De finlandssvenska matematiklärarnas beredskap att undervisa programmering

Emil Borg

Magisteravhandling i matematik

Fakulteten för naturvetenskaper och teknik

Åbo Akademi

Handledare: Mikael Lindström

Kim-Erik Berts

Vasa, 2022

Abstrakt

Författare	Årtal
Emil Borg	2022

Arbetets titel

”De finlandssvenska matematiklärarnas beredskap att undervisa programmering”

Avhandling för magisterexamen i matematik	Sidantal
Vasa: Åbo Akademi. Fakulteten för naturvetenskaper och teknik	51

Abstrakt

Programmering, som har undervisats i grundläggande utbildningens alla årskurser sedan hösten 2016 medförde en stor förändring för matematiklärarna i årskurserna 7–9. Den grundläggande orsaken till att förändringen inte välkomnades varmt var att en stor andel av lärarna inte hade mycket erfarenhet av programmering från sina studier. Denna avhandling belyser vilka förutsättningar Svenskfinlands lärarkår har och hur de arbetar med utmaningarna som har uppstått som resultat av förändringen. Problemet som undersöks och står i centrum är lärarnas varierande kunskap inom området.

I dagsläget specificerar inte läroplanen vilka programmeringsmiljöer eller språk som bör undervisas. Därför presenteras ett urval av valmöjligheterna som lärare ställs inför idag, och deras för- och nackdelar diskuteras. Lärarnas val av undervisningsmiljö och till vilken grad programmering integreras med övrig matematikundervisning granskas.

Forskningsfrågorna som avhandlingen ämnar besvara är:

1. Vilken kunskapsnivå i programmering och vilka behov av sådan fortbildning finns det bland matematiklärarna i årskurs 7–9?
2. Är ålder och undervisningserfarenhet en faktor i hur matematiklärare i årskurs 7–9 förhåller sig till att undervisa programmering? Vilka övriga faktorer finns det?
3. Vad styr lärarnas val av undervisningsmiljö och arbetssätt?

Undersökningen, som genomfördes i slutet av vårterminen 2021, var enkätbaserad och data från 49 deltagare i 23 kommuner registrerades. Frågorna var delvis av kvalitativ, och delvis av kvantitativ natur, och några av lärarnas åsikter representeras som citat eller frekvenstabeller.

Ramverket TPACK används för att avgöra om lärarna begränsas av tekniska, pedagogiska eller innehållsmässiga utmaningar. Det visar sig att bara en knapp majoritet anser att programmeringen är så allmännyttigt att den borde undervisas åt alla, medan en minoritet håller med om att detta bör ske som en del av matematikundervisningen.

Endast 60% tycker att deras kunskap räcker till för uppdraget. Eftersom kunskapen har ett samband med attityden kunde situationen förbättras genom ytterligare fortbildningar för lärare, och universitetskurser för blivande lärare borde övervägas. Det som däremot inte har något samband med lärarns upplevda kompetens att undervisa programmering, eller lärarens avlagda kurser i programmering är åldern, vilket tyder på att lärarna varken studerar mer eller mindre datarelaterade kurser än förr. Det visar sig också att lärare har fortsatt använda sig av läromedel som inte ännu inkluderar programmering, medan de som å andra sidan har förnyat sin lärobok är mycket mer benägna att inspireras till att undervisa i en textbaserad programmeringsmiljö.

Den tekniska utrustningen skolan har tillgång till utgör ibland ett hinder, men är oftast inte det som uppges vara avgörande. Många lärare noterar däremot att eleverna uppvisar stora skillnader i kunskapsnivå, speciellt i programmering, och att det utgör ett problem. På liknande sätt avslöjar denna studie att det förkommer varierande åsikter, intresse och kunskapsskillnader hos lärarna, och friheten i läraryrket resulterar i att dessa skillnader i programmering hos eleverna riskerar förstärkas i årskurserna 7–9 då undervisningens mängd och metod kastar.

