allt mera kunniga kan de täcka ämnet på ett djupare sätt (Heintz m.fl., 2017). Uteblivna kunskapskrav i kursplanen kan dock medföra risken att programmering inte ses som ett obligatoriskt inslag och därmed inte införs. Dessutom kan det vara orsaken till att undervisningens kvalité varierar mycket och styrs främst av lärares intresse och tiden hen disponerar och inte av konkreta till ämnet relaterade mål (Heintz m.fl., 2015).

För att hjälpa lärare att tolka och förstå digital kompetens inom ramen för grundskolans reviderade läroplan, gav Skolverket ut ett stöddokument. Att använda stöddokument istället för att skriva om hela läroplanen är ett bra sätt att hålla läroplanen uppdaterad och samtidigt undvika att skriva om den vilket är ett stort arbete; tanken med skrivningarna i läroplanen är ju att de ska hålla i flera år (Heintz m.fl., 2017). I stöddokumentet för grundskolan *Få syn på digitaliseringen på grundskolenivå* förtydligar Skolverket att: *”*I läroplanerna finns programmering med som en del av den digitala kompetens som eleverna ska ges möjlighet att utveckla” (2017, s.10). Vidare beskrivs det vad programmering innebär:

> I programmering ingår att skriva kod, vilket har stora likheter med generell problemlösning. Det handlar bland annat om problemformulering, att välja lösning, att pröva och ompröva samt att dokumentera. Men programmering ska ses i ett vidare perspektiv som även omfattar kreativt skapande, styrning och reglering, simulering samt demokratiska dimensioner. Det här vidare perspektivet på programmering är en viktig utgångspunkt i undervisningen och programmering ingår därmed i alla aspekter av digital kompetens. (Skolverket 2017, s.10)

Att programmering och problemlösning har beröringspunkter bekräftas av Helenius, Misfeldt, Rolandsson och Ryan (2018) i ett stödmaterial för gymnasiet *Om programmering i matematikundervisning* för gymnasieskolan. Författarna menar att programmering består av ett antal ordnade stegvisa instruktioner för att utföra en given uppgift eller att lösa ett problem. Datalogiskt tänkande som inom forskningskretsar också brukar kopplas till programmering och problemlösning nämns däremot inte i kursplanen för matematik. Begreppet diskuteras dock i stöddokumentet *Få syn på digitaliseringen på grundskolenivå på följande sätt:*

> Inom datavetenskapen används begreppet datalogiskt tänkande. Det handlar bland annat om problemlösning, logiskt tänkande, att se mönster och att skapa algoritmer som kan användas vid programmering. Även detta är exempel på kunskaper som beskrivs i olika delar av läro- och kursplanerna. (Skolverket 2017, s.9)

Detta är det enda tillfället då datalogiskt tänkande nämns explicit, men citaten ovan visar samtidigt att det som menas med datalogiskt tänkande finns utskrivet på annat sätt, d.v.s. genom att nämna problemlösning, logiskt tänkande, mönster, algoritmer och programmering.

Hur bör lärare gå till väga i sin undervisning av matematikämnet rent praktiskt förblir emellertid en tolkningsfråga.

4 Forskningsöversikt

Vi kommer först beskriva hur programmering kopplas till matematik och datalogiskt tänkande genom exempel på både empiriska och konceptuella studier. Sen följer en översikt över vad forskningen säger om lärares förhållningssätt till införande av programmering i läroplanen.

4.1 Programmering och matematik

Kopplingar mellan programmering och matematik undersöktes redan på 1970-talet då Feurzeig och Papert skapade ett programmeringsspråk LOGO (Feurzeig, Papert & Lawler, 2011). LOGOs skapare menade, att matematik inrymmer många abstrakta begrepp som är svåra att förstå och beskriva men som blir lättare att begripa genom att arbeta med programmering som också bygger på många gemensamma koncept. Det blir lättare på grund av att abstrakta begrepp inom programmering används på ett konkret sätt och därför blir synliga (Feurzeig m.fl., 2011).

LOGO kan beskrivas som ett digitalt verktyg som möjliggör för elever att lära sig tänka matematiskt och logiskt. Detta i sin tur kan bidra till att kunna lösa matematiska problem (Feurzeig m.fl., 2011). Vidare har LOGO en motiverande effekt eftersom det är roligt att jobba med LOGO, eleverna jobbar enligt *hands-on* arbetssätt, och därför kan användandet av programmering sänka hinder som elever upplever när de lär sig matematik (Feurzeig m.fl., 2011).

Forsström och Kaufmann (2018) i sin litteraturöversikt tittade på studier gjorda mellan 1995 och 2018, som handlade om användning av programmering och robotar i matematikundervisning. Syftet var att undersöka om det finns någon potential i denna användning. Forsström och Kaufmann (2018) definierar programmering som process som handlar om att utveckla och implementera instruktioner i datorprogram så att dator ska kunna utföra specifika uppgifter, lösa problem och stödja mänsklig interaktion. Forskaren har tre utgångspunkter i sin studie: motivation att lära sig matematik, elevernas prestationer i matematik, samarbete mellan elever och lärares förändrade roll. Resultat visar att det inte är möjligt att generalisera och påstå att användandet av programmering påverkar motivation positivt. Resultat gällande elevernas framgångar i matematik varierade i olika åldersgrupper och olika teman, därför går det inte dra slutsatsen att programmering alltid resulterade i en förbättring. Det gick dock identifiera geometri som passande tema att undervisa i med hjälp av Scratch och robotar. När lärare får en roll som facilitator och problemlösare och med en avslappnad stämning i klassrummet möjliggjordes för annars lägre presterande elever att komma fram och delta mera aktivt. I sin sammanfattning skriver Forsström och Kaufmann (2018) att, fastän det var flera studier som visade positiv utfall gällande motivation och elevernas framgångar, så låter sig inte resultatet generaliseras och flera studier inom området behövs.

Lie, Hauge och Meaney (2017) gjorde en empirisk studie för att undersöka matematisk tänkande och problemlösningsprocess som uppstår vid programmering och se vad

programmering kan ha för värde för att lära sig matematik. Forskarna ser programmering som hjälpmedel att utveckla datalogiskt tänkande och matematiskt tänkande. De menar att matematik finns inom programmering implicit. Sett i ett historiskt perspektiv skapades datorer för att utföra matematiska beräkningar och det är särskilt genom algoritmiska konstruktioner att programmering kan kopplas till aritmetiskt, matematisk och datalogiskt tänkande. Genom att programmera och hitta relevanta kopplingar till matematiska begrepp och matematiskt språk blir man bättre på programmeringsspråk och vice versa – genom att använda datalogiskt tänkande vid programmering förbättras inlärningen av matematik. Studien gjordes genom att barn fick programmera i Scratch. Två exempel presenteras i resultaten. I det första exemplet kunde man konstatera att matematiska kommandon användes för att lösa ett datalogiskt problem när elever skulle hantera figurer i Scratch men samtidigt menar författarna att det är svårt att säga hur den nyvunna kunskapen skulle kunna överföras och användas inom matematikenslärande. I det andra exemplet fanns det också inslag av datalogiskt tänkandet och sammanfattningsvis menar forskarna att barn utvecklade de matematiska begrepp som är kopplade till programmering och utvecklade flera matematiska kompetenser såsom resonerande, kommunikation, modellering och problemlösning.

Strawhacker och Bers (2018) i sin studie undersökte vilka kognitiva aspekter barnen utvecklar när de programmerar. Barnen fick göra uppgifter i ScratchJr och sen mättes deras kunskaper i programmeringsspråket samt deras sätt att resonera undersöktes. Forskarna menar att vanligtvis undersöks det hur det matematiska tänkandet samverkar med att lära sig programmera och att tidigare forskning har undersökt exempelvis hur programmering bidrar till utveckling av kompetenser i matematik, logik och sekventiell uppställning (*sequential ordering*). Programmering brukar förknippas med dessa kompetenser eftersom de innehar tänkandet som är logiskt och detaljerat. Strawhacker och Bers (2018) menar dock att koppla programmering endast till logik är ett snävt sätt att se på programmering. I sin studie utgår de från den motsatta utgångspunkten och lyfter fram den rumsliga, sociala och verbala aspekter som viktiga. De visar hur genom programmering kan barnen utveckla inte bara förmågan att förstå och följa instruktioner (genom exempelvis använda sin kropp i analoga övningar) men även förmågan att tänka abstrakt. Forskarnas slutsats är att det är viktigt att införa programmering i tidiga åldrar för att detta ska samverka med de andra kognitiva domänerna.

Armoni (2016) i sin artikel ifrågasätter relationen mellan kodning, programmering och datalogiskt tänkandet. Hon skriver bl.a. att det har blivit vanligt att kalla kodning för programmering fast de ligger på två olika abstraktionsnivåer. Programmering kan sägas ligga på mera avancerad nivå och det är missvisande att sätta ett lika med tecken mellan de två. Vidare kopplar hon till datalogiskt tänkande och menar att datalogiskt tänkande bör vara kärnan inom datavetenskap. I verkligheten är det dock vanligt förekommande att de läroplanen som sägas bygga på datalogiskt tänkandet utvecklas egentligen färdigheter i att programmera och inget annat.

Det finns få studier som undersöker relationen mellan programmering och andra färdigheter. Scherer (2016), exempelvis, lyfter frågan om huruvida färdigheter i programmering kan spilla över till andra ämnen och kontext och menar att flera empiriska studier som undersöker

denna fråga behövs. I artikeln fokuserar han på processer som sker när en programmerar och när en löser problem och menar att det finns många konceptuella likheter mellan dessa två processer (Tabell 1). Därför, enkelt uttryckt, är det sannolikt att de som kan programmera är bättre på att lösa problem.

Tabell 1. Processer inom programmering och problemlösning

Computer programming	Problem solving
■ Representing, storing, and retrieving information in order to understand the problem (i.e., knowledge acquisition of basic problem elements such as objects, relations, initial and final states of objects) ■ Discovering algorithms for information processes —finding a method in order to represent the real-world problem, develop, and execute an action plan and code ■ Evaluating the performance of the designed complex systems on the basis of the written code; bridging the gap between the problem statement and the solution	■ Exploring and understanding the problem (e.g., by decomposing the problem into sub-problems) ■ Representing and formulating the problem by creating representations of the problem situation and formulating hypotheses ■ Planning and executing the sequential steps to solve the problem ■ Monitoring progress and reflecting on the problem, the solution, and solution strategy

Kommentar. Anpassat från Scherer, 2016, s.3

Scherer, Siddiq och Viveros (2019) gjorde en meta-analys av 105 studier med syfte att undersöka om att lära sig programmera medför en ökad kognitiv förmåga som kan användas i andra sammanhang. Forskare nämner en handfull av tidigare studier som undersökte frågan, bl.a. en artikel av Liao och Bright som 1991 gjorde en meta-analys av studier mellan 1969-1989, och deras analys bekräftade att elever som lärde sig programmera även utvecklade förmågor som behövs i problemlösning, mera konkret förmågan att resonera, tänka logiskt och planera.

I sin metastudie där de (Scherer m.fl., 2019) analyserade hur programmering påverkar andra färdigheter, skriver forskarna att problemlösning och modellering är det som programmering och matematik har gemensamt, d.v.s. för att kunna programmera och för att kunna lösa matematiska problem behöver använda samma typ av färdigheter. De kallar det för ‘shared subskills’ och skriver att dessa möjliggör att en så kallad transfereffekt uppstår. I fotspår av Perkins and Salomon (1992 i Scherer m.fl., 2019) beskrivs transfer i Scherers m.fl. (2019) artikel som en situation där lärande som härstammar från en kontext påverkar lärande och prestationer i andra, kanske nya kontext. Men andra ord, en transfereffekt innebär att genom att lära sig programmera utvecklar elever vissa kognitiva färdigheter som sedan kan användas i andra sammanhang, exempelvis i matematik.

Vidare kopplar Scherer m.fl. (2019) programmering med datalogiskt tänkande. Datalogiskt tänkande är ett koncept som ofta används i relation till programmering och problemlösning. En av de mest ofta citerade definitionerna av datalogiskt tänkande kommer från Wings (2006) artikel och lyder följande: ”Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science” (s.33). Wing (2006) menar också att datalogiskt tänkande

är ett allmängiltigt förhållningssätt och en samling av färdigheter som alla, inte bara dataforskare, skulle ha nytta av att lära och kunna använda. Hennes artikel där hon introducerade datalogiskt tänkande och hennes syn på datalogiskt tänkande som något mycket viktigt har startat heta diskussioner kring vilken roll datalogiskt tänkande spelar inom utbildning (Sung m.fl., 2017). Ofta omnämns datalogiskt tänkande som något helt avgörande för framtiden.

Kritik mot Wings definitionen och den roll hon tillskriver datalogiskt tänkande kommer från diSessa (2018) som har utvecklat ett annat begrepp – *computational literacy*. Computational literacy ska inte förväxlas Computational literacy ska inte heller likställas med datalogiskt tänkande, som Grover och Pea (2013) gjorde i sin artikel. diSessa analyserar Wings definitioner av datalogiskt tänkande och ifrågasätter flera aspekter av den. diSessa menar att utgångspunkten i Wings skrivningar är datavetenskap[1] fastän samtidigt beskriver hon datalogiskt tänkande som något allmängiltigt[2]. Denna allmängiltighet möjliggörs genom att kärnan i datalogiskt tänkande enligt Wing är abstraktion, men diSessa menar att abstraktion betyder olika saker i olika domäner (exempelvis i matematik, fysik, datavetenskap) och kan därmed inte beskrivas som något allmängiltigt och som relevant för alla domäner. Följaktligen kan inte heller datalogiskt tänkande sägas kunna utveckla några generella egenskaper.

Vidare kritiserar diSessa (2018) även Wings beskrivning hur ett problem kan lösas[3], d.v.s. att problemet kan designas och lösningen kan planeras innan att själva lösningsprocessen har skett: ”One essentially never figures out how to solve an important problem of a discipline and maps out exactly how it can be solved before actually doing the solving” (diSessa, 2018, s.19). Detta menar diSessa betyder att Wings definition är inget allmänt sätt att lösa problem på utan ett sätt att lösa problem som är relevant inom en specifik domän, i det här fallet inom datavetenskap.

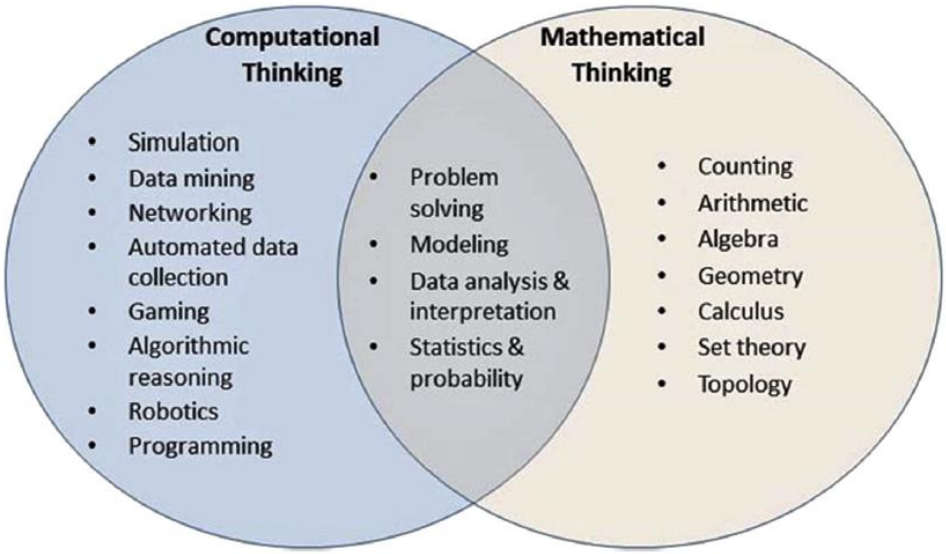
Fastän Wing var först att definiera datalogiskt tänkande, kan tankar att datalogiskt tänkande är något grundläggande inom matematikundervisning spåras tillbaka till Paperts arbete (1972 i Sung m.fl., 2017) där han beskriver användandet av datakunskap som en viktig del i utbildnings utveckling och modernisering.

För att komma tillbaka till Scherers m.fl. (2019) metastudie, förutom att beskriva vad programmering och matematik har gemensamt och förutom att koppla programmering med datalogiskt tänkandet, gör de ytterligare en koppling. Forskarna menar att problemlösning vid programmering kan kopplas till datalogiskt tänkande genom att det inom bägge processerna behövs samma typ av färdigheter.

[1] Computational thinking involves solving problems, designing systems, and understanding human behavior, *by drawing on the concepts fundamental to computer science*. (Wing 2008)
[2] The educational benefits of being able to think computationally—starting with the use of abstractions—enhance and reinforce intellectual skills, *and thus can be transferred to any domain*. (Wing 2014)
[3] Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer— human or machine—can effectively carry out. (Wing 2014)

Samtidigt påpekar forskarna att processerna inte är identiska. Men eftersom programmering innebär problemlösning, kan programmeringen användas för att lära om och lära ut datalogiskt tänkande, d.v.s. genom att skriva en kod och skapa program blir eleverna tvungna att använda datalogiskt tänkande.

Å andra sida jämför Shute, Sun och Asbell-Clarke (2017) datalogiskt tänkande med andra sätt att tänka på, exempelvis med matematiskt-, ingenjörs-, design- och systemtänkande. Kärnan i det matematiska tänkandet har tre beståndsdelar, varav problemlösning är en. I artikeln definieras problemlösning som den mest framträdande gemensamma nämnare i avseende till relation mellan matematiskt och datalogiskt tänkande (Wing, 2008 i Shute m.fl., 2017). Förutom problemlösning, andra beröringspunkter är modellering, dataanalys och tolkning, statistik och sannolikhet (Figur 1).



Figur 1. Likheter och skillnader mellan det datalogiska tänkande och matematiska tänkande (från Shute et al., 2017, s.146)

Weintorp m.fl. (2016) anser också att det finns en ömsesidig relation mellan matematik och datalogiskt tänkande. De bidrar till varandras förståelse. Datalogiskt tänkande bidrar till en djupare förståelse av matematik, och matematik ger en naturlig och meningsfull kontext att tillämpa datalogiskt tänkande på.

Denna korta genomgång visar att det finns en stark konceptuell koppling, om än omdebatterad, mellan programmering, matematik och datalogiskt tänkande där en gemensam nämnare är problemlösning.

4.2 Lärares resonemang kring programmering i grundskolan

Vi har inte hittat någon studie som skulle behandla lärares resonemang kring införandet av programmering i läroplanen.

Däremot hittade vi studier som handlar om lärares attityder och uppfattningar. Dessa studier är ofta pilotstudier där ett begränsat antal lärare har svarat på frågorna eller studier som föregås av en kurs eller kompetenslyft, riktat specifikt att lära om programmering och hur den kan användas i undervisning. I andra studier har under projektets gång forskargrupp gett stöd till lärare att undervisa på ett nytt sätt.

Funke, Geldreich och Hubwieser (2016) har exempelvis intervjuat sex lärare i Tyskland om deras attityder gentemot datavetenskap och programmering i grundskolan. Forskarna ville bygga upp en kurs i dessa ämnen för barn i åldrarna 8-10 år men ville först samla in lärares åsikter om det. Resultatet visade att lärare hade lite förståelse av vad datavetenskap innebär. De visste inte heller hur de skulle klara av att undervisa för deras kompetens inte räckte till. Samtidigt tyckte lärarna att det var viktigt att lära barnen programmera för att det både utvecklar deras sätt att tänka och förbereder dem för framtiden.

Duncan, Bell och Atlas (2017) i sin pilotstudie redogör för lärare i grundskolan deras erfarenheter att undervisa i datavetenskap, med fokus på datalogiskt tänkande och programmering. Studien inkluderade 22 lärare, 1000 elever och 9 skolor. Studien gjordes i Nya Zeeland innan det nya ämnet i grundskolan skulle införas. Lärare fick hjälp med lärresurser som inkluderade många analoga aktiviteter samt övningar i Scratch, ScratchJR och Bee-bot. I sin preliminära redogörelse rapporterar forskarna att lärare kände sig osäkra att undervisa om programmering, algoritmer och representationer, men att den osäkerheten har under projektets gång minskat. Många lärare berättade om mycket positiva erfarenheter särskilt vad det gäller elevernas intresse, motivation och interaktion. De tysta och blyga eleverna kom fram och deltog mera aktivt, d.v.s. arbetet med programmering hjälpte att utveckla sociala färdigheter. Forskare menar också att lärare kunde integrera programmering och lärande om programmering i andra ämnen. Det var enkelt att göra det i lågstadiet där en och samma lärare ansvarar för alla ämnen och kan därför på ett naturligt sätt koppla ihop arbetet i olika ämnen. Några lärare har exempelvis tillämpat sekvensering (*sequencing concepts*) i idrott genom att be elever att designa ett träningsprogram. I matematik användes Scratch för att lära om former och geometri och ScratchJr användes som ett redskap att utveckla skrivandet. Studien är intressant även för att den visade vilka typiska missuppfattningar som fanns bland lärare varav de flesta handlade om ämneskunskap som exempelvis vad olika begrepp betyder och varför ett visst begrepp ska läras ut. I sina slutsatser skriver författarna att med rätt stöd och kompetensutbildning lyckades de vanliga lärare som förut aldrig undervisade om programmering, algoritmer och datarepresentationer att förmedla kunskaper till barnen, att integrera delar i andra ämnen och att anpassa undervisning till olika elevers nivå.

Sentance och Csizmadia (2017) har gjort en studie i England om lärares strategier att lära ut datavetenskap (*computer science*) samt om vilka utmaningar de har stött på. Studien gjordes 2014 med syfte att vara vägledande för de lärare som skulle börja undervisa efter införandet av den nya läroplanen där fokus flyttades från IKT (informations- och kommunikationsteknologi) till datalogiskt tänkande och programmering. Lärare som deltog i

studien var de praktiserande lärare som sedan tidigare undervisar i ämnet och har därmed en viss erfarenhet. Forskare har delat upp lärares svar i tre kategorier: utmaningar relaterade till lärare, utmaningar relaterade till elever och utmaningar som har med resurser att göra. De mest återkommande teman var:

- lärares ämneskunskap
- elevernas begränsade förmåga att förstå innehållet
- tekniska utmaningar i skolan
- stor variation av nivåer i samma klass
- elevernas vilja och förmåga att lösa problem

De flesta lärare (ca.75%) som svarade på enkäten i Sentance och Csizmadias (2017) studie undervisar elever i åldrarna 11-16 eller 11-18. Fokus i den här uppsatsen ligger på elever i ålder 7-13 därför finns risken att de utmaningarna som identifierades i den engelska studien gäller främst undervisning i högre årskurs. Dessutom i England är datavetenskap ett eget ämne medan i Sverige infördes det som en del i andra ämnen i grundskolan. Det som dock är anmärkningsvärt är att till skillnad från Duncans m.fl. (2017) studie, rapporterar lärare i Sentance och Csizmadias (2017) studie att många elever saknar ”resilience”, d.v.s. förmåga att återhämta sig efter att ha gjort fel, att fortsätta och att inte ge upp. Dessutom skrivs det i rapporten (Sentance & Csizmadias, 2017) att de elever som har svårighet i matematik har sämre motivation att lära sig programmera. Vidare har många svårt att tänka datalogiskt, d.v.s. att dela upp problem i mindre delar och att kunna tillämpa de separata delarna på ett systematiskt sätt, exempelvis att använda både FOR och IF loopar i syfte att bygga en algoritm.

Däremot i alla tre ovan nämnda studierna sammanfaller de utmaningarna som själva lärare fick erfara. Många lärare kände oro och osäkerhet vad det gäller deras ämneskunskaper. Många lärare i Sentance och Csizmadias (2017) studie har ägnat sig åt att lära sig programmering på egen tid. Ytterligare en stor utmaning för dem är att hinna med alla elever som har så olika nivåer. Dessutom, uttryckte lärare behov av att hitta flera pedagogiska sätt att stödja elever samt behov av mera vägledning i bedömningen. Även problematiken att göra ämnet till något roligt och intressant återfanns bland svaren. Sist men inte minst, kände lärare en avsaknad av stöd från skolchefens sida med avseende på komplexiteten att undervisa i ämnet, men även att få hjälp från skolans IT avdelning med hårdvara och mjukvara har varit svårt.

Att skolans ledning har en stor roll menar även Heintz m.fl. (2017). Enligt författarna erbjöd Skolverket flera olika möjligheter att inhämta erfarenhet och kunskap om programmering innan den reviderade läroplanen skulle träda i kraft, exempelvis kurser om programmering i matematik och teknik, programmering som tvärvetenskapligt tillvägagångssätt, konferenser som handlade om hur lärare rent praktiskt skulle gå till väga och stöddokument. Den avgörande rollen får dock en skolas ledning, som måste se till att lärare har resurser i form av tid, material, åtkomst till nätverk, menar forskarna (Heintz m.fl., 2017). För att kunna agera på det sättet, måste ledningen ha förståelse för lärares behov och de utmaningar som

införandet av programmering innebär. Heintz m.fl. (2017) refererar till de svårigheter som England identifierade ett år efter att förändringarna i läroplanen gjordes. Det främsta hindret att anamma nya regler har varit lärares begränsade kunskaper i programmering samt deras låga självförtroende att kunna undervisa i det nya ämnet. Ett sätt att hjälpa lärare skulle vara att säkerställa en fortlöpande kompetensutveckling, lämpliga läromedel och stöd från andra yrkesutövande lärare.

Mannila (2017) menar även att lärarutbildningen bör anpassas till de nya kraven, för att det inte räcker med att få kunskap enbart om själva stoffet, utan det behövs att lära sig nya pedagogiska strategier och angreppssätt för att undervisa matematiken på.

Även Sentance och Csizmadia (2017) i sin studie underströk att det finns en pedagogisk förskjutning med avseende på hur ämnet ska undervisas och menade att just detta är en av dem viktigaste resultaten i studien. Tidigare inom ämnet datavetenskap i England låg fokus på IKT, d.v.s. det handlade om att lära sig en viss mjukvara, medan nu ligger vikten vid att lära sig tänka datalogiskt. Detta kräver helt andra sätt att undervisa. Enligt forskarna har dessa sätt kopplingar med en konstruktivistisk syn på lärande, bl.a. att elever behöver delta aktivt i lektionen. Lärande måste vara kopplat till autentiska och betydelsefulla situationer där elever lär sig genom att själva erfara en viss lärandesituation och sedan koppla den till det de redan vet. Lärande ska vara experimentellt och elever ska inte vara rädda att göra misstag när de exempelvis försöker att själva lösa problem.

Dostál, Wang, Brosch, Nuangchalerm och Steingartner (2018) gjorde en enkätstudie[4] för att ta reda på hur lärare i Tjeckien förhåller sig och hanterar nya krav i läroplanen, 87 lärare som undervisar i datavetenskap (computer science) deltog i studien. Enligt forskarna är lärare förändringsagenter. Det är de som implementerar förändringarna och därför påverkar deras attityder och förhållningssätt gentemot förändringen hur de kommer att agera: stödja och acceptera eller avvisa och ställa sig emot ändringarna. Dostal m.fl. (2018) menar att om reformen inte stämmer överens med lärares övertygelser, som i sin tur bygger på deras erfarenhet, så resulterar detta i en konflikt. Samma tanke återfinns i Rolandssons (2013a) artikel där han skriver att en utbildningsreform knappast kan lyckas om lärares övertygelser inte tas hänsyn till.

Men hjälp av en enkät kunde Dostal m.fl. (2018) identifiera tre grupper bland lärare:

- proaktiva innovatörer - de som aktivt och på egen hand engagerar sig i nya sätt att undervisa;
- reaktiva innovatörer - de som förstår vikten av att vara innovativ och kan genomföra förändringarna men kommer inte vis initiativet första; och
- opponenter - de som har negativ inställning mot förändringarna.

[4] I artikeln beskrivs enkätstudien på följande sätt: ‘a research survey implemented at 58 basic schools in the Czech Republic.’ (s. 1) - det är lite oklart vad ‘basic’ betyder, men sannolikt handlar det om grundskolan.

I Dostals m.fl. (2018) studie låg fokus på hur lärare förhöll sig till förändringen i läroplanen och detta kopplades till lärares ålder. Samtidigt lyfter författare fram även andra aspekter som viktiga vid förändring, exempelvis att ha en gedigen strategi som omfattar en kommunikationskampanj om förändringen, insatser i form av stödmaterial, manualer och kompetensutveckling för lärare. Detta i sin tur kräver förståelse för att lärare har en nyckelroll när förändringar i en läroplan ska implementeras.

Larke (2019), i sin artikel diskuterar hur förändringen av läroplanen gick till i England och vikten av lärares syn och övertygelser. Hon upptäckte vidare att många lärare såg innehållet i den nya läroplanen som mycket snävare men också vanskligare att undervisa i utan kompletterande stöd och kompetensutveckling. Dessutom, de förkunskaper som krävdes av elever fanns ofta inte med eller så var kraven högre än vad elever kunde uppnå under årets gång. Enligt forskaren, agerar en lärare som en *gatekeeper* och filtrerar de utbildningskraven som eleverna ställs inför genom egna erfarenheter och värderingar, på det sättet påverkas verkställandet av den nya läroplanen. I studien visar Larke (2019) att många lärare inte tyckte att de nya kraven gynnar elevernas lärande och valde ofta bort att undervisa enligt de nya riktlinjerna. Hon understryker att användningen av digital teknologi i undervisningen fortfarande var förvånansvärt lite utbrett med tanke på att den nya läroplanen implementerades för tre år sedan. Att lärare valde bort programmering var dock ett logiskt resultat av hur lärare bedömde situationen, d.v.s. valet formades av lärares misslyckade försök att tillämpa den nya läroplanen och synen på kravens innehåll var orealistiska.

4.3 Sammanfattning av forskningsöversikt

Tidigare forskning visar att matematik inrymmer abstrakta begrepp men med programmering kan dessa konkretiseras. Programmering sänker hindret som eleven kan uppleva i matematik och bidrar till att tänka logiskt för att lösa problem. Forskare hävdar också att programmering medför ökad kognitiv förmåga, samt utvecklar modelleringsfärdigheter som hjälper att hitta strategier för problemlösning i matematik. De menar vidare att datalogiskt tänkande är en viktig byggsten i både programmering och matematik för att lösa problem. Samtidigt har det framförts kritik gällande begreppets innebörd och dess starka koppling till datavetenskap.

Det finns fåtal studier gällande programmering i skolan. En av studierna visar att lärare har begränsade kunskaper om datavetenskap och programmering trots att lärarna tycker den är en viktig kunskap att utveckla för framtiden. En större studie berörde analoga aktiviteter, övningar i visuella miljöer och fysiska enheter som Bee-bot. Det visade engagemang från elever och en ökad motivation och intresse för programmeringen speciellt inom tillbakadragna elever. En annan studie synliggjorde också att om elever har svårt för matematik har de sämre motivation till att lära sig programmering. Ytterligare en studie kom fram till att en viktig aspekt som forskningen belyste var att skolledningen bör ge tid för lärare att utveckla kunskaper som möter de nya kraven som införs i läroplanen.

5 Teoretiskt ramverk - TPACK

TPACK är ett konceptuellt ramverk som utvecklades i början av 2000-talet av Mishra och Koehler (Parr, Bellis & Bulfin, 2013). Mishra och Koehler byggde på Shulmans modell PCK från 80-talet, som består av tre komponenter: PK - Pedagogical Knowledge, CK - Content Knowledge och PCK - Pedagogical Content Knowledge.

Shulmans PK innefattar kunskap om hur man lär ut och hur man lär sig, inklusive olika pedagogiska strategier, teorier och visioner. Modellens CK handlar om ämneskunskap och omfattar begrepp, kunskap om vilka delar av ett visst ämne kan läras ut samt kunskap om läroplanen (Vivian & Falkner, 2019). Detta är två skilda kunskapsområden som både är viktiga och behövs i lärande, de är sammanflätade och tillsammans formar en ny domän som heter PCK - Pedagogical Content Knowledge. PCK följaktligen innebär lärares kunskap att integrera ämneskunskaper med pedagogiska kunskaper på ett sätt som hjälper elever att lära sig ett visst ämne och till svenska skulle det kunna översättas med ordet ämnesdidaktik. Shulman också menade att utan PK, blir det inget PCK (Brantley-Dias & Ertmer, 2013). Med andra ord, integrationen mellan dessa två skilda områden - det pedagogiska och det ämnesmässiga - utgör kärnan i modellen (Pareto & Willermark, 2019). Mer konkret inbegriper PCK kunskapen om läroplanen och bedömning, samt vanliga till ämnet relaterade missuppfattningar och specifika tekniker att främja lärande i ett visst ämne (Kopcha, Ottenbreit-Leftwich, Jung & Baser, 2014).