Sökord

Programmering, matematik, ämneslärare, programmeringsmiljö, grundskolans högstadium, läroplan, algoritmiskt tänkande, TPACK

Innehåll

1 Inledning

1.1 Bakgrund

Viljan att förnya matematikundervisningen genom att införa programmering som ett verktyg eller till och med ett eget temaområde är ingalunda ny. Programmering kunde så klart existera som ett separat ämne. Som delområde inom matematiken kunde programmering vara jämförbart med stora helheter såsom geometri eller ekvationslösning, med dess egna metoder och enorma tillämpningsområde. Artikeln ”Programming and Learning Implications for Mathematics Education” (Hansen & Zweng, 1984) inleds med följande ord: *“*Programming to promote mathematics learning is not new*”*. Nästan fyra decennier senare väljer jag därför att upprepa påståendet, men med lite mera slagkraft. Det är sannerligen ingen nyhet vid det här laget! Visionen av hur programmeringen som delmängd i undervisningsämnet borde se ut har inte förändrats så mycket.

Däremot har grundskolans läroplan genomgått många förändringar, och till följd av den senaste som trädde i kraft 2016 har ämneslärare och klasslärare försatts i en situation där de ska undervisa ett ämne de kanske inte har mycket erfarenhet av, om någon alls. (Ifous, 2020) En liknande situation råder i grannlandet Sverige, där regeringen 2017 klubbade igenom ett liknande tillägg i den svenska grundskolans läroplan. Skillnaden till den finska implementationen är att algoritmer och programmering har listats som ett av fem mål under algebra. (*Läroplan för grundskolan, förskoleklassen och fritidshemmet*, 2019)

Att lärare kan känna sig osäkra i en situation där de undervisar något nytt ämnesområde är inte långsökt. Det är möjligt att lärarna har stött på programmering under sina studier. Däremot är det inte alls garanterat att en lärare i dagens läge har fått chansen att vara deltagare i en lektion med programmeringsinslag under sin egen grundskoltid.

Har läraren fått den chansen så var utrustningen och programmeringsspråket högst sannolikt väldigt olik dagens.

1.1.1 Nu och då

Kontrasten mellan situationen år 1984, som nämndes i förgående avsnitt, och 2022 är däremot påtaglig. Särskilt ifall möjligheterna som finns tack vare teknikens exponentiellt växande utbredning och tillgänglighet beaktas. Grafiska användargränssnitt i datorer som privatpersoner hade råd och intresse att köpa lanserades 1984 av företaget Apple, i en tid då datorer var långt ifrån varje människas egendom. I och med denna övergång förvandlades datorn från en programmerbar maskin till en vardaglig och användarvänlig apparat, vilket även anses vara nyckeln till *Apple IIs*, och senare produktvarumärket *Macintoshs* stora kommersiella framgång. (Barnes, 2010)

Att använda teknologi 1984 var ett medvetet val och krävde förkunskap och ansträngning för att komma i gång, oavsett hur enkel uppgift man ämnade utföra. Idag utnyttjar vi ofta datateknologi utan att tänka på det. Användare tvingades själva utföra många steg i en process som idag är automatiserad. Samtidigt gjorde förarbetet att användaren hade större insikt i mekanismen. Barn som är födda och uppväxta med den digitala tekniken saknar förståelse för den. Detta är ett uppenbart problem, att en diskrepans i kunnande och användningsgrad växt så stor och ännu ökar. I dagens samhälle är det snarare ett medvetet val att avstå från att använda teknologi. (Mannila, 2017)

Tröskeln till att lära sig om datorer och programmering har aldrig varit lägre, och den sjunker ännu. Antalet datorer per elev i de finländska grundskolorna ökar stadigt, och ökningen kommer sannolikt inte att bromsa in före ett 1:1 förhållande nås. Samtidigt är Finland på efterkälken i detta avseende. Enligt PISA-undersökningen 2018 (OECD, 2020) låg antalet datorer per elev på 0,6. Jämfört med Norge, Sverige och Estland som låg på 1,0; 1,1 och 1,2 datorer per elev så har många finska lärare än så länge kvar det extra steget att boka skolans gemensamma datorer. Av både personlig erfarenhet och samtal med kolleger kan jag intyga att detta extra moment i början av en lektion kan

stjäla allt för mycket undervisningstid. Först då elever har personliga enheter att arbeta på kan lärare få chansen att starta programmeringslektionerna lika lätt som då klassen ombeds att öppna matteboken på önskad sida. Brist på material kan tänkas påverka både lärares och elevers upplevelser av programmering negativt då tid och energi används på att lösa triviala problem.