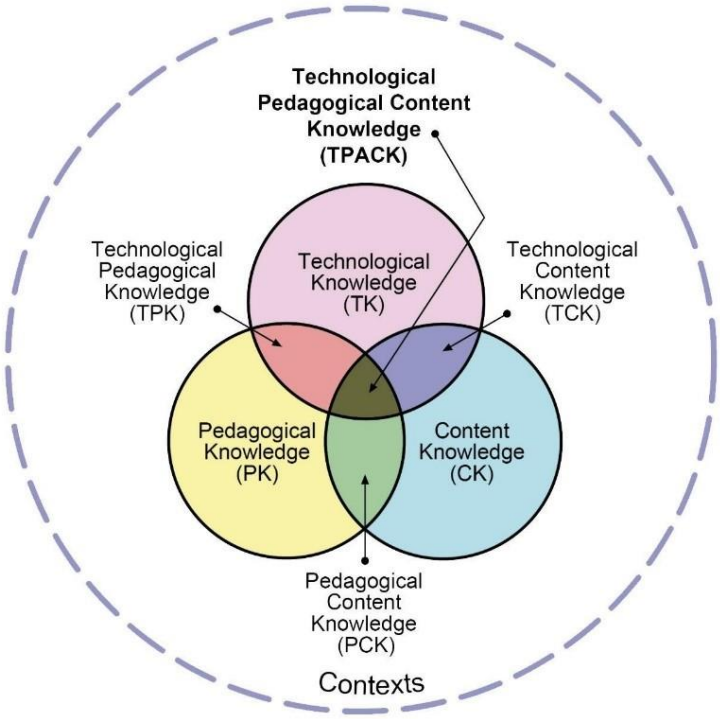
PCK-modellen var först och främst avsedd för forskare, lärarutbildningar och ansvariga för läroplanen och skulle illustrera den komplexa kunskapen som lärare besitter (Brantley-Dias & Ertmer, 2013). Den spreds vidare från USA till andra länder och har påverkat hur läroplan och kursplaner utformas i flera västerländska utbildningssystem över hela världen (Parr m.fl., 2013).

Till den befintliga PCK-modellen har Mishra och Koehler lagt till en ny aspekt, nämligen teknologi och utvidgade modellen till TPACK - Technological, Pedagogical And Content Knowledge. Det teknologiska beskriver kunskapen om hur teknologiska resurser ska användas och tillämpas (Pareto & Willermark, 2019) i lärande.

Modellen är direkt kopplad till undervisningssituation och beskriver områden som är viktiga i undervisning. Denna studie kan sägas i stora drag fokusera på programmering i matematikundervisning och programmering är nära förknippad med teknik. Eftersom TPACK-ramverket nämner teknik explicit blir modellen relevant att använda i studien.

Nedan följer beskrivning av modellen.

TPACK ramverk består av tre huvudkomponenter (teknik, pedagogik, ämnesinnehåll) som överlappar och skapar totalt sju stycken nya områden (se Figur 2).



Figur 2. TPACK-ramverket (Koehler & Mishra, 2009)

De mest viktiga områden skapas emellertid i skärningspunkterna där cirklarna möts (Willermark, 2018) och dessa är: TCK - teknisk ämneskunskap, PCK - pedagogisk ämneskunskap, TPK - teknisk pedagogisk kunskap samt centrum av ramverket TPACK som omfattar alla tre områden: teknisk pedagogisk ämneskunskap (TPACK). Ytterligare en komponent ligger utanför Venndiagrammet och utgör kontexten där lärande sker. Koehler och Mishra (2009) understrycker kontextens roll och vikt i undervisning. Konkreta exempel på vad kontext innebär skulle kunna vara 'årskurs, elevers bakgrund och tillgången till teknik' (Willermark, 2018).

Här följer definitioner på varje område som beskrivet av Koehler och Mishra (2009).

Content Knowledge (CK) - står för ämneskunskap. Vilken ämneskunskap en lärare behöver ha varierar beroende på ämnet eftersom olika ämnen bygger på olika idéer, teorier, sätt att undersöka det området, begrepp, m.m. Ämneskunskap ses som en viktig komponent inom TPACK modellen (Koehler & Mishra, 2009).

Pedagogical Knowledge (PK) - innebär kunskapen om hur man lär ut och hur man lär sig. Mer konkret innebär PK förståelse för olika tekniker att planera och bedriva undervisning, leda arbetet i klassrummet, bedöma elevers kunskap. En lärare som har djupa kunskaper i PK förstår processerna bakom hur elever skapar kunskaper och färdigheter.

Technological Knowledge (TK) - innebär kunskap om informationsteknologi som sträcker sig bortom den traditionella datorliteracy och omfattar bl.a. kunnande i att skilja på när teknologi möjliggör och när den hindrar att nå ett visst mål. Eftersom teknologi utvecklas snabbt, är TK i en ständig tillblivelse. Lärare bör ha en bred förståelse för

informationsteknologin för att tillämpa den produktivt i sin undervisning. Att kunna manövrera i denna utveckling och anpassa sig för att använda teknologi på ett produktivt sätt är också en del av TK.

Pedagogical Content Knowledge (PCK) - handlar om att lärare använder sina pedagogiska kunskaper för att lära ut ett visst innehåll. Läraren behöver finna olika sätt att presentera ämnet på och skräddarsyr och anpassar sina instruktioner till de olika förkunskaper som eleverna har och på det sättet transformerar ämnet efter behov. Här är förståelse för bl.a. de vanligt förekommande och till ämnet relaterade missuppfattningar viktig.

Technological Content Knowledge (TCK) - inbegriper förståelse för hur ämnesinnehåll och teknologi samverkar. Ofta handlar det om representationer. Enligt Koehler och Mishra (2009), blir ett visst ämne förändrat när teknologin används. Med andra ord, vilken teknologi väljs för att införliva och illustrera ett visst ämne och till ämnet relaterade idéer, kan bildligt talat å ena sidan förminska och begränsa ämnet eller å andra sida få ämnet att blomma ut. Denna samverka är ömsesidig, d.v.s. ämnesinnehåll påverkar valet av teknologi samtidigt som inte alla ämnen låter sig representeras med likadana teknologiska verktyg.

Technological Pedagogical Knowledge (TPK) - skulle kunna förklaras som en djupare förståelse av relationen mellan teknologi och lärande, men utifrån pedagogisk utgångspunkt. Här blir termen *affordanser* viktig. Lärarna behöver tillägna sig kunskap om teknologins men också kontextens affordanser, d.v.s. vilka möjligheter och begränsningar finns inbyggda i teknologi och kontexter, vilken handling en viss teknologi och en viss kontext framkallar. Detta gäller både elevernas agerande och lärares pedagogiska strategier.

Technological Pedagogical Content Knowledge (TPACK) - innefattar den kunskap som blir till när lärare använder alla de andra, ovan omnämnda kunskaper. Koehler och Mishra (2009) understryker att TPACK inte är samma som kunskap inom de tre huvudkomponenterna var för sig, utan TPACK inbegriper förståelse för hur de tre områden samspelar och påverkar varandra och kontext samt lärares kunnande att använda TPACK-kunskap för att hitta passande lösningar i undervisningssituation. Samtidigt är det viktigt att ha kunskap i varje område för ju mer kunskap inom dessa tre områden läraren skaffar sig desto större expertkunskap hen har för att kunna använda TPACK-kunskap i undervisningen.

TPACK har flera tillämpningsområden. I forskningen kan ramverket användas för att undersöka lärares kunskap om hur teknologi kan integreras i lärande (Koehler & Mishra, 2009) eller för att ”studera lärares undervisningspraktik” (Willermark, 2018, s.39). Modellen kan fungera som utgångspunkt i studier för blivande lärare (Koh & Divaharan, 2011).

TPACK modellen används mycket både i skolas och forskningssammanhang (Pareto & Willermark, 2019). Det finns mycket forskning om TPACK och om PCK där den senare ses nästan som ett paradigm eftersom den är vägledande för hur vi tänker om lärares undervisning (Vivian & Falkner, 2019). Modellen är dock komplex och har också fått kritik (Parr m.fl., 2013; Willermark, 2018). En återkommande kritik är att det saknas tydlighet i vad varje område innebär. Området teknik har kritiserats för sin vaghet, otydlighet och för att den

omfattar alla typer av verktyg. Mishra & Koehler (2006, i Willermark, 2018) menar dock att det är viktigt att behålla den breda definitionen så den inte förlorar sin relevans i den snabba tekniska utvecklingen.

Vidare kritik säger att diffusa gränser medför svårigheter att kunna tillämpa modellen praktiskt. Det kan exempelvis vara svårt att mäta och sen jämföra kunskaper inom enskilda domäner när det inte går att på ett entydigt sätt beskriva vad ett visst område innebär (Willermark, 2018; Pareto & Willermark, 2019). Ramverket ger inte heller några konkreta råd för hur lärare skulle kunna förbättra och utveckla sin undervisning med hjälp av TPACK. Även själva införandet av den tekniska domänen ifrågasätts. Shulman (1986, s.9, i Brantley-Dias & Ertmer, 2013) beskrev CK bl.a. som ”the knowledge of instructional materials that are useful for teaching a certain content including materials such as software, visual materials, and films, among others”. Brantley-Dias och Ertmer (2013) menar att hade Shulman skrivit sin artikel idag, är det inte helt fel att tänka att teknologin skulle räknas som en del av lärandematerial, d.v.s. som en del av CK. Den här åsikten håller dock bara så länge som man ser teknologi som jämställd med traditionella verktyg. Koehler and Mishra (2009) menade emellertid att teknologi är väldigt annorlunda och att dess införande komplicerar lärandet avsevärt. Så blir det på grund av att teknologin är ogenomskinlig, mångsidig och föränderlig, den kan ha olika funktioner och det är inte självklart hur den ska användas jämfört med en penna (Koehler & Mishra, 2009). En penna exempelvis är genomskinlig, d.v.s. den har tydliga egenskap för hur den ska användas. På grund av den komplexiteten som teknologin innefattar blir det för lärare en utmaning att använda teknologi i undervisningen. Teknologins användning påverkar emellertid undervisningspraktiker så mycket, att lärare är tvungna att tänka om vad de tre kunskapsaspekterna, d.v.s. teknik, pedagogik och ämnesinnehåll är (Mishra & Koehler, 2006, i Brantley-Dias & Ertmer, 2013). Med andra ord, med införandet av teknologi i lärande, den samlade kunskapen som lärare behöver ha har blivit ändå mera komplex och för att kunna beskriva den behövde PCK ramverket utvidgas, menar Mishra och Koehler (2006, i Brantley-Dias & Ertmer, 2013), varav tillkommande av ett nytt område ‘teknologi’ i Shulmans PCK-modellen.

Slutligen kritiserades även det att modellen inte närmare utvecklar vad kontexten betyder, fastän denna framhålls som väldigt viktig. Kontext-aspekten ofta saknas också i forskningen om TPACK (Pareto & Willermark, 2019).

Trots kritiken anser vi TPACK modellen vara användbar för den erbjuder en utgångspunkt att analysera en undervisningssituation. Den gör det genom att ‘framhålla interaktionen mellan teknik, pedagogik och ämnesinnehåll’ (Willermark 2018, s.46). De kunskapsdomäner som ingår i ramverket är aktuella även i vår studie och även om gränserna mellan dem inte är klara, erbjuder modellen ett sätt att närma sig den komplexa situationen i dagens matematikundervisning. En undervisningssituation innehar komplexitet eftersom det krävs av lärare att hen har kunskaper av olika karaktär och med hjälp av TPACK-modellen kan undervisningssituation brytas ner i mindre delar. I denna studie kommer ramverket användas för att synliggöra vilken roll programmering får i matematikundervisning. Tanken är att genom att fråga lärare om deras resonemang, uppfattningar och erfarenheter identifiera de

framträdande aspekterna som sedan kommer placeras inom TPACK-ramverket i syftet att se hur programmeringen används i matematikundervisning.

Ett konkret exempel på hur TPACK-modellen skulle kunna tillämpas är en matematiklektion där addition lärs ut med hjälp av programmering. Programmering skulle då kunna tolkas som teknologi som används i undervisning. Eller kanske som ett pedagogiskt redskap för att lära ut ett visst ämnesinnehåll. Eller är det kanske en del av ämnesinnehåll?

6 Metod

I detta kapitel presenterar vi studiens design med datainsamling och analys av empiriskt material (Tabell 2), samt tar upp forskningsetiska aspekter. I datainsamlingsdelen redogör vi för val av metod, urval av respondenter och genomförande av intervjuerna. I analysdelen redogör vi för arbetet med analys av det empiriska materialet.

Metoden vi har använt för datainsamling är semi-strukturerade intervjuer och det empiriska materialet består av intervjuer med åtta matematiklärare i grundskolans årskurser 2–6. Bearbetning och analys av det empiriska materialet består i att i ett förta steg kategorisera och koda intervjuer i NVivo-verktyget och i ett andra steg att samarbeta i Google-dokument med vidare kategorisering och analys av data.

Tabell 2. Forskningsdesign översikt

Empiriskt material	**Metod för datainsamling**	**Metod för analys av empiriskt material**
Intervjuer med åtta matematiklärare i grundskolans årskurser 2-6	Semi-strukturerade intervjuer	Induktiv analys med stöd av NVivo-verktyg och Google-dokument

6.1 Datainsamling

6.1.1 Val av metod

Studiens syfte styr val av datainsamlingsmetod (Repstad, 2007). Vårt syfte var att komma åt hur matematiklärare resonerar om programmeringskravet i läroplanen samt hur de arbetar med införande av detta i sin undervisning. Intervjun, enligt Repstad (2007), är en relevant metod för att komma åt och samla in personers åsikter. Cohen, Manion och Morrison (2017) menar att intervjuer gör det möjligt för oss som forskare att samla in data om hur matematiklärare uttrycker sig om och ser på programmeringskravet ur sin egen synvinkel. Författarna belyser att intervjumetoden används som det viktigaste sättet att samla information när det gäller att skaffa information om vad en person tänker, vet, gillar, värderar och tror. De menar vidare att intervjumetoden är flexibel och ger möjlighet till spontanitet och att vid behov ställa följdfrågor för att få djupare förståelse.

Den typ av intervjuer vi har använt är semi-strukturerade intervjuer med öppna frågor. Semi-strukturerade intervjuer är flexibla och intervjufrågorna kan utvecklas vid behov. Cohen m.fl. (2017) säger att det ämne och de frågor som ska behandlas skall specificeras i förväg i dispositionsform, sedan bestämmer intervjuaren sekvens och arbete med frågor under intervjun. Vi har formulerat intervjufrågorna i förväg, men vi bestämde oss för att vara följsamma genom att ställa följdfrågor under intervjuerna.

Eftersom en semi-strukturerad intervju kan resultera i väsentligt olika svar, minskar svarens jämförbarhet mellan respondenterna. Detta är dock ett något mindre problem i vår studie då vi är intresserade av variationer i lärares resonerande och sätt att arbeta, och inte tänker oss att generalisera till lärare i allmänhet.

6.1.2 Kritiska aspekter av metodvalet

Repstad (2007) belyser att intervjumetoden har kritiserats för att vara alltför idealistisk och individualiserad, d.v.s. att den fokuserar på enskilda personers åsikter och förbiser sociala och materiella strukturer och ramvillkor. Vår intervjumetod fokuserar också på matematiklärarnas uppfattningar och åsikter, som Repstad (2007) lyfter fram. Detta skulle kunna betyda att vi kunde missa andra viktiga kontextuella aspekter. Å andra sidan syftade studien till att undersöka matematiklärarnas åsikter, uppfattning och resonemang.

En annan kritisk aspekt är att en del intervjuer genomfördes med kollegor, som kan vara känsligt om frågorna innefattar personliga frågor. Sådana frågor förekommer inte i vår studie. Vi har dessutom delat ut information till respondenterna om vår studie och deras anonymitet enligt akademiska principer samt behandlingen av deras data. Se Bilagorna 2 och 3.

Enligt Repstad (2007) är en möjlig negativ aspekt, när det är två som intervjuar, att det är resurskrävande och att svarspersonen kan uppleva sig vara i minoritet. Men vi övervägde fördelarna med att genomföra intervjuarna tillsammans, för att kunna komplettera varandra och diskutera varje intervju efteråt. Detta lyfter även Repstad (2007) fram.