1.1.2 Programmering som en grundfärdighet

Den distinkta skillnaden mellan att studera programmering och användandet av programmering för att uppnå andra lärandemål betonades tidigt (Hansen & Zweng, 1984). Linda Mannila upprepar samma tanke (Björklund, 2015). Hon menar att programmeringen kan främja utvecklandet av färdigheter inom flera områden, och att datalogiskt tänkande kan jämföras med grundfärdigheter så som läsning eller skrivning. Därför är det viktigt att låta programmeringen överskrida ämnesgränserna och inte bli ett självändamål. Utöver matematik har också ämnen såsom teknisk slöjd, fysik, kemi och musik god potential att bistå med praktiska programmeringsövningar tack vare ett brett utbud av billiga mikrokontroller med tillbehör.

En av dessa är *Micro:bit*, en mångsidig programmerbar mikrodator som utvecklades av BBC i undervisningssyfte efter att en liknande läroplansuppdatering skapade behovet i Storbritannien. Elever som arbetar med att skapa något som involverar en mikrodator övar inte bara på programmering. De får samtidigt en kontext till problemen de löser och tränar upp övriga viktiga framtidskompetenser, såsom arbetssätt och kommunikationsförmåga. (Korhonen m. Fl., 2019.)

1.2 Forskningsfrågor

Avhandlingen tar avstamp i en tidigare undersökning (Kallio-Kujala, 2017) om hur klasslärare i Svenskfinland hanterar och förhåller sig till att undervisa programmering. Målet är att utvidga den här forskningen till att också inkludera matematiklärarna i årskurs 7–9, för att se hur de förhåller sig till situationen och på så sätt bidra till en heltäckande bild av hela grundskolan. Från början var det meningen att granska hur lärarna ser på kopplingen mellan matematiken och programmeringen och i vilka delområden av matematiken de väljer att lyfta fram programmering som ett verktyg, men allt efter att mina studier framskred och jag fick chansen att stiga in i olika klassrum i landet insåg jag att grundförutsättningarna, dvs material och läromedel, verkade spela en större roll i hurudan undervisning eleverna fick ta del av.

Trots att jag på förhand hade den uppfattningen att nyutbildade lärare skulle vara bättre rustade och mera positivt inställda till detta nya och teknologirelaterade tillägg i läroplanen än veteranerna på fältet stötte jag så småningom på motexempel. Den förhandsuppfattningen började jag ifrågasätta och önskade utmana.

Fenomenet där lärare följer läroboken som om den representerar läroplanen är sedan länge känt (Emanuelsson, 1986), men hur arbetar lärarna då temat är så nytt att läroboken inte alls innehåller något om det? Väljer lärarna att fortsätta använda en gammal lärobok av ekonomiska skäl eller för att de inte hittar nytt material som de gillar? Följande frågor har jag utgått ifrån då jag utförde undersökningen.

4. Vilken kunskapsnivå i programmering och vilka behov av sådan fortbildning finns det bland matematiklärarna i årskurs 7–9?
5. Är ålder och undervisningserfarenhet en faktor i hur matematiklärare i årskurs 7–9 förhåller sig till att undervisa programmering? Vilka övriga faktorer finns det?
6. Vad styr lärarnas val av undervisningsmiljö och arbetssätt?

2 Programmering i undervisningssammanhang

2.1 Separat eller integrerat med matematiken?

Beslutet om hur programmeringen ska förverkligas faller idag till stor del på matematiklärare som på egen hand eller inom ämnesgruppen planerar sina lektioner: ska programmeringen integreras med matematiken och hur, eller ska den förvisas till enskilda lektionssekvenser i slutet av terminen? En hybridstrategi som observerats är att läraren först fokuserar på datalogiska begrepp under en lektion och tacklar de vanliga utmaningar som uppstår kring programmeringen, och senare använder programmeringen som ett verktyg för matematiskt lärande (Ifous, 2020).

Integreringen i matematiken gör nu att alla ämneslärare i matematik förväntas undervisa det. Detta är ett problem för de ämneslärare i matematik som inte har deltagit i programmeringskurser under sin utbildning. Några lärare befinner sig nu i en situation där de tvingas undervisa ett ämne som de i värsta fall inte har någon erfarenhet av (Ifous, 2020). Fortbildningar har under årens lopp sakta men säkert åtgärdat brister, och lärarna har med kollegialt samarbete och egna studier utvecklat sitt kunnande.