6.1.3 Intervjuguiden

Syftet med denna studie är att undersöka hur lärarna resonerar kring och arbetar med programmeringskravet i läroplanen för ämnet matematik. De mer specifika forskningsfrågorna är följande:

- Hur resonerar lärare kring det programmeringskrav som införts i matematikämnet?
- I vilken utsträckning ser lärare att matematikundervisningen kan stödjas av programmering?
- Hur har lärare infört programmering i matematikundervisningen?

Intervjuguiden som vi har skapat består av 18 frågor (Bilaga 1), och innehåller fem teman. Dessa teman är:

- Bakgrund
- Kunskaper om programmering och datalogiskt tänkande
- Programmeringskrav
- Programmering – matematikundervisning

- Programmering i den egna matematikundervisningen

Tre av intervjufrågorna är direkt kopplade till forskningsfrågorna och återfinns i de tre sista temana. De resterande frågorna i intervjuguiden försåg oss med faktakunskap, såsom vilken bakgrund lärarna har och hur de fick reda på programmeringskravets införande. Slutligen har vi frågor av mera allmän karaktär, som exempelvis, vilken är din personliga åsikt om ändringarna/de nya kraven? och hur har eleverna tagit emot programmeringsinslagen i undervisningen? Intervjufrågor som inte är direkt kopplade till forskningsfrågorna inkluderades i intervjuguiden med syfte att samla in kontextuella data. Det området som studien handlar om är relativt nytt och kontextuella data behövdes för att få en bättre insikt i den rådande situationen.

6.1.4 Urval och genomförande

I urvalet utgick vi från bekvämlighetsprincipen och tillgänglighet både geografiskt och tidsmässigt (Cohen m.fl., 2017). En sådan bekvämlighetsprincip var att en av oss är kollega i en skola där några lärare har deltagit i intervjuerna. Vi har även tagit kontakt med lärare i närliggande områden. Cohen m.fl. (2017) menar att sådana urval inte kan representera alla lärares åsikter och resonemang och därmed kan inte lärarnas yttranden i denna studie generaliseras till hela lärarkåren.

Cohen m.fl. (2017) rekommenderar att göra intervjuer med så många människor som behövs för den sökta informationen. Å andra sidan menar Repstad (2007) att insamlade data ska vara hanterbara. Det viktiga är att se till att de valda respondenterna kan förse oss forskare med information, d.v.s. att respondenterna måste vara insatta i det vi frågar om.

I vår studie deltar åtta lärare som är verksamma på låg- och mellanstadiet, fyra lågstadie- respektive fyra mellanstadielärare. Fyra lärare är kollegor till en av oss som genomförde studien: Tre lågstadielärare och en mellanstadielärare. Resterande är en lågstadielärare respektive tre mellanstadielärare som jobbar i två andra kommuner. Tre av de fyra mellanstadielärarna har någon form av datavetenskaplig bakgrund. Lågstadielärarna har 10–25 års erfarenhet i läraryrket. En av mellanstadielärarna är nyutexaminerad och resterande har 10–15 års erfarenhet som lärare. Ursprungligen hade vi tänkt inkludera även högstadielärare i vår studie, men dem tvingades vi utesluta av det enkla skälet att vi inte har lyckats få tag i några.

För att kunna komma i kontakt med matematiklärare skickade vi e-post till chef på respektive skola. En chef bör ha inblick i sin egen verksamhet och hen känner till sina medarbetare och kan välja kandidat som respondent (Repstad, 2007). Men vi fick ingen respons från cheferna. Vi bestämde oss då för att kontakta lärare, både genom att fråga personligen eller via e-post, och efterhöra om de kunde ställa upp på intervju.

De berörda lärarna har fått ett brev som består av presentation om oss själva och om vår forskningsstudie. Brevet innehöll även information om behandling och hantering av lärarnas data och svar. Vi har även skapat en samtyckesblankett om deltagandet i studien som lärarna har skrivit under (Bilagor 2 och 3).

Våra respondenter har fått information om intervjutiderna, att det skulle ta ungefär 30 minuter och att intervjuerna blev inspelade. Utifrån intervjuguiden har vi genomfört våra semi-strukturerade intervjuer med en lärare i taget. Vi var två som intervjuade lärarna där en av oss satt fysiskt med läraren i samma rum och den andra var uppkopplad genom Zoom-verktyget för videosamtal.

Vi har spelat in intervjuerna med hjälp av diktafon.

6.2 Analys av empiriskt material

I den här delen av metodkapitlet beskriver vi processen hur vi har analyserat det empiriska materialet. Resultatet av datainsamlingen är inspelade intervjuer med åtta grundskolelärare som jobbar i årskurserna 2-6. Tabell 3 ger en sammanfattning av det empiriska materialet.

Tabell 3. Sammanfattning av det empiriska materialet

Lärare i årkurs	Datum för intervju	Intervjulängd i minuter
3	30 mars 2020	52 min.
6	2 april 2020	33 min.
3	8 april 2020	38 min.
4	17april 2020	48 min.
3	22 april 2020	37 min.
5	29 april 2020	32 min.
4	4 maj 2020	36 min.
2	7 maj 2020	37 min.

Ljudfilerna överfördes till NVivo-verktyget på datorn. NVivo är ett datorprogram som används vid analys av kvalitativa data, exempelvis intervjuer, videomaterial, bilder, enkäter med öppna frågor. Detta är ett så kallat CAQDAS-program där CAQDAS står för *Computer-assisted qualitative data analysis software* (Cohen m.fl., 2017). NVivo användes för att lagra, organisera och göra en initial kodning av data. Som Cohen m.fl. (2017) påpekar är det inte mjukvaran som gör dataanalysen utan det är människor som gör det. En viss mjukvara kan användas som ett hjälpmedel för att strukturera och förbereda data för analys. Kodning i NVivo möjliggör att bryta ner hela intervjun till mindre enheter (exempelvis enskilda frågor) och att skapa länk till relevanta avsnitt i intervjuinspelningen. Detta gör det enkelt att återkomma till dessa i analysen.

Innan vi satte igång med kodningen diskuterade vi hur vi skulle gå till väga för att säkerställa att vi arbetade enligt samma principer. Vi valde att i en första fas börja med ett induktivt

tillvägagångssätt. Enligt Hsieh och Shannon (2005) används denna ansats när syftet är att beskriva ett fenomen och när kunskap om fenomenet är begränsat. En induktiv ansats innebär, i vår studie, att vi inte hade några förutbestämda kategorier. Vi valde att lyssna på respondenternas svar och fokuserade på vad de sa, hur de beskrev och definierade programmering och dennas koppling till matematik och hur de berättade om sina erfarenheter i klassrummet vad det gällde att införa programmering i matematikundervisning etc. Och när vi gjorde det tilldelade vi varje svar en viss kod.

En kod är en etikett som forskaren ansluter till en bit av innehåll för att beskriva och kategorisera den biten. Cohen m.fl. (2017) kallar detta för *open code*, öppenkod. I NVivo kodade vi genom att lyssna på respondenternas svarssekvens till den aktuella intervjufrågan eller till en följdfråga och gav den en etikett eller kod. Koden *AnalogProg* är ett exempel som betyder *analogprogrammering.* Denna kod tilldelades de intervjusekvenser där lärarna berättade hur de jobbade med analog programmering. Dessa sekvenser kunde återfinnas bland svaren på flera frågor, exempelvis:

- Hur arbetar du med programmeringen i din matematikundervisning?
- Vad är programmering för dig?

Samtidigt som vi gjorde denna initiala kodning och tilldelade beskrivande etiketter, grupperade vi koderna till en överordnad kategori. I detta steg beskrev den överordnade kategorin innehållet eller temat i en bredare mening. Denna process kallar Cohen m.fl. (2017) för *axial code,* axiellkod. Syftet med den typen av kategorisering och kodning är att komma fram till ett specifikt sammanhang, aktivitet eller ett förhållande som uppnås i ett tema eller innehåll. *Praxiskategorin* exempelvis kan innehålla flera koder, olika arbetssätt eller tillämpningar som lärare har definierat.

Efter att alla åtta intervjuerna var kodade, transkriberade vi ordagrant i NVivo de delarna som innehöll lärarnas resonemang om relationen mellan programmering och matematik samt deras erfarenheter i undervisningen. Vi har däremot inte transkriberat de sekvenser som handlar om faktafrågor, exempelvis frågor om vilken behörighet lärarna har, hur länge de har jobbat i skolan etc. Frågor som behandlar andra aspekter, exempelvis hur eleverna har tagit emot programmeringsinslagen i undervisningen eller vilka förändringar lärarna behöver göra i sin lektionsplanering har vi inte transkriberat, förutsatt att de inte innehöll de ovan nämna aspekterna. Men andra ord, har vi gjort en ordagrann transkribering av de sekvenser som vi bedömde vara relevanta i förhållande till studiens forskningsfrågor. De sekvenser som inte var det har vi istället gjort en kort sammanfattning av, också i NVivo. Detta steg skulle kunna beskrivas som ett deduktivt angreppssätt eftersom vi utgick ifrån studiens tre forskningsfrågor. Ett deduktivt tillvägagångssätt innebär att arbeta med i förväg definierade kategorier som sedan kopplas ihop med en text (Mayring, 2000). Både den induktiva och deduktiva ansatsen är viktiga för den kvalitativa analysen menar Mayring (2000). Transkriberingen skedde genom att lyssna på de valda delarna och vi delade upp arbetet med transkriberingen genom att bestämma vilka respondenter var och en skulle transkribera.

Den initiala kodningen av intervjuerna gjorde vi individuellt i NVivo på var sin dator, vilket innebär att i början var koderna olika. Vi har sedan diskuterat våra överordnade kategorier i NVivo och försökt slå ihop dem till gemensamma kategorier. Helt identiskt blev det dock inte, men den övergripande strukturen var ändå den samma. Nästa steg i analysen var att göra det Cohen m.fl. (2017) kallar för analytisk kodning, vilket skulle kunna förklaras med att forskaren tolkar data och sätter ett nytt namn utifrån sin tolkning. En sådan djupare analys går ett steg längre och innebär mera än en deskriptiv kod. Vi fann det svårt att göra detta i NVivo på grund av att vi saknade tillgång till varandras data och bestämde att gå över till att använda Google-dokument som en arbetsyta för den fortsatta analysen. Google-dokument är ett textredigeringsprogram i molnet. Det tillåter flera personer att jobba med samma dokument samtidigt och på det sättet möjliggör det samarbete. Denna kollektiva analysfas i Google-dokument innebär i denna studie en fördjupning samtidigt som den är en validering av våra tidigare separata analyser. Vi säkrar upp våra tolkningar och kategoriseringar. Bytet till Google-dokument innebar också att vi flyttade det transkriberade materialet från NVivo till Google-dokument. Enligt Cohen m.fl. (2017) möjliggör användande av NVivo bl.a. att räkna frekvenser, identifiera mönster, hierarkier och utveckla teorier. Att vi bytte till Google-dokument skulle kunna innebära att vi missade möjligheten att fullfölja detta, å andra sidan har inte det varit studiens syfte.

Vi förfinade och utvecklade kodningen vi påbörjat i Nvivo genom att gå från våra separata analyser till en gemensam analys. Vi tog en forskningsfråga i taget, läste och läste om den transkriberade texten samt lyssnade flera gånger på de sekvenser i det inspelade materialet, som enligt vår bedömning var relevanta. Vi använde koderna i NVivo för att söka efter flera relevanta sekvenser i det inspelade materialet. På det sättet försökte vi kategorisera svaren till varje forskningsfrågan. För att komma till gemensamma kategorier hade vi en kontinuerlig diskussion om innehållet. Eftersom vi bor på olika orter förde vi diskussionen via telefonsamtal och analyserade varje vald textsekvens tillsammans. Vi lyssnade och försökte förstå lärarnas resonemang för att komma åt och sätta ord på den relation mellan programmering och matematik som de försökte förmedla. Ett exempel av en sådan kategori är ”Analog programmering”, som visar ett av lärarnas arbetssätt. Med denna kategori bedömde vi att lärare menar att programmeringen är konkret och lär elever att följa en struktur för att lösa ett problem.

Vår intervjuguide bestod av 18 frågor. Analys av de valda intervjudelarna resulterade i totalt 12 huvudkategorier och några av dessa innehåller underkategorier. De 12 huvudkategorierna är uppdelade i tre grupper i enlighet med våra tre forskningsfrågor. Den första forskningsfrågan innehåller sex huvudkategorier, de två andra forskningsfrågorna innehåller 3 huvudkategorier var.

6.3 Forskningsetik

Vi har följt Vetenskapsrådets forskningsetiska principer som finns beskrivna i Vetenskapsrådets rapport God forskningssed (2017). Alla deltagare har fått information om studien i form av ett brev innan intervjun (Bilaga 1). I informationsbrevet beskrevs syftet med studien, varför just den personen har blivit kontaktad - för att delta i en intervju, hur data skulle behandlas och användas samt att deltagandet var frivilligt. Den berörda hade också rätt att när som helst avbryta utan att ange något skäl. Inför varje intervju tillfrågades deltagaren om hen hade några frågor eller funderingar. Slutligen skrev varje deltagare på samtyckesblanketten (Bilaga 2), där de instämde att de deltog frivilligt samt att video- och / eller ljudinspelningar kunde användas för denna studie. Därmed har kravet om ett informerat samtycke uppfyllts.

För att säkerställa deltagarnas konfidentialitet har deras namn blivit utbytta med slumpmässiga siffror mellan 1 och 8, d.v.s. siffrorna är inte knutna till den ordningen som intervjuerna gjordes i. I rapporten framgår det inte heller på vilka skolor eller orter intervjuerna gjordes.

7 Resultat

Syftet med studien har varit att undersöka hur lärare resonerar kring införandet av programmering i matematikundervisning samt hur de går till väga i praktiken. Syftet med det här kapitlet är att presentera och analysera datainsamlingen samt identifiera underliggande mönster i lärarnas svar.

Studien styrdes av tre forskningsfrågor:

- Hur resonerar lärare kring det programmeringskrav som införts i matematikämnet?
- I vilken utsträckning ser lärare att matematikundervisningen kan stödjas av programmering?
- Hur har lärare infört programmering i matematikundervisningen?

Med hjälp av en induktiv analys analyserades lärarnas yttranden kring de intervjufrågor som har en direkt koppling till studiens tre forskningsfrågor. De olika resonemang som lärarna förde kan i stora drag sägas forma två motpoler. Den ena representerar uppfattning att matematik och programmering har något gemensamt medan i den andra kan kopplingen mellan matematik och programmering sägas vara mindre explicit.

Dataanalysen har resulterat i totalt 12 huvudkategorier som redovisas i de fyra nästkommande avsnitten. Först ges det en sammanfattning av studiens resultat, sen framställs resultatet mera detaljerat. Resultaten kommer att presenteras i samma ordning som de tre forskningsfrågorna.

Det främsta syftet med analysen av intervjuer har varit att identifiera olika tankesätt. Vi har inte haft som mål att redogöra för exakt hur många lärare som står bakom varje kategori utan att undersöka vilka resonemang som finns. Vi har inte heller haft som syfte att göra skillnad mellan låg- och mellanstadiets lärare. Detta eftersom denna studie undersöker ett nytt kunskapsområde i läroplanen och syftet har varit att skaffa oss en övergripande bild över hur lärare resonerar och inte vilka lärare som säger vad.

I denna studie resonerade lärare på flera olika sätt samtidigt, uttryckte flera åsikter och i undervisningen blandade de flera tillvägagångssätt. Ett återkommande mönster i svaren var att det fanns en relation mellan kategorier och om lärarna har en kunskap i datavetenskap eller inte. De lärare som hade någon kunskap i datavetenskap kunde ge fler konkreta exemplen på hur programmering skulle kunna användas i matematik. Det gäller för tre lärare på mellanstadiet. Med utgångspunkt i detta kan man säga att lärarna på mellanstadiet var mer insatta i vad det innebär att använda programmering i matematikundervisning än lärarna på lågstadiet. Dock gäller detta endast i denna studie och någon generell slutsats kan inte dras. Men, för att skapa en lite mer nyanserad bild över vilka resonemang som var mer respektive mindre dominerande i vår studie, kommer andelen lärare i relation till forskningsfrågorna

redovisas i stora drag i avsnitt 7.1, men detta kommer inte att utvecklas mer detaljerat i analysen.