Detta problem hade eventuellt inte uppstått ifall ett separat ämne hade införts. Medan Finland och Sverige har infört programmering i matematikämnet har Storbritannien introducerat skolämnet *computing* för åldrarna 5 till 16. Utöver programmering tränar ämnet upp diverse digitala kompetenser, som i den finska läroplanen finns utströdda i nästan alla ämnen, samt praktisk integrering av elektronik i handarbeten och andra praktiska skolprojekt (Department for Education, 2013).

2.2 Läroplansgrunderna för grundläggande utbildningen

Läroplansgrunderna fastställer elevernas mål i programmering på följande vis:

> ”De fördjupar sitt algoritmiska tänkande. De programmerar och tränar samtidigt god programmeringspraxis. Eleverna tillämpar egna eller färdiga datorprogram i matematikstudierna.” (Utbildningsstyrelsen 2014, hädanefter LPGLU 2014)

Dessa tre mål kan man arbeta för på olika sätt. Lärarkåren förväntas själv välja plattform och verktyg, och behöver därmed inneha tillräckligt god kunskap och information för att göra ett gott val. Ingen specifik plattform, miljö eller programspråk fastställs. Däremot måste även lokala läroplaner beaktas. De utarbetas av utbildningsanordnaren och anpassas för att gå i tandem med kommunernas övriga verksamhet. Ofta finns det specifikationer i dessa lokala läroplaner gällande vilka verktyg som används, för nödvändig utrustning kan då lånas ut till flera skolor under årets lopp. Trots det är det inte omöjligt att elever från samma skola med olika lärare har jobbat med programmering på väldigt olika sätt, särskilt på orter där noggrannare detaljer om temat inte inkluderats i lokala läroplaner.

I slutet av 2020 kompletterade utbildningsstyrelsen LPGLU 2014 med ”Bedömningskriterier för årskurs 9 vid slutbedömningen”. Detta klargjorde vilken sorts progression eleverna kan tänkas följa. För vitsordet 5 ska eleven känna igen steg i en algoritm och testa färdiga program, för vitsordet 7 skall eleven behärska villkor och upprepningar och självständigt kunna bekanta sig med program. För högre vitsord ska dessa verktyg användas då eleven gör egna program, och eleven ska själv kunna avgöra vilka strukturer som behövs (dvs tillämpa algoritmiskt tänkande).

2.2.1 Övergångsperioden efter den nya läroplanens införande

Läroplanen för den grundläggande utbildningen introducerar programmering för hela grundläggande utbildningen, inte bara årskurserna 7–9. Trots detta tillägg jämfört med läroplansgrunderna från 2004, som överhuvudtaget inte nämner temat, har Statsrådets förordning om ändring i timfördelningen i grundskolan [793/2018] inte rubbat matematikens 32 årsveckotimmar. En förändring hade varit en fingervisning om programmeringsundervisningens omfattning. Även i årskurserna 1–6 föreskriver LPGLU 2014 att tid under matematiklektioner används på att träna upp förmågan att följa och ge instruktioner, samt att eleverna bekantar sig med t.ex. visuella programmeringsmiljöer. Detta skapar följaktligen en 10-årig övergångsperiod, under vilken elever som har börjat programmera allt tidigare, arbetar sig uppåt igenom årskurserna. Med andra ord, då läroplanen togs i bruk år 2016, kommer vi inte att se dess fulla effekt innan skolåret 2025–2026. Våren 2020 examinerades de första niondeklassarna som har följt LPGLU 2014 under hela sin tid i årskurs 7–9.

Samtidigt har även lärarna mycket utrymme att utveckla sin undervisning inom denna tidsram. Jämsides förändras dessutom teknologin och världen omkring klassrummet. Mitt under denna period drabbades världen av coronapandemin som tvingade varje lärare att se över sin tekniska verktygslåda. En undersökning visade att lärare upplevde att deras tekniska kompetens steg medan den upplevda pedagogiska kompetensen sjönk med erfarenheten av distansundervisning (Hasanovic, 2020).

2.2.2 Algoritmiskt och datalogiskt tänkande

Förmågan att kunna lösa problem genom att bryta ner dem i mindre beståndsdelar, ordnade i en sekvens och lösa dessa delproblem med en motsvarande sekvens av instruktioner är essensen i algoritmiskt tänkande. Det algoritmiska tänkandet förbereds i årskurserna 1–6 genom att eleverna tränar på att skapa och utföra stegvisa instruktioner i diverse algoritmer – till exempel divisionsalgoritmen och andra uträkningar med uppställning. Algoritmiskt tänkande är inte ett uttalat mål före årskurserna 7–9.