7.1 Sammanfattning av resultat

När det gäller den första forskningsfrågan, d.v.s. frågan kring införandet av programmeringskravet, uttryckte tre lärare att det fanns en relationell aspekt mellan matematik och programmering. De menade att matematik och programmering har något gemensamt samt att programmering kan bidra med viss nytta inom matematikämnet. De övriga fem lärarna sa att de fann det svårare att hitta relationen mellan programmering och matematik. De utgick i sina resonemang från exempelvis skrivningarna i läroplanen och läromedel. Lärarna som hörde till den första gruppen hade kunskap i datavetenskap, medan lärarna som hörde till den andra gruppen inte hade det, vilket kanske påverkade deras sätt att resonera. De olika resonemangen kring den första forskningsfrågan återges i sex huvudkategorier som redovisas i avsnitt 7.2.

Gällande den andra forskningsfrågan, som handlade om hur programmering kan stödja matematikundervisning, har alla lärare uttryckt förtroende för att programmering kan göra det. Samtidigt fanns det variationer gällande vilka aspekter, och hur många av olika aspekter som varje lärare lyfte fram. Fyra lärare, varav tre har kunskap i datavetenskap, kunde både beskriva konkreta sätt hur programmering kan stödja elever i matematikinlärning och reflektera mer abstrakt i denna fråga. Det verkar som att sådana nyanserade formuleringar var vanligare bland de lärare som har kunskap i datavetenskap. De resterande lärarna kunde också urskilja beröringspunkter mellan programmering och matematik, men resonemangen var mer allmänna. Följaktligen skulle det vara möjligt att säga att i denna studiens urval har hälften av lärarna kunnat påvisa hur rent konkret programmering kan stödja inlärning av ett visst område inom matematikämnet. Olika sätt att resonera kring denna forskningsfråga har delats upp i tre huvudkategorier som redogörs för i avsnitt 7.3.

Som svar på den tredje och sista forskningsfrågan, om hur lärare gick till väga med programmering i sin undervisning, berättade tre lärare att de använde läroboken i matematik som den främsta resursen i sin undervisning och gjorde uppgifter som fanns i läromedlet. De resterande lärarna använde resurser som låg utanför läromedlen eller båda typerna av resurser. De tre huvudkategorierna gällande den sista forskningsfrågan redogörs i avsnitt 7.4.

Nedan följer en mer detaljerad redovisning av de empiriska resultaten.

7.2 Hur resonerar lärare kring det programmeringskrav som införts i matematikämnet?

Data gällande den första forskningsfrågan *Hur resonerar lärare kring det programmeringskrav som införts i matematikämnet?* samlades in främst genom följande fråga i intervjuguiden:

Intervjufråga 3 d: Hur förstår du programmeringskravet i matematik?

Analysen resulterade i följande kategorier:

a) Programmering används för att nyttiggöra matematik
b) Matematik och programmering delar begrepp
c) Problemlösning finns i både programmering och matematik
d) Syftet med förändringen är att lära sig programmera
e) Programmering är resultat av digitalisering
    1) *Programmering likställs med digitala verktyg*
    2) Programmering är en del av digitala kunskaper
f) Programmeringskravet är innehållet i läromedel

Nedan presenteras kategorierna närmre med exempel.

7.2.1 Programmering används för att nyttiggöra matematik

Enligt ett av resonemangen, har programmeringen en praktisk funktion i matematik. I den här kategorin används programmering i syfte att uppnå en viss nytta, exempelvis rationalisera eller rutinisera matematiska beräkningar. Programmering används på grund av att den förenklar något i matematik. Det skulle kunna beskrivas som en pragmatisk syn på programmeringsanvändning i matematik.

Exempel 1:

> *Som jag förstår det så är det att man ska använda det [programmeringen] som ett hjälpmedel för att göra beräkningar. Det gäller väl att hitta användningsområden där man... eleverna ser att de faktiskt har nytta av. A men när nyttan kommer, någonting som ska göras många gånger, en sannolikhets simulering t.ex. eller något sånt. Sen så står det väl om algoritmer. [...] Och där är jag inte helt säker faktiskt på vad man är ute efter men jag skulle kunna tänka mig att man menar att det skulle... Det kan säkert vara väldigt mycket saker, men t.ex. att man gör ett program som kan räkna medelvärde, det är en typisk algoritm*. (Lärare 6)

7.2.2 Matematik och programmering delar liknande begrepp

Enligt detta resonemang har matematik och programmering liknande begrepp som funktioner, variabler och värden. Det finns alltså ett samband mellan programmering och matematik genom dessa gemensamma begrepp, vilket utgör motivet till införandet i läroplanen.

Exempel 1:

> *Jag minns att vi har diskuterat programmering ihop med matematik och det har varit ganska mycket att vi har pratat om det här med funktioner och så där. [...] Att man skickar in ett värde i en funktion och så kommer det ut någonting - och jämfört den med programmering. Flera såna där områdena inom matematiken som egentligen tar med programmering på olika sätt just för att det följer.. det är ett färdigt recept och så följer det här en funktion t.ex. då. Att man har variabler och man har konstanter som tillsammans då hjälper till att det händer någonting.* (Lärare 1)

7.2.3 Problemlösning finns i både programmering och matematik

I detta resonemang handlade det om att programmering kan lära eleverna strategier att lösa matematiska begrepp på. Att lösa problem är centralt både i programmering och i matematik och utgör den viktigaste beröringspunkten mellan dessa två områden. För att kunna göra det behövde eleverna lära sig en viss strategi. Med andra ord, användning av strategier innebar att eleverna utvecklade ett tankesätt för problemlösning.

Exempel 1:

> *Jag tror att deras [Regeringens] huvudpoäng är att man ska få det här problemlösningstänket egentligen. Att man kopplar algoritmer både liksom till en uppställning i matematik som till, a men en del av en kod och liksom att man lär sig - ju men tänka på ett visst sätt.* (Lärare 5)

Exempel 2:

> *Det är lite det jag sa innan att jag tycker att genom att kunna programmera så kan man problemlösning och problemlösning hjälper matematiken väldigt mycket. Att felsöka en kod det är ju nästan samma sak som felsöka en uppställning så... Eee, att man tar det lugnt och liksom i en viss ordning.* (Lärare 5)

7.2.4 Syftet med förändringen är att lära sig programmera

Vidare resonerades det om att ändringen i läroplanen betydde att eleverna skulle lära sig programmera. Detta resonemang skilde sig från de tre första på det sättet att någon förklaring till *varför* programmering infördes i matematikämnet inte utvecklades i någon större

utsträckning. Det framgick också att en del lärare inte kunde se ett samband. Programmering kan tolkas som om det är ett eget ämne med ny kunskap som eleverna behöver utbildas i. I den processen kan varje lärare bestämma sin egen programmeringsundervisning, exempelvis hur mycket tid hen ska ägna åt att lära ut programmering. Läraren såg alltså ett syfte med införandet, men inte i förhållande till matematikämnet, utan som ett eget inslag och en separat ämneskunskap som elever lär sig om. Det som är viktigt med det nya kravet är, med andra ord, att barnen skulle få lära sig programmera.

Exempel 1:

> *Men det är så ungefär jag har uppfattat det liksom. Du har ju algoritmer till mycket och så ska du skapa dem då via ett program, som då scratching. Så det är ju så jag tänkt.* (Lärare 8)

Exempel 2:

> *Men jag menar detta är ju en sån liten snäv sak i läroplanen så du kan ju liksom bara göra två sidor i matteboken så är det klart eller så kan du bygga ut det hur mycket som helst.* (Lärare 4)

7.2.5 Programmering är digital kompetens

Programmering likställs med digitala verktyg

Ett närliggande resonemang sätter den pågående digitaliseringen i fokus. Införandet av programmering tolkas enligt skrivningarna i läroplanen, där programmering ses som en del av de stora satsningarna gällande digitaliseringen inom skola. Det reflekterades inte hur programmering kan kopplas till matematik och speciellt inte för de yngre årskurserna på lågstadiet. Programmeringsinförandet tolkades som en nödvändighet att lära sig använda digitala verktyg och programmering sågs som en del av digitala verktyg eller digitala kunskaper.

Exempel 1:

> *Ja det är digitala verktyg och det gör vi, vi använder Ipads, många appar t.ex. Em.... Att undersöka problemställningar och matematiska begrepp, det är ju kanske inte för dem på lågstadiet, tror jag, om jag nu ska vara realistisk eller praktisk. Så det är ju, det gör vi, det här med digitala verktyg och att vi har med programmering och sånt, det har vi faktiskt börjat med på lågstadiet.* (Lärare 2)

Exempel 2:

> *Och sen det digitala, så att man ska få in det, försöka få in det i alla ämnen mer.* (Lärare 8)

Programmering ingår i det stora begreppet digitalisering i grundskolans nya läroplan och det är en ny kunskap som eleverna behöver utbildas i. Detta innebär att någon djupare koppling mellan programmering och matematik uteblir.

Exempel 3:

> *Jag tolkar det som att det finns eh.. en tanke från läroplanens författarna, eller ja... Regeringen egentligen. Att man vill att man ska ha digitala kunskaper.* (Lärare 7)

7.2.6 Programmeringskravet är innehållet i läromedel

I det sista resonemanget besvarades frågan om varför programmeringskravet infördes i läroplanen. Lärarna hade förstått varför programmeringskravet infördes genom att läromedel som användes i skolan behandlade programmering. Lärarna sa att de visste om förändringen, detta hade diskuterats på arbetsplatsen men de hade inte förstått innebörden. Det verkade som att läromedlet gav stöd både gällande till innehållet och pedagogik, d.v.s. stöd i vad som skulle läras ut. Detta reflekterar ett instrumentellt förhållningssätt till programmeringskravet och mindre av en reflektion över kravets innebörd.

Exempel 1:

> *Jag vet att vi läste liksom kursplanen och så kände jag mig orolig och vet att många andra tänkte Wow gud vad innebär det här egentligen, men sen när vi har pratat om det, man har pratat om den där kodboken och vi har tittat i våra matteböcker och då tänkte vi Jaja det här fixar vi. Och jag vet att vi diskuterade någon gång vilken nivå ska man lägga sig på och då så kändes det ändå som men det här fixar vi.* (Lärare 4).

Exempel 2:

> *Jag använder mig av läromedlen och tittar, för att då tänker jag att de har liksom ja.. förstått bättre än vad jag förstått vad det innebär. Så att då... det har blivit liksom att jag ser i matteboken att det är det här och det här [vi ska göra], liksom på det sättet.* (Lärare 3)

7.2.7 Sammanfattning och kommentar

I denna del sammanfattar vi vad lärare säger gällande forskningsfrågan: *Hur lärare resonerar kring programmeringskravet som införts i matematikämnet*.

En del lärare resonerade om att programmering och matematik har något gemensamt. De menade att både matematik och programmering använder samma strategi, begrepp och tankesätt för problemlösning. Vidare, med hjälp av programmering blir det möjligt att

rutinisera och effektivisera matematiska beräkningar. Med andra ord, ansåg lärarna att det fanns en relation mellan matematik och programmering.

En del lärare gjorde inte någon koppling mellan programmering och matematik. En lärare menade, exempelvis, att det viktiga var att lära sig programmera. Samma lärare tyckte att programmeringskravet är otydligt och att lärare själva därför får bestämma hur det ska tolkas och tillämpas. En annan lärare tolkade införandet av programmering som användning av digitala verktyg i undervisningen. Det nya kravet innebar för ett par lärare att de använde läromedlet som stöd och undervisade om det innehåll som fanns i läromedlet. Det kan sägas att dessa lärare uppvisade ett instrumentellt förhållningssätt gentemot programmeringskravet.

7.3 I vilken utsträckning ser lärare att matematikundervisningen kan stödjas av programmering?

Här kommer det att redovisas resultat av den andra forskningsfrågan *I vilken utsträckning ser lärare att matematikundervisningen kan stödjas av programmering?*

För att besvara forskningsfrågan har vi ställt nedanstående fråga:
Intervjufråga 4 a: Hur kan programmering stödja elevernas utveckling i matematik?

Sammantaget identifierades tre huvudkategorier:

a) Programmering och matematik använder samma arbetssätt
b) Programmering som hjälpmedel i undervisning
    1) *Programmering är kontext för matematik*
    2) *Programmering för att konkretisera matematik*
c) Programmering som hjälpmedel för elever
    1) *Programmering har en instruerande funktion*
    2) *Programmering bygger upp självförtroende*

Kategorierna med exempel presenteras nedan.

7.3.1 Programmering och matematik använder samma arbetssätt

Den mest återkommande iakttagelsen i frågan om hur programmering kan stödja matematik har varit att när elever gjorde analoga övningar så blev de tvungna att jobba på ett visst sätt. Lärarna påpekade att inom programmering behövde man jobba systematiskt och strukturerat. Samma gällde när man skulle lösa en matematisk uppgift. Det betydde att inom båda områdena jobbade man på samma sätt - strukturerat och systematisk då man gjorde saker stegvis och i en viss ordning. Att kunna jobba på detta sätt beskrevs av lärare ofta som en vinst och som något som kunde vara användbart både i matematik och i andra ämnen. Detta

skulle kunna översättas till att genom programmering kan generella förmågor såsom att förstå och följa instruktioner utvecklas.

Exempel 1:

> *...man måste ju med programmeringen börja på ett visst ställe och sen kommer nästa och sen kommer nästa och det är ju samma med ett matematiskt problem.* (Lärare 3)

Exempel 2:

> *Nej men det var väl det jag sa. Jag tänker att man utvecklar det här med att ta det i rätt ordning, det tänker jag att man utvecklar med programmering och det gör man nytta hela livet ju.* (Lärare 3)

Exempel 3:

> *Just det här att man tänker att man bygger ut att det ska kopplas ihop med det här sekvenstänkandet asså problemlösning. [...] Asså att kunna läsa en skriftlig instruktion och tänka i många steg hur ska jag komma fram till en lösning är ju många gånger det som är det allra svåraste i matematik, tycker jag för mina barn. De kan vara jätteduktiga på att liksom räkna rena tabeller eller vad man ska säga men när du sätter in en läskontext så blir det jättejobbigt. Och man hoppas ju att det här med liksom, att kunna följa en struktur att följa en ordning att göra en instruktion med många led att det på något sätt ska kunna gå hand i hand, så.* (Lärare 4)

Exempel 4:

> *Det är lite det jag sa innan att jag tycker att genom att kunna programmera så kan man problemlösning och problemlösning hjälper matematiken väldigt mycket. Att felsöka en kod det är ju nästan samma sak som felsöka en uppställning så... Eee, att man tar det lugnt och liksom i en viss ordning.* (Lärare 5)

Exempel 5:

> *Programmering är ju för mig att läsa instruktioner, förstå instruktioner, och följa dem på ett korrekt sätt så man får resultatet man är ute efter. Så är det för mig. Och det är det som lockar mig. Det hjälper eleverna på annat håll, exempelvis i svenska, i matte, i alla andra ämnen, att kunna läsa instruktioner, förstå instruktioner och till sist följa dem.* (Lärare 2)

7.3.2 Programmering som hjälpmedel i undervisningen

Det finns ett annat sätt att reflektera över beröringspunkter mellan programmering och matematikundervisning där programmering tolkas som ett redskap i matematikundervisningen, som ett av verktygen i lärarens pedagogiska verktygslåda som kan uppfylla olika funktioner.

Programmering är kontext för matematik

Några lärare berättade att när barnen programmerade i olika programmeringsmiljöer blev det nödvändigt att lära sig några matematiska begrepp för att komma vidare. Detta skulle kunna ses som att programmeringsmiljöer, exempelvis visuella sådana kan fungera som kontext för att lära sig matematik. Med andra ord, programmering skapar en situation som skapar ett behov av att använda matematiska kunskaper.