Ett konkret exempel på tidigt algoritmiskt tänkande i årskurserna 1–2 är de enkla s.k. *Bee Bot*-robotar som till exempel ska navigera igenom labyrinter. Lösningen kan då skräddarsys för den specifika labyrint som roboten ska traversera och behöver inte generaliseras för att fungera på en godtycklig labyrint. Då eleverna senare arbetar med grafiska programmeringsmiljöer i de årskurserna 3–6 gör de främst sekvenser av händelser. Denna form av programmering tränar upp grunderna till det algoritmiska tänkandet, och det viktiga är inte att det sker med digitala verktyg då ”ett matrecept, noterna till en komposition, uppställning med division och handarbetsbeskrivningar är algoritmer” (Utbildningsstyrelsen, 2021). Mer intresserade och äldre elever kan ges fördjupningsuppgifter där lösningar generaliseras, vilket är ett nyckelsteg för att själv skapa en algoritm. Till generalisering krävs också villkors- och upprepningsstrukturer.

Algoritmiskt tänkande är en viktig komponent i datalogiskt tänkande. Skillnaden är att då man tänker datalogiskt avgör man dessutom ifall, och hur, ett problem kan överföras till en form som en dator kan lösa (Utbildningsstyrelsen, 2021). Problemlösaren behöver även en god överblick över vilka alternativ det finns, så att hen kan välja det som lämpar sig bäst. Wing (2008), som var en tidig förespråkare till att datalogiskt tänkande skulle tränas upp i skolan, påminner också om vikten av att inte i utbildningen vara för snabb att ersätta mekaniskt räknande med maskinellt. En elev som till exempel löser ekvationer med en applikation kan enligt Wing vaggas in i en falsk tro om att hen förstår hur och varför det fungerar. Det är alltså riskabelt att helt ersätta traditionella metoder med teknologiska lösningar.

2.3 Vad är programmering?

Programmeringsbegreppet var inledningsvis endast reserverat för handlingen som utfördes då man förberedde en uträkning för en dator. Detta gjordes alltid genom att skapa en sekvens av logiska operationer den (datorn) kunde utföra, och av denna orsak måste instruktionerna alltid vara skrivna i datorkod. Denna syn måste däremot vid ett senare skede överges då tillämpningarna slutade vara enbart matematiska uträkningar. Det har förslagits att man antar en bred definition på vad programmering är, kanske

till och med inkluderar alla former av datoranvändning trots att genomsnittsanvändaren inte längre behöver ha någon kunskap om programkod. (Blackwell, 2002) Både Skolverket i Sverige och Utbildningsstyrelsen i Finland har följt denna väldigt breda definition på vad programmering är (Utbildningsstyrelsen, 2021).

Ytterligare ett sätt att närma sig begreppet är att utgå från definitionen på vad ett program är och definiera programmering som skapandet av program. Utöver stegvisa instruktioner, skrivna i kod kan man finna större strukturer inuti koden. Korta program består kanske av bara en algoritm, men de kan kedjas ihop för att låta programmet ta hand om flera uppgifter i tur och ordning. Ofta delar man in kodstycken som återanvänds i *funktioner*. (Skolverket, 2019)

För att program ska vara meningsfulla och ha en inverkan på sin miljö ska de använda sig av information, eller data, från omvärlden, och producera ett resultat. Data kan med fördel (men måste inte) lagras i *variabler* under exekveringen. Om data inte bara läses in då programmet startar, och utdata inte enbart produceras i slutet blir förloppet beroende av omgivningens villkor, till exempel en användare som interagerar med hjälp av mus, tangentbord eller pekskärm. Ett sådant program är interaktivt och programmeraren tvingas förutsäga flera möjliga situationer som kan uppstå till följd av att de utomstående omständigheterna inte alltid är de samma. Villkor och upprepning som möjliggör sådana program är centrala i LPGLU, och nämns specifikt som kunskapskrav för vitsordet 7 i avgångsbetyget (Utbildningsstyrelsen, 2020).