Exempel 1:

> *Man skulle kunna ju använda det lite som ett lockbete. Det här är någonting roligt och spännande och att... Ju men som jag använder Scratch för att lära dem koordinatsystem så att man inte står där med massa rutor på en tavla och bara Här har vi X3 och Y5, det blir liksom inte så roligt. Men om man kopplar det till ett spel istället så har de ett syfte av att lära sig koordinater. Och just ett spel det är någonting de stöter på hela tiden så det gör ju att det blir mycket närmare för dem i deras vardag.* (Lärare 5)

Exempel 2:

> *Vissa saker stöter man på men jag har inte haft någon tanke med det, det har jag inte haft. [...] Men jag har hittat en sida som heter code.org som har ett verktyg som ser ut rätt mycket som scratch, de har färdiga scenarier, man fyller i saker, det finns t.ex. ett med de här figurerna från Frost, filmen Frost. Elsa och [hennes kompisar] ska åka skridskor och då ska man fylla i att de ska åka framåt, svänga höger och då kommer in vinklar t.ex. och grader. [...] Då plötsligt för att lösa det och barnen tycker att det är kul och vill lösa uppgiften och då inser de plötsligt: Okey nu kommer det något nytt här, någonting med grader, men då måste vi lära oss det snabbt så att vi kan komma vidare med Elsas skridskoåkning. Det kom in en hel del såna saker det gjorde det men inte med någon större tanke från mig, utan det bara blev en bra bonus.* (Lärare 6)

Programmering för att konkretisera matematik

Att programmering kan konkretisera det abstrakta inom matematik var ett annat påstående. Det menades att med hjälp av programmering går det att visa att matematiken är logisk och konkret. Genom praktiska övningar skulle eleverna kunna få ökad förståelse för vad matematiken är vilket skulle kunna tolkas som att det gör matematiken till något meningsfullt och användbart.

Exempel:

Men jag tror det kan vara ett sätt att få en större förståelse för det här med matematik. Att det finns tydliga regler och att det finns en logik i matematiken. Att ju mer man får in av praktiska moment desto lättare blir det att också kunna konkretisera det här som egentligen bara är siffror som flimrar förbi att det kan också bli något verkligt av det. [...] Matematiken kan vara ett steg in i programmering och programmering kan vara ett steg i matematiken. (Lärare 1)

7.3.3 Programmering som hjälpmedel för elever

Programmering kan fungera som ett hjälpmedel för barnen i deras lärande.

Programmering har en instruerande funktion

Lärare menade att vid programmering fick eleverna omedelbar feedback av programmet, vilket barnen uppskattade. Barnen kunde testa något nytt och göra ändringar på en gång. Den omedelbara feedbacken skulle kunna liknas med en instruktion som eleverna tog till sig.

Exempel:

Så det blir ju som någon slags självrättande läromedel när man programmerar för man får ju hela tiden svaren. Facit kommer ju i form av att det blir en produkt av det man gör. Till skillnad från att man sitter och räknar uppgifter och sen kommer svaren mycket senare. Jag tror att det är ganska uppskattad. (Lärare 1)

Programmering bygger upp självförtroende

En del lärare sa att de flesta barnen var intresserade av datorer och att de hade lätt för att lära sig grunderna i programmering. Lärarna framförde också att arbeta stegvis fanns med implicit när man jobbade med programmering. Att arbeta på detta sätt utvecklade kunskaper hos elever som gav dem självförtroende och i sin tur ledde det till att de visade större intresse för matematik.

Exempel:

Men det logiska tänkandet, att många faktiskt har lite lättare för programmering så då bygger det ju självförtroende i matematik. Och jag brukar alltid förklara att matematik bygger på varandra, man måste kunna första delen för att kunna bygga på andra och tredje. I programmering blir det väldigt tydligt att har man med sig det första, och förstår vad som händer när man programmerar en viss del, då kan man fortsätta till nästa. Det är ingen idé att hoppa på det här sista steget och tro att nu ska jag programmera något jättestort eller göra ett jättestort spel eller sådär utan

man tar det stegvis. Och så funkar det ju matten, hela tiden. Så jag tror självförtroende och logiskt tänkande. (Lärare 7)

7.3.4 Sammanfattning och kommentar

Lärarna lyfte flera aspekter om programmering som skulle kunna kopplas till matematik. En av aspekterna som sågs vara relevant även i andra ämnen handlar om arbetssätt, d.v.s. att kunna arbeta strukturerat och systematiskt.

Programmering kunde även hjälpa till att förstå abstrakta matematiska begrepp, med andra ord - programmering konkretiserade och begripliggjorde matematiken. Ett ytterligare påstående var att programmeringsmiljöer användes för att införa och förklara matematiska begrepp. Dessa aspekter skulle kunna sägas bygga på synen att matematik och programmering har något gemensamt på en begreppslig nivå, att det är två ämnen som ligger nära varandra.

Vidare påstods det att programmering har en instruerande egenskap i form av omedelbar feedback. Slutligen menade några lärare att programmering kunde användas för att bygga upp självförtroende.

Det är intressant att, för en del lärare framstod programmering och matematik som besläktade med varandra och för en del andra lärare var det helt skilda områden. Det framstår också som att de som arbetat med programmering oftare i sin undervisning omfattade den första åsikten och de som arbetat med programmering mer sällan i sin undervisning tolkade det som två olika ämnen.

Den aspekt som framstod särskilt mycket bland de nämnda ovan handlade om att programmering konkretiserade och begripliggjorde matematik. Det verkar som att denna aspekt hade en direkt koppling till matematik medan resterande var av mera allmän karaktär.

7.4 Hur har lärare infört programmering i matematikundervisningen?

I denna del kommer det redovisas resultat för den tredje och sista forskningsfrågan: *Hur har lärare infört programmering i matematikundervisningen?*

Datainsamlingen har gjorts genom följande intervjufråga:
Intervjufråga 5a: Hur arbetar du med programmering i din matematikundervisning? Alt. Hur tänker du arbeta med programmering i matematikundervisningen (ge konkreta exempel, visa)?

Svaren på frågorna som berörde den tredje forskningsfrågan visade att lärare hade framförallt tre utgångspunkter när de jobbade med programmering. Den första handlade om att bekanta

sig med ett nytt arbetssätt och ett nytt sätt att tänka på, d.v.s. fokus låg på att lära sig följa exakta instruktioner och förstå att ordningen var väldigt viktig. Då användes det analoga övningar eller enkla övningar med robotar. En annan utgångspunkt var att öva på att programmera och detta gjordes i digitala programmeringsmiljöer, exempelvis Scratch. Här kunde inslagen av matematik förekomma och detta kunde ske både avsiktligt och oavsiktligt. När lärarna arbetade med analoga övningar och med programmering i visuella programmeringsmiljöer, använde de ofta externa lärresurser, exempelvis TV-program, Youtube-kanaler, appar och liknande. Den tredje utgångspunkten var att följa läromedel och göra de övningar som fanns i läromedlet.

Följaktligen sammanställdes analysen av data gällande den tredje forskningsfrågan i dessa kategorier:

a) Analog programmering
b) Programmering i visuella programmeringsmiljöer
c) Programmering i läromedel

Nedan följer en mera detaljerad förklaring med exempel för vad varje kategori innebär.

7.4.1 Analog programmering

Alla lärare har använt analog programmering i sin undervisning. Inledningsvis handlade det vanligen om att lära sig följa instruktioner. En lärare poängterade att det fanns tydliga regler som skulle följas. En annan lärare menade att det handlade om att konkretisera programmering. Detta skulle kunna ses som att för en del lärare var analog programmering till för att lära sig att jobba strukturerat, följa instruktioner och för en del lärare var analog programmering ett sätt att komma in i vad programmering är. Analog programmering verkade vara kopplad till resonemanget att programmering och matematik använder samma arbetssätt.

Exempel 1:

> *Man har en liten robot som ska styras och ska gå in i affären och man ska ge honom instruktioner med pilar. Vart hamnar han om man följer de här pilarna och så här åt det här hållet.* (Lärare 3)

Exempel 2:

> *A men det är ficklampor, ettor och nollor, man kan pärla, se mönster, asså den typen. Det tror jag är jättebra på lågstadiet förutom de här små robotarna men att man verkligen går in i vad det handlar om liksom. Så det tycker jag är väldigt bra att börja med analoga. [...] Att de liksom inte*

direkt så där hoppar på bara för att det är kul utan att man förstår grunderna. (Lärare 7)

Exempel 3:

...innan vi gjorde det [programmering i Scratch] så fick de själva göra praktiska övningar som inte var på dator. Då fick de ju göra a, koda en dans två och två. Två steg åt höger, två fram, tre ditt för att det då aa.. [...] Jag kände att det blev ju mer konkret när de först fick göra det utan dator för att förstå varför vi gör det för annars så blir det bara en rolig grej på dator. De förstår liksom inte varför de gör det. Men när du gör det så konkret, då när du ska göra en dans två och två ihop så blev det väldigt tydligt att det är ju väldigt tydliga regler hur du ska göra. Du måste ju följa instruktioner för att det ska bli rätt. (Lärare 8)

7.4.2 Programmering i visuella programmeringsmiljöer

Efter att ha jobbat med analog programmering gick lärarna vidare till att undervisa i att programmera i visuella programmeringsmiljöer. I visuella miljöer hade några lärare infört matematiska begrepp, exempelvis X- och Y-koordinatsystem och vinklar. Det skulle kunna ses som ett sätt att praktisera matematik. Det handlade om följande resonemang: matematik och programmering delar samma begrepp; syftet med programmeringsinförandet var att lära sig programmera; programmering är kontext för matematik; införandet av programmering innebar användandet av digitala verktyg. Alla dessa sätt att resonera kring matematik och programmering manifesterade sig genom undervisning i programmering inom visuella programmeringsmiljöer.

Exempel 1:

Man skulle kunna ju använda det lite som ett lockbete. Det här är någonting roligt och spännande och att... Ju men som jag använder Scratch för att lära dem koordinatsystem så att man inte står där med massa rutor på en tavla och bara Här har vi X3 och Y5, det blir liksom inte så roligt. Men om man kopplar det till ett spel istället så har de ett syfte av att lära sig koordinater. Och just ett spel det är någonting de stöter på hela tiden så det gör ju att det blir mycket närmare för dem i deras vardag. (Lärare 5)

Exempel 2:

Men jag har hittat en sida som heter code.org som har ett verktyg som ser ut rätt mycket som scratch, de har färdiga scenarier, man fyller i saker, det finns t.ex. ett med de här figurerna från Frost, filmen Frost. Elsa och [hennes kompisar] ska åka skridskor och då ska man fylla i att de ska åka framåt, svänga höger och då kommer in vinklar t.ex. och grader. [...] Då plötsligt för att lösa det och barnen tycker att det är kul och vill lösa uppgiften och då inser de plötsligt: Okey nu kommer det något nytt här,

någonting med grader, men då måste vi lära oss det snabbt så att vi kan komma vidare med Elsas skridskoåkning. Det kom in en hel del såna saker, det gjorde det, men inte med någon större tanke från mig, utan det bara blev en bra bonus. (Lärare 6)

7.4.3 Programmering i läromedel

I sin undervisning hade alla lärare behövt avgränsa programmering som ett eget område, d.v.s. det främsta syftet hade varit att lära ut programmering. Det skulle kunna betyda att lärare såg programmering som ett eget ämne med eget ämnesinnehåll. Som nämnts ovan, hade en del lärare använt flera olika resurser i sin undervisning. Men några lärare hade begränsat sitt material till matematikboken. Att undervisa enligt innehållet i läromedlet innebar också att endast det innehållet hade använts, d.v.s. lärarna kompletterade inte med externa resurser såsom webbsidor, appar eller annat eget sökt material. Jämförs undervisningspraktik med olika sätt att resonera på, kunde arbetet enligt läromedel kopplas till tolkningen av programmeringskravet där kravet förstås genom innehållet i läromedel.

Exempel 1:

Eftersom den faktiskt finns i t.ex. matteboken och det är ett eget kapitel i matteboken så är det väldigt vanligt att ta upp ett nytt kapitel, ett nytt område och gå igenom. Men innan vi har fått det här läromedlet så brukade jag ta upp det som ett eget område, exempelvis sa jag: Nu kommer vi att göra något spännande, asså man introducerar det som ett eget område, man knyter inte till något ämne. (Lärare 2)

Exempel 2:

Ja det är klart, all matematik är ju... bygger ju på logiskt tänkande, att man ska kunna ta det i rätt ordning och så. Men det känns ändå som att det [programmeringen] är en liksom ett eget område. Asså än så länge i alla fall. Det är inte så liksom integrerat i matteundervisningen hela tiden. Utan det blir precis som jag sa att den här veckan nu ska vi jobba med programmering här. (Lärare 3)

7.4.4 Sammanfattning och kommentar

Lärarna berättade hur de använde programmering i matematik. Enligt deras beskrivning fann vi att de tog upp programmering i två steg. Det första steget var att använda analog programmering. Genom så kallade analoga övningar tränade barnen på att följa mönster, styra varandra, styra tecknade figurer i matteboken. Det andra steget i det didaktiska upplägget var att eleverna arbetade i den visuella miljön, där en lärare exempelvis införde matematiska begrepp såsom koordinatsystem och vinklar.

Resultatet visade att programmering i första hand undervisades om i syfte att lära sig programmera och att lärarna använde programmering som ett eget ämnesinnehåll i undervisning. De flesta lärare använde flera olika lärresurser i sin undervisning. Det fanns också några få som berättade att matematikboken innehöll ett eget kapitel om programmering och det blir naturligt att ta upp det som ett nytt område, med andra ord, de använde matematikboken som lärresurs.

Dessa olika sätt att arbeta på kunde kopplas till några, men inte alla, sätt att resonera kring kravets införande, samt kring hur programmering kunde stödjas i matematikundervisning, d.v.s. till de två första forskningsfrågorna. Det tillvägagångssätt som täckte störst andel av olika sätt att resonera på var att programmera i visuella miljöer. Att undervisa i analog programmering kan kopplas med resonemanget att programmering och matematik använde samma arbetssätt. Att undervisa med hjälp av läromedlet kan kopplas med resonemanget att programmeringskravet var innehållet i läromedlet.

I resonemanget om kravets införande fanns det emellertid reflektioner som belyste att programmering kunde nyttiggöra matematik, samt att programmering utvecklade ett tankesätt som hjälpte till vid problemlösning. Dessa aspekter framgick dock inte i lärares praktiker. Reflektionen att programmering kunde fungera som hjälpmedel i matematikundervisning (d.v.s. elever fick omedelbar feedback och kunde anpassa sina handlingar) verkade inte heller ha funnit uttryck i undervisningen. Med andra ord, hade det framförts ett antal resonemang kring kopplingen mellan programmering och matematik, men inte alla har tagit fäste i praktiken. Den direkta kopplingen mellan programmering och matematik verkade vara mest framträdande i arbetet inom visuella programmeringsmiljöer.

För att sammanfatta delen Resultat i relation till studiens forskningsfrågor, kan det sägas att det finns en skillnad mellan lärares resonemang om kravets införande samt deras uppfattningar gällande hur programmering kan stödja matematik å ena sidan, och lärares praktiker å andra sidan. Resonemang och uppfattningar var mer varierande och belyste fler aspekter av relationen mellan matematik och programmering än vad undervisningspraktiker gjorde. En möjlig förklaring till varför lärarna i denna studie resonerade på många olika sätt skulle kunna vara lärarnas varierande erfarenheter inom ämnet datavetenskap: Några lärare var väl bekanta med detta område och för några vad det något helt nytt. Samtidigt, att undervisning bedrevs på ganska likartat sätt (men till varierande utsträckning) skulle kunna tolkas som att ha bakgrund i ämnet datavetenskap inte räckte till och att det fanns andra faktorer som påverkade hur lärare undervisade. Hur detta skulle kunna kopplas till tidigare forskning, kursplanen i matematik och teoretiskt ramverk TPACK presenteras i nästa del Diskussion.

8 Diskussion

I det här kapitlet övergår vi från att tolka data, d.v.s lärarnas svar, till att sätta kategorierna i relation till det teoretiska sammanhanget (Repstad 2007). Studiens resultat kopplas till följande diskussionspunkter: Kursplanen för ämnet matematik, det teoretiska ramverket TPACK och tidigare forskning. Slutligen berättar vi om studiens validitet.

Syftet med denna studie har varit att undersöka hur lärare i grundskolan resonerar kring och arbetar med programmeringskravet i läroplanen för ämnet matematik. Studien styrdes av tre forskningsfrågor:

- Hur resonerar lärare kring det programmeringskrav som införts i matematikämnet?
- I vilken utsträckning ser lärare att matematikundervisningen kan stödjas av programmering?
- Hur har lärare infört programmering i matematikundervisningen?

8.1.1 Kategorier i relation till kursplanen

I kategorierna som identifierades vid analysen i relation till kursplanen för ämnet matematik, framgår det att det finns gemensamma beröringspunkter. I kursplanens syfte beskrivs programmering som ett verktyg för att undervisa i ämnet matematik. Det står vidare att med hjälp av programmeringen ska eleverna ‘kunna undersöka problemställningar och matematiska begrepp, göra beräkningar [och] presentera och tolka data’ (Skolverket, 2019, s.54). Liknande resonemang har också gjorts av lärare då de beskrev att med hjälp av programmering skulle det vara möjligt att nyttiggöra matematik. De menade också att både matematik och programmering handlar om problemlösning samt att programmering är kontext för matematik. Ytterligare iakttagelser från lärarnas sida har varit att genom programmering kan matematik konkretiseras. Dessa formuleringar skulle kunna beskrivas som breda och omfattande, likaså är syftet i kursplanen brett och omfattande.