Det finns ingen tydlig gränsdragning till vad som är eller inte är programmering. Man kan till exempel fråga sig ifall en musiker som skriver sin musik digitalt, en webbdesigner som skriver html-kod eller en dataanalytiker som skapar komplexa modeller i kalkylblad kan ses som programmerare. I alla dessa fall kan skapelserna ses som program, då de utgörs av kommandon som en dator senare utför. Automatisering av databehandling innefattar ofta samma steg som ”vanlig” programmering. Ett tydligt tecken på programmering är att man på något sätt överger direkt manipulation av en process. Arbetsinsatsen är abstrakt planerande i stället för direkta reaktioner på processens nuläge (Blackwell, 2002).

2.4 Programmeringens roll i skolmatematiken

Programmering började som en gren av tillämpad matematik som sedan växte för att utdela avkastning inom både ren och tillämpad matematikforskning. De breda tillämpningsmöjligheterna i hela samhället gjorde att den digitala revolutionen skedde fortare än någon kunde ana. Klassrummets utveckling är vanligtvis en reflektion av vardagens. I takt med teknologins utveckling har nya verktyg gjort entré. De har använts parallellt med, eller ersatt äldre metoder. Kulramen och räknestickan har exempelvis ersatts av fickräknaren, som i sin tur har börjat ersättas av smarttelefoner. Passare och linjal som på senare årskurser har konkurrerat med grafräknare får dela uppmärksamheten med datorprogram såsom *Geogebra*. Tillämpandet av diverse verktyg och teknologi för att avlasta arbetsminnet och kunna utföra mera komplexa uppgifter är centralt i ämnet matematik (Martínez, 2021). När en maskin utför stegen man själv definierar och modifierar behöver man inte befatta sig med utförandet, utan kan i stället fokusera på processen och målet. Om målet förskjuts kan kanske enskilda steg i processen bytas ut, och då undviker man även att återgå till ett tomt papper och börja om.

Läroplanen uppmuntrar denna utveckling då eleverna ska ”tillämpa egna eller färdiga datorprogram i matematikstudierna”. Detta antyder att programmeringen i viss mån ska integreras i resten av matematikundervisningen. Däremot saknades det krav på när och hur detta skulle ske, eller med vilka program eller språk, så lärarkåren hade väldigt fria tyglar. Samtidigt kan kommunala planer fastställa en gemensam linje, t.ex. gällande användning av visuella programmeringsmiljöer eller undervisningsrobotar.

Det finns väldigt många olika matematiska digitala verktyg att välja mellan, och det är knappast långsökt att anta att lärare ofta väljer sådana som de själva behärskar bäst. Det är även här som lärarnas egen kunskap och värderingar kan tänkas föra undervisningen i olika riktningar.

2.5 Olika programmeringsmiljöer

Diskussioner om programmering i pedagogiskt syfte leder ofta till frågor om kodning och syntax, vilket språk och vilken plattform (online eller offline-baserad[1]) som används. Det bör däremot påpekas att innan läraren bestämmer sig för plattform, måste hen först bestämma sig för vilken slags miljö som kommer att fungera bäst. Onekligen är kodning centralt för programmering, men det finns alternativ som kan användas för att undvika komplikationer med syntax eller tiden som krävs för att introducera plattformen. Programmering i skolsammanhang handlar om själva problemlösningsprocessen, i vilken det bör beaktas att problemet måste delas upp i mindre delar (Mannila, 2017). Dessa delproblem löses stegvis med givna verktyg, men själva miljön behöver inte nödvändigtvis inkludera användning av någon maskin. Genom att börja med enklare miljöer kan det vara lättare att avgränsa problemet och vilka verktyg som finns tillgängliga. En metaanalys (Hu m.fl., 2021) rapporterar om små till medelstora positiva effekter på studieframgångar då grafisk programmering används i stället för textbaserad. Däremot minskade också nyttan för äldre åldersgrupper i rapporten. Beroende på vilken tid på läsåret och vilken årskurs av 7–9 man undervisar så är målgruppen mellan 12 och 16 år gamla. Det abstrakta tänkandet mognar mycket under den här tiden, vilket också reflekteras av progressionen i den övriga matematikundervisningen. Därför finns det skäl att varken överge miljöer som kan verka ”barnsliga” eller förkasta textbaserade programspråk som för svåra.

2.5.1 Analog miljö

Med analog miljö menas att man tränar det datalogiska tänkandet utan att använda teknologi. I stället kan traditionella redskap som finns i klassrummet användas.