Den bredden och vidden som definierades i stycket ovan står i kontrast till texten i centralt innehåll i kursplanen. Under det centrala innehållet i matematikkursplanen beskrivs programmering som stegvisa instruktioner och algoritmer som ska användas i visuella programmeringsmiljöer. Detta skulle kunna tolkas som ämnesinnehåll fast i ämnet datavetenskap. Studiens empiriska material visar på att det stämmer överens med hur programmering undervisas i grundskolan - programmering undervisas som ett eget inslag och undervisningen handlar om att lära sig om programmering samt att programmera. Både det centrala innehållet och lärare i sin undervisning tolkar programmeringskravet på ett snävare sätt i förhållande till syftet i kursplanen. Detta utgör den andra beröringspunkten mellan lärares resonemang och formuleringar i kursplanen.

Det går att säga att läroplanen är otydligt skriven och till en viss del kan den ses som motsägelsefull. Det som står i syftet kontrasterar med det som står i centralt innehåll. Det

framgår inte hur eleverna kommer att nå syftet som beskrivs i läroplanen genom att följa instruktionerna i det centrala innehållet. Möjligen menade läroplanens författare att en progression bör ske i lärandet med start i å.k. 1 och fram till å.k. 9, d.v.s. en progression genom låg-, mellan- och högstadiet. Det saknas dock etablerade riktlinjer för hur lärare ska gå tillväga i sin undervisning för att kunna genomföra denna progression. En viktigt roll här kan tänkas vara avsaknaden av kunskapskrav gällande programmering. Detta framgick från lärarnas svar, varje enskild lärare kan bestämma själv på vilken nivå och hur mycket hen ska undervisa i programmering.

Studiens resultat har visat att det finns en stor variation i hur lärare resonerar och förstår programmeringskravet i matematikämnet. Det verkar som att de lärare som har bakgrund i datavetenskap har en mera nyanserad förståelse, de kunde se fler tillämpningsområden för programmering än de lärare som inte hade samma bakgrund. Vad det gäller praktiskt tillämpning av programmerings kravet, genomfördes undervisning på ganska enhetligt sätt - först analog programmering, sen programmering i visuella programmeringsmiljöer, oftast med hjälp av både matteboken och externa resurser - men lärare ägnade olika mycket tid åt det. Detta innebär att fastän det har blivit obligatoriskt att föra in programmering i matematikundervisning, görs det i olika stor utsträckning. Konsekvensen av det kan tänkas vara att det inte är möjligt att säkra att alla elever får lära sig lika mycket och att klyftorna i kunskaper och färdigheter gällande programmering och kanske även matematik kan bli större än vad de är idag.

8.1.2 TPACK

Vi ville se hur lärare resonerar kring programmeringskravet i matematikämnet och hur de tillämpar det i sin undervisning. I vår studie har vi identifierat ett antal resonemang där lärare uttrycker vad programmeringskravet i läroplanen har inneburit. Vi har också frågat lärarna hur de undervisar. Genom att lyssna på lärarnas resonemang och erfarenheter försöker vi identifiera vilken roll de tillskriver programmering i matematikundervisningen. TPACK i det här fallet används som ett verktyg för att beskriva den rollen.

I förhållande till TPACK-ramverket skulle det kunna sägas att analog programmering används främst som ämnesinnehåll, d.v.s. det viktiga är att lära sig om programmering. Det betyder att det handlar om kategorin CK.

Synen på programmering som hjälpmedel i undervisning skulle kunna placeras inom område TCK - kunskap om teknologiskt ämnesinnehåll då matematik används för att ‘införliva och illustrera ett visst ämne och till ämnet relaterade idéer’, d.v.s. programmeringen används för att lära ut ett visst innehåll i matematik.

Programmeringen som hjälpmedel för elever skulle kunna vara en del av TPK - teknologisk pedagogisk kunskap som handlar om att känna till vilka affordanser som programmering

innefattar. Här skulle programmeringens självinstruerande aspekt samt dess effekt på självförtroende kunna passa.

Lärarna resonerar i att programmeringen har en praktisk funktionalitet i matematik som kan förlikas med TK och även TCK, d.v.s. att programmeringen nyttiggör och konkretiserar matematiken. I fallet TK ses programmeringen som ett stöd för matematikämnet (ämnesinnehåll). Samtidigt lär exempelvis eleverna om programmerings användning i att lösa matematiska problem. Även i resonemanget om att programmering och matematik delar begrepp, exempelvis funktioner, värde och variabler kan programmeringen appliceras i TCK eftersom elever kommer att lära sig om dessa begrepp både i matematik och programmering.

Resonemanget som beskriver problemlösning skulle kunna identifieras med TPACK. Detta innebär att ett ämnesinnehåll i matematik (CK) använder programmeringsmiljö som verktyg (TK) att lösa problem på ett särskilt strategiskt sätt (PK).

Ett par resonemang identifierar och applicerar programmering i CK, där lärare tycker att programmeringen är ett ämnesinnehåll eller ämneskunskap som eleverna bör få kunskap om. Argumentet är att programmeringen infördes som ett nytt innehåll i kursplanen för matematikämnet. Detta argument återkommer även i resonemanget om att programmeringen ingår i kunskapen om digitala verktyg. Eleverna behöver lära sig programmering som en ny kunskap CK. Detta syfte med att lära programmering som ett eget ämnesinnehåll CK kan förflyttas så småningom och falla i TK eller TCK. Vi menar att programmering kan användas i senare skede som ett tekniskt verktyg (exempelvis för att nyttiggöra matematik) och/eller som ett verktyg att lära ut ett visst innehåll i matematikämnet.

8.1.3 Tidigare forskning

Här presenteras resultat och analys i förhållande till tidigare forskning.

Några av de resonemang som framfördes av lärare kan kopplas till tidigare forskning. Det har exempelvis resonerats om att programmering kan konkretisera matematik. Feurzeig och Papert, som skapade LOGO (Feurzeig m.fl., 2011), påpekade att matematik och programmering i många fall bygger på samma begrepp och att genom programmeringsspråket LOGO kan dessa abstrakta begrepp konkretiseras, det blir lättare att lära sig dem. Forskarna pratade även om den motiverande effekten som programmering har. Ett liknande resonemang för lärare i denna studie.

I den ovan redovisade forskningen nämns det att det gemensamma mellan programmering och matematik är problemlösning och modellering (Scherer m.fl., 2019). Dessutom utgör problemlösning den gemensamma nämnaren mellan matematik och datalogiskt tänkande, men även mellan datalogiskt tänkande och programmering. I vår studie fanns det några resonemang kring problemlösning och datalogiskt tänkande. Läroplanen talar inte om att det finns en relation eller samexistens mellan matematik, programmering och datalogiskt

tänkande. Istället finner vi enklare riktlinjer för att undervisa och hur stegvisa instruktioner och algoritmer skapas. I ljuset av att datalogiskt tänkande målas upp som något väsentligt i den moderna världen (Grover & Pea, 2013), kan dennas avsaknad i läroplanen framstå som märklig. Som Heintz m.fl. (2016) påpekade, är det dock ganska vanligt att inte skriva om datalogiskt tänkande på ett öppet och direkt sätt i läroplanen. Istället finns datalogiskt tänkande ofta med i någon annan form. Det skulle kunna tolkas å ena sidan som att barnen lär sig det datalogiska tänkandet genom att skapa algoritmer och följa instruktioner. Å andra sidan blir det svårt att avgöra om det är sant, inte minst eftersom datalogiskt tänkande kan definieras på olika sätt. Exempelvis i stöddokumentet Få syn på digitalisering på grundskolenivå beskrivs datalogiskt tänkandet med orden ”problemlösning, logiskt tänkande, att se mönster och att skapa algoritmer” (Skolverket 2017, s.9). Det är dock värt att notera att Armoni (2016) påvisade, att det är tveksamt att datalogiskt tänkande kan utvecklas genom programmering eller något ännu enklare (i Armonis ord – kodning). Det är följaktligen oklart om syftet med att införa programmering är att utveckla datalogiskt tänkande, att lära ut programmering eller att förbättra inlärning av matematik.

Ytterligare en nivå läggs på om hänsyn tas till resultatet i studien av Strawhacker och Bers (2019). Denna studie har visat att programmering kan kopplas till andra kognitiva aspekter såsom det rumsliga tänkandet, den verbala och sociala utvecklingen hos barn. Detta återspeglas inte i läroplanen, däremot återfinns det bland vår studies resultat en iakttagelse att elever genom programmering kan utveckla ett visst tänkande som kan vara nyttigt också i andra sammanhang.

Även resultatet i Duncans m.fl. (2017) studie kan kopplas till resultatet i denna studie. Duncan m.fl. (2017) visade att med rätt kompetensutbildning kunde lärare som förut inte undervisade i datavetenskap bedriva undervisning i detta ämne och förmedla nödvändiga kunskaper till eleverna. Det framgick också att just kunskaper i ämnet datavetenskap var viktiga för lärare, eftersom de hjälpte till att lösa de missuppfattningarna de hade gällande begreppens innebörd och användning. Det verkar vara sannolikt att de skillnader i resonemang och undervisningspraktiker som visades i vår studie skulle kunna kopplas, i alla fall delvis, till lärarnas kompetens inom datavetenskap. Det skulle innebära att kompetensutveckling är viktig för att lärare ska kunna förverkliga förändringen i läroplanen. Tidigare forskning visar dock att det krävs ytterligare åtgärder. Förutom kunskaper i ämnet datavetenskap som sådant behöver lärare utveckla pedagogiska och ämnesdidaktiska strategier (Mannila, 2017; Sentance & Csizmadia, 2017), få lämpliga läromedel (Heintz m.fl., 2017) och skolledningen behöver visa aktivt stöd och säkerställa en stödfunktion för lärare genom exempelvis skolans IT-avdelning (Heintz m.fl., 2017; Sentance & Csizmadia, 2017).

8.2 Studiens validitet

Validitet måste säkerställas i alla stadier i en forskning (Cohen m.fl., 2017). Vi har kunnat koppla våra forskningsfrågor till det teoretiska ramverk vi har använt oss av. Vi har kunnat förstå matematiklärares resonemang om hur de ser på programmeringskravet i

matematikämnet. Det finns alltså en logisk förklaring till hur och varför matematiklärare svarar på ett visst sätt.

Vi kan inte generalisera resonemang som matematiklärare gör om programmeringskravet i matematikämnet, eftersom vi enbart har intervjuat åtta matematiklärare. Men en del av deras resonemang styrks i den tidigare forskningen.

I studien har vi använt intervjuer som en kvalitativ metod. För att få en djupare förståelse kan flera metoder kombineras. Vi ville, med tanke på den relativt korta tiden för examensarbetet, tidsaspekten, ha hanterbara data. Vi vill dessutom säkerställa att vi får svar till forskningsfrågorna i alla intervjuer. Vi har ställt samma intervjufrågor till alla respondenter. Några respondenter känner en av oss som genomför studien, vilket kan minska tillförlitlighet. Men våra respondenter är medvetna om att all deras data anonymiseras. Respondenternas data transkriberas utifrån de inspelade ljudfilerna. I dataanalysen har vi upprepade gånger lyssnat på intervjuerna när vi kategoriserat (och kodat) hur vi förstår att matematiklärare resonerar. Resultatet av analyserna, styrks också av samstämmighet med tidigare forskning.

Vi vill även betona att vi inte har använt alla intervjufrågorna till forskningsfrågor utan enbart de intervjufrågor som är mest relevanta för att besvara forskningsfrågorna. De andra intervjufrågorna är kompletterande frågor och de har bidragit till en bättre förståelse av det område som vi undersökte.

8.3 Slutsatser

Det finns aktörer som tycker att programmering är viktigt för framtiden och att matematiken är en anledning att få in programmeringen i skolans läroplan.

Forskare ser att det finns en koppling mellan matematik och programmering. De menar att programmering kan användas på ett meningsfullt sätt genom exempelvis att programmering begripliggör matematiken. Detta stämmer överens med resonemang som några lärare gör om att programmering konkretiserar matematiken.

Motivationen till införandet av programmeringen i läroplanen tycks vara otydlig för några av våra lärare. IT-politiken hävdar att programmering är en viktig kunskap i samhället och att det kommer att skapa framtida jobb. Digitaliseringskommissionen (SOU 2016:89) menar att digitaliseringen är viktig för demokratin och framtida utveckling, eftersom den skapar nya möjligheter för både enskilda individer och samhället i stort.

Skolverket inför programmering som verktyg för att det ska användas i matematikämnet. I praktiken har vissa matematiklärare resonerat som att programmering är ett ämnesinnehåll och införandet innebär att undervisa i programmering. Det fanns åsikter bland lärare som menar att programmeringen utvecklar generella kunskaper. I vår tidigare forskning hittar vi England som har infört programmeringen som ett eget IKT-ämne. Även om lärare använder programmeringen i matematik behöver både lärare och elever tillägna sig kunskaper om programmering i första skede.

I vår studie var alla lärare eniga om att programmering är en viktig kunskap som elever behöver lära sig om. Dessutom visar vår studie att alla respondenter använder programmering i sin undervisning om än till olika stor utsträckning. De använder både så kallad analog programmering och programmering i visuella miljöer. Våra respondenter uppfyller därmed programmeringskravet så som det finns beskrivet i läroplanen. Det går dock inte att konstatera att lärarna, utifrån befintliga förutsättningar, kan säkerställa att alla elever får utbildning i programmering av samma kvalitet.

8.4 Avslutande reflektioner

När studiens resultat sätts i ett sammanhang genererar det några reflektioner. Det verkar finnas en viss diskrepans i läroplanen. Å ena sida framhålls det vikten av att utveckla datalogiskt tänkande och problemlösningsfärdighet, detta är en del av de kunskaper som alla behöver ha på 2000-talet. Å andra sidan påstås det att det är viktigt att lära sig programmera för att det finns ett stort behov av människor som kan programmera. Frågan uppstår vilken roll har matematiken i det hela?

Enligt några forskare så finns det en koppling mellan matematik, programmering och datalogiskt tänkande. De menar att programmering kan användas på ett meningsfullt sätt, exempelvis att programmering konkretiserar matematiken samt att digitala verktyg möjliggör att elever kan lära sig tänka matematiskt och logiskt och därmed kan lösa matematiska problem (Feurzeig m.fl., 2011). De hävdar också att genom programmering utvecklas datalogiskt tänkande som påstås vara en mycket viktig förmåga i dagens och framtida samhälle (Grover & Pea, 2013; Scherer m.fl., 2019; Sung m.fl., 2017). I Sverige var några forskare med och fungerade som stöd när förändringarna gällande läroplanen förbereddes (Heintz m.fl., 2017). De var dock inte de enda aktörerna som medverkade i processen. Efter att ändringarna trädde i kraft, beklagade Heintz m.fl. (2017) att programmering har införts som ett snävt innehåll, de menade att viktiga delar av akademiskt ämne datavetenskap gick förlorade. Heintz m.fl. (2017) skriver inte rakt ut att det handlar om datalogiskt tänkande, men eftersom de har varit med i den grupp som aktivt arbetade med att främja datalogiskt tänkande (Heintz m.fl., 2015), tror vi att det är just det de menar, d.v.s. att datalogiskt tänkande är en av de delarna som saknas i läroplanen.

De andra medverkande aktörerna skulle kunna beskrivas som politiker och ”marknad”. Sveriges IT-politik exempelvis bygger på synen att programmeringen är en viktig kunskap i samhället och att teknisk utveckling kommer att skapa framtida jobb. Digitaliseringskommissionen (SOU 2016:89) hävdar att digitaliseringen kan göra att vi utvecklar ett demokratiskt och hållbart välfärdssamhälle som innebär utveckling. Denna utveckling ger nya förutsättningar, behov och villkor för individ och samhälle, för företag och offentlig sektor, för arbetsliv och utbildning och för civilsamhället. Detta är en bred och omfattade ansats som kan delvis återspeglas i stöddokumentet: *Få syn på digitaliseringen på grundskolenivå* (2017), men inte så mycket i läroplanen för grundskolan.

I kursplanens centrala innehåll för matematik verkar det förmedla något helt annat. Utifrån denna text skulle det vara möjligt att påstå att Skolverket införde programmering i grundskolan som ny kunskap som ska läras ut, d.v.s. barnen ska lära sig att programmera. Sättet som lärare införde programmering i sin undervisning på, i alla fall hittills, kan sägas stämma överens med det påståendet - resultatet i vår studie har visat att fokuset ligger på att lära elever att programmera. Att ha många programmeringsexperter kan tänkas vara attraktivt för marknaden. Sett utifrån utgångspunkten som grundar sig i matematiken, kan dock samma fråga ställas - var finns matematiken i det hela? Än så länge verkar det som att matematik används som en anledning att föra in programmering i skolans läroplan. Den utpekade relationen mellan programmering och matematik framgår än så länge inte i kursplanen på ett tydligt sätt. Det går att påstå att den finns framskriven i kursplanens syfte men det ges inte klara riktlinjer för hur lärare ska åstadkomma detta syfte.

Det är intressant att fundera på varför relationen mellan matematik och programmering är otydlig i läroplanen. Är det fråga om transparens? Okunnighet? Återspeglar det viljan att skona lärare från allt för stor stress? Eller är det kanske ett halvlyckat försök att förena intressen av väldigt olika aktörer, exempelvis forskarnas, politikers och marknadens? Oavsett vilken anledningen är, den uteblivna relationen kan tänkas vara problematisk för lärare eftersom att det inte är tydligt hur programmering kan bidra till lärande i matematik. Skulle man ändå gå längre, då kunde man ställa en fråga om det inte är tvärtom - att det är snarare matematik som används för att lära programmering.

Samtidigt kan den vaghet som just nu finns i kursplanens formuleringar ses som ett naturligt skede och det finns hopp att med tiden kommer det att konkretiseras hur programmering bör undervisas och vilka krav som gäller för elever. Analysen utifrån TPACK-ramverket har också visat att programmering inte har någon bestämd roll och att det finns en förflyttning i vad som görs med programmeringen i matematikundervisning.

Det finns flera möjligheter för framtida studier. Det skulle vara intressant att undersöka innehållet i läromedel som lärare använder. Det skulle också vara givande att göra observationer för att titta närmare på hur undervisningen i matematiken går till. Vidare skulle det vara intressant att få veta lite mer om hur barnen förstår varför de lär sig programmering i matematikämnet. En annan möjlig forskningsinriktning skulle kunna vara att undersöka om programmeringskravet i matematikämnet bidrar till att utveckla matematikkunskaperna hos eleverna.

Referenser

Ahmed, G., Nouri, J., Zhang, L. & Norén, E. (2020). Didactic Methods of Integrating Programming in Mathematics in Primary School: Findings from a Swedish National Project. *SIGCSE '20: Proceedings of the 51st ACM Technical Symposium on Computer Science Education February 2020*, ss. 261–267. https://doi.org/10.1145/3328778.3366839

Armoni, M. (2016). Computer science, computational thinking, programming, coding: The anomalies of transitivity in K-12 computer science education. *ACM Inroads*, *7*(4), 24-27.

Brantley-Dias, L., & Ertmer, P. A. (2013). Goldilocks and TPACK: Is the construct "just right?". *Journal of Research on Technology in Education*, *46*(2), ss. 103-128. doi:10.1080/15391523.2013.10782615

Brown, N. C. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education* (TOCE), 14(2), ss. 1-22. doi:10.1145/2602484

Cohen, L., Manion, L. & Morrison, K. (2017). *Research Methods in Education* (8. utg.). London: Routledge.

diSessa, A.A. (2018). Computational Literacy and “The Big Picture” Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning*, *20*(1), 3-31.

Dostal, J., Wang, X., Nuangchalerm, P., Brosch, A., & Steingartner, W. (2018). Researching computing teachers' attitudes towards changes in the curriculum content - an innovative approach or resistance? Paper presented at the *Proceedings of the 2nd International Conference on Informatics and Computing, ICIC 2017*, *2018-January* 1-6. doi:10.1109/IAC.2017.8280531 Retrieved from www.scopus.com

Dunca, A., Bell, T., & Atlas, J. (2017). What do the Teachers Think? Introducing Computational Thinking in the Primary School Curriculum. *Proceedings of the Nineteenth Australasian Computing Education Conference*, 31 January 2017, 65-74. DOI: http://dx.doi.org/10.1145/3013499.3013506

Europaparlamentets och rådets rekommendation 2006/962/EG av den 18 december 2006 om nyckelkompetenser för livslångt lärande. *Europeiska Unionens Officiella Tidning*, L 394, 10-18. Hämtad från http://eur-lex.europa.eu/legal-content/SV/TXT/?uri=LEGISSUM:c11090.

Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, *19*(5), 487-501. doi:10.1080/10494820903520040

Forsström, S.E., & Kaufmann, O.T. (2018). A Literature Review Exploring the use of Programming in Mathematics Education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18-32.

Funke, A., Geldreich, K., & Hubwieser, P. (2016). Primary school teachers' opinions about early computer science education. Paper presented at the *ACM International Conference Proceeding Series*, 135-139. doi:10.1145/2999541.2999547 Retrieved from www.scopus.com

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

Heintz, F., Mannila, L., Nordén, L.-Å., Parnes, P., & Regnell, B. (2017). Introducing programming and digital competence in swedish K-9 education. *10th International Conference On Informatics In Schools: Situation, Evolution, And Perspectives, Issep 2017, Helsinki, Finland, 2017-11-13 - 2017-11-15, 10696 LNCS*, 117-128.

Heintz, F., Mannila, L., Nygårds, K., Parnes, P., & Regnell, B. (2015). Computing at School in Sweden – Experiences from Introducing Computer Science within Existing Subjects. *Proceeding Of The 8th International Conference On Informatics In Schools: Situation, Evolution, And Perspective (Issep)*, 118-130.

Helenius, O., Misfeldt, M., Rolandsson, L., & Ryan, U. (2018). *Om programmering i matematikundervisning*. Hämtad 2020-05-20 från https://larportalen.skolverket.se/LarportalenAPI/api-v2/document/path/larportalen/material/inriktningar/1-matematik/Grundskola/417_matematikundervisningmeddigitalaverktygII_åk1-3/del_01/Material/Flik/Del_01_MomentA/Artiklar/MA2_1-3_01A_01_omprogrammering.docx

Hsieh, H., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research, 15*(9), 1277-1288.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38-43.

Kjällander, S., Åkerfeldt, A. & Petersen, P. (2016). *Översikt avseende forskning och erfarenheter kring programmering i förskola och grundskola*.

Koehler, M. J., & Mishra, P. (2009). What Is Technological Pedagogical Content Knowledge? *Contemporary Issues in Technology and Teacher Education (CITE Journal), 9*(1), 60-70.

Koh, J., & Divaharan, S. (2011). Developing pre-service teachers' technology integration expertise through the TPACK-developing instructional model. *Journal of Educational Computing Research, 44*(1), 35-58. doi:10.2190/EC.44.1.c

Kopcha, T. J., Ottenbreit-Leftwich, A., Jung, J., & Baser, D. (2014). Examining the TPACK framework through the convergent and discriminant validity of two measures. *Computers and Education, 78*, 87-96.

Larke, L. R. (2019). Agentic neglect: Teachers as gatekeepers of England's national computing curriculum. *British Journal of Educational Technology, 50*(3), 1137-1150. doi:10.1111/bjet.12744

Lie J, Hauge IO, Meaney T. Computer Programming in the Lower Secondary Classroom: Mathematics Learning. *Italian Journal of Educational Technology*. 2017;25(2) 10.17471/2499-4324/911

Mannila, L. (2017). *Att Undervisa I Programmering I Skolan : Varför, Vad Och Hur?* Upplaga 1 ed. Lund: Studentlitteratur.

Mayring, P. (2000). Qualitative content analysis. *Forum : Qualitative Social Research*, 1(2).

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1-17. doi:10.1080/20004508.2019.1627844

Pareto, L., & Willermark, S. (2019). TPACK in situ: A design-based approach supporting professional development in practice. *Journal of Educational Computing Research, 57*(5), 1186-1226. doi:10.1177/0735633118783180

Parr, G., Bellis, N., & Bulfin, S. (2013). Teaching english teachers for the future: Speaking back to TPACK. *English in Australia, 48*(1), 9-22. Retrieved from www.scopus.com

Regeringensbeslut (2017). *Stärkt digital kompetens i skolans styrdokument. H*ämtad från ttps://www.regeringen.se/contentassets/acd9a3987a8e4619bd6ed95c26ada236/informationsmaterial-starkt-digital-kompetens-i-skolans-styrdokument.pdf

Repstad, P. (2007). *Närhet och distans: kvalitativa metoder i samhällsvetenskap.* Lund: Studentlitteratur.

Rolandsson, L. (2013a). Changing Computer Programming Education; The Dinosaur that Survived in School. *Proceedings - 2013 Learning and Teaching in Computing and Engineering*, LaTiCE 2013, ss. 220-223

Rolandsson, L. (2013b). Teachers' beliefs regarding programming education. *Technology teachers as researchers: Philosophical and empirical technology education studies in the swedish TUFF research school*, ss. 285-309. doi:10.1007/978-94-6209-443-7 Retrieved from www.scopus.com

Rolandsson, L. & Skogh, I.-B. (2014). Programming in School: Look Back to Move Forward. *ACM Transactions on Computing Educations*, 14(2).

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 22(2), 469-495. doi:10.1007/s10639-016-9482-0

Scherer, R. (2016). Learning from the past-the need for empirical evidence on the transfer effects of computer programming skills. *Frontiers in Psychology,* 7 (1390) doi:10.3389/fpsyg.2016.01390

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

Shute,V., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142-158.

Skolverket (2019). *Läroplan För Grundskolan, Förskoleklassen Och Fritidshemmet 2011 : Reviderad 2019*. Sjätte Upplagan ed. 2019.

Skolverket. (2017). *Få syn på digitaliseringen på grundskolenivå: Ett kommentarmaterial till läroplanerna för förskoleklass, fritidshem och grundskoleutbildning*. Stockholm: Skolverket Wolters Kluwer.

SOU 2014:13. *En digital agenda i människans tjänst - en ljusnande framtid kan bli vår*. Stockholm: Fritzes Offentliga Publikationer.

SOU 2016:89. *För Digitalisering I Tiden: Slutbetänkande*. Stockholm: Wolters Kluwer, 2016. Print. Statens Offentliga Utredningar, 2016:89.

Strawhacker, A., & Bers, M. (2019). What they learn when they learn coding: Investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development, 67*(3), 541-575.

Sung, W., Ahn, J., & Black, J. B. (2017). Introducing computational thinking to young learners: Practicing computational perspectives through embodiment in mathematics education. *Technology, Knowledge and Learning, 22*(3), 443-463. doi:10.1007/s10758-017-9328-x

Vetenskapsrådet (2017). *God forskningssed*. Stockholm: Vetenskapsrådet.

Vivian, R., & Falkner, K. (2019). Identifying teachers' technological pedagogical content knowledge for computer science in the primary years. Paper presented at the *ICER 2019 - Proceedings of the 2019 ACM Conference on International Computing Education Research,* 147-155. doi:10.1145/3291279.3339410

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127-147. doi:10.1007/s10956-015-9581-5

Willermark, A. (2018). *Digital Didaktisk Design. Att utveckla undervisningspraktiken i och för en digitaliserad skola* (Doktorsavhandling, Informatics with Specialization in Work-Integrated Learning, 13). Trollhättan: BrandFactory AB. Tillgänglig: http://discover.hv.se/iii/encore/record/C__Rb1117713

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

Wong, G. K. W., Cheung, H. Y., Ching, E. C. C., & Huen, J. M. H. (2016). School perceptions of coding education in K-12: A large scale quantitative study to inform innovative practices. Paper presented at the *Proceedings of 2015 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2015,* 5-10. doi:10.1109/TALE.2015.7386007 Retrieved from www.scopus.com

Bilaga 1 - Intervjuguide

1. **Bakgrund**
    a. Hur länge har du jobbat som lärare, vilka å.k. och ämnen undervisar du i?
    b. Vilka ämnen har du behörighet att undervisa i?

2. **Kunskaper om programmering och datalogiskt tänkande**
    a. Vad är programmering för dig?
    b. Hur skulle du beskriva begreppet datalogiskt tänkande?

3. **Programmeringskrav**
    a. Hur fick du reda på förändringar i läroplanen?
        i. Har du tagit del av den läroplanens formuleringar? I så fall NÄR och HUR/I VILKEN FORM?
        ii. Har du läst originaltexten eller tagit del av dem i andra typer av dokument? I så fall vilka?
        iii. Har den nya läroplanen gåtts igenom i skolan/lärarkollegiet eller har du gjort det själv?
    b. Känner du till dokumentet ”Få syn på digitalisering i grundskolan”?
        i. I så fall, hur har du använt det?
    c. Använder du något annat som stöd i din matematikundervisning som har med programmering att göra?
        i. I så fall, hur använder du det?
    d. Hur förstår du programmeringskravet i matematik?
    e. Vilken är din personliga åsikt om ändringarna/de nya kraven?
    f. Vad tror du kommer att ske med programmering i grundskolan i framtiden?

4. **Programmering – matematikundervisning**
    a. Hur kan programmering stödja elevernas utveckling i matematik?
    b. Vilka fördelar och nackdelar ser du med programmering som redskap i matematikundervisning?

5. **Programmering i den egna matematikundervisningen**

a. Hur arbetar du med programmeringen i din matematikundervisning? Alt. Hur tänker du arbeta med programmeringen i matematikundervisningen? (ge konkreta exempel, visa)
b. Vilka förändringar (anpassningar) behöver du göra i din planering av mattelektionerna efter införande av programmering i läroplanen?
c. Hur har eleverna tagit emot programmeringsinslagen i undervisningen?
d. Ser du sambandet mellan att man är bra på att programmera så blir man bättre i matematik? Vilket?

Bilaga 2: Informationsbrev

Hej!

Vi är två studenter som läser Magisterprogrammet Lärande, kommunikation och IT vid institution för Tillämpad informationsteknologi på Göteborgs universitet. Mer information om utbildningen hittar du här.

Anledningen till att vi kontaktar dig är att vi skriver en magisteruppsats om matematikundervisning i grundskolan. Vi undersöker lärares resonemang samt praktisk tillämpning av det nya kravet att använda programmering i matematikundervisningen. Utifrån våra forskningsfrågor har vi identifierat dig som en person som kan vara relevant att kontakta för en intervju.

Vi vill bjuda in dig att delta i studien genom att svara på några frågor i form av en intervju som kommer att ta ca 30 min. Intervjun kommer också att spelas in för vår egen dokumentation. Inspelningen kommer att förvaras i säkerheten och endast vi som gör denna studie kommer att ta del av den. All data kommer att raderas efter att studien är avslutad.

Personuppgifter kommer inte att delas med tredje part och de kommer att hanteras enligt *Personuppgiftslagen* (1998:204)[1] och *Dataskyddsförordningen* (GDPR). Varken ditt namn eller namnet på skolan som du jobbar i kommer att skrivas ut i uppsatsen.

Informationen som kommer att samlas in ska användas enbart i syfte att svara på studiens forskningsfrågor, dvs den kan inte återanvändas i andra studier.

Ditt deltagande är frivilligt och du har rätt att när som helst avbryta utan att behöva förklara varför. Så som studien nu är utformad är den helt beroende av kvalitativa intervjuer och därför skulle din medverkan uppskattas oerhört mycket.

Om du har några frågor kring ditt deltagande, är du välkommen att höra av dig till oss.

Med vänlig hälsning

Arkan Shefram
arkan.shefram@grundskola.goteborg.se
070-773 04 52

Ksenija Peggar
ksenija.peggar@outlook.com
072-551 92 11

[1] Dataskyddsombud på Göteborgs universitet är Johanna Wallin, e-post: dataskydd@gu.se

Bilaga 3: Blankett för samtycke

Samtycke till deltagande i forskningsstudie

Jag har tagit del av informationen om forskningsstudien ”Programmering i mellanstadiet - *Hur resonerar och tillämpar lärare programmering i matematikämnet”* samt jag har fått information om att deltagandet är frivilligt och att deltagande kan avbrytas när som helst.

□ Ja, jag deltar i studien. Video- och/eller ljudinspelningar kan användas för denna studie.

Ort och Datum:

Namnteckning: _____

Namnförtydligande: _____