[1] Programmeringen sker traditionellt i en s.k. IDE, Integrated Development Environment, medan särskilt webbaserade övningar ofta sker i en s.k. sandlåda (från engelskans *sandbox*). Plattformen kan också vara spel där karaktären programmeras likt en Bee-Bot eller annat beteende kan modifieras.

Ett praktiskt exempel på hur programmering kan utföras utanför en teknisk miljö, är genom att låta eleverna programmera varandra (Ifous, 2020). Metoden går ut på att elever turas om att skapa sekvenser av instruktioner som sedan en klasskamrat utför. Eleverna som utför instruktionerna ska endast reagera på giltiga kommandon, och fokusera på att utföra dem exakt, vilket även betyder att fel inte rättas under exekveringsskedet. Kamraten som instruerar tvingas då att tillämpa ett algoritmiskt tänkande för att klara av uppgiften. Uppgifter kan exempelvis vara av geometrisk natur, men det finns även potential i att repetera bekanta algoritmer från tidigare årskurser, till exempel beräkningar med uppställning. I en lektionsplan med namnet ”*öva klockan*” (Utbildningsstyrelsen, 2022) ska en elev ge instruktioner åt två klasskamrater, som representerar tim- och minutvisare.

Analog programmering lämpar sig för alla årskurser, eftersom uppgifternas svårighetsgrad kan anpassas för äldre elever. En variant av analog programmering som kan tänkas fungera för äldre elever är att skriva *pseudokod*. Pseudokod brukar användas för att förklara hur man löser ett problem stegvis, men utan att skriva i något specifikt språk. Detta skapar en sorts skiss, eller grovplanering och kan vara ett steg innan man börjar skriva programkod. Det är vitt använt i såväl akademiska kretsar som i industrin just för att pseudokod effektivt förmedlar kodens innehåll till andra människor. I undervisningssyfte har analog programmering utöver dessa ovan nämnda exempel också observerats som stöd för bedömning, till exempel i form av exit tickets, vilket innebär att eleven får besvara någon central fråga om lektionens innehåll på en lapp som lämnas in då eleven går ut. Alternativt kunde det vara en uppgift där eleven använder sig av något nytt kommando i ett kort pseudokodprogram. (Ifous, 2020)

2.5.2 Undervisningsrobotar

Undervisningsrobotar är populära inslag i grundskolan, särskilt i årskurserna 1–6 men de används även i årskurserna 7–9. Olika slags robotar tillsammans med tillhörande applikationer erbjuder en lekfull och nerskalad programmeringsmiljö. Samtidigt kan lekfull eller spelifierad undervisning också vara en nackdel. Elever kommer oftast ihåg

det roliga inslaget från underhållande lektioner i stället för begreppen läraren vill förtydliga (Barton, 2018). Till exempel kan de datalogiska färdigheter som läraren försöker förmedla bli sekundära. Trots det finns det onekligen också en positiv effekt för motivationen att se programkodens konsekvenser bli mera konkreta med hjul, lampor, högtalare och interaktion med omvärlden tack vare sensorer.

Man kan se undervisningsrobotar som en kinetisk undergrupp till programmerbara mikrochip, till vilken produkter så som *Arduino* och *Micro:bit* hör. Undervisningsrobotarna och mikrochipsen för även med sig begränsningar. Det kan till exempel både vara till för- och nackdel att instruktionsuppsättningen är begränsad. De flesta undervisningsrobotarna, till exempel Lego Mindstorms, utnyttjar *blockprogrammering* (se avsnitt 2.5.3) där varje fysisk komponent som ansluts till roboten kan kontrolleras med hjälp av några färdigt existerande kommandon. Elever som på egen hand fördjupat sig i programmering kan anmärka på avsaknaden av funktionalitet i dessa robotar, men får utmaningen att arbeta kring dessa begränsningar. Ofta är det möjligt att med enkla, rudimentära kommandon göra komplicerade saker och det är också vanligt att se alternativet att skapa användardefinierade funktioner.

Till skillnad från helt digitala miljöer, som i så gott som alla fall är gratis att använda, har hårdvaran sitt pris. Det förekommer att införskaffningar görs centraliserat i syfte att ge denna möjlighet åt flera skolor. Samma utrustning lånas sedan ut till olika skolor i tur och ordning under läsårets lopp.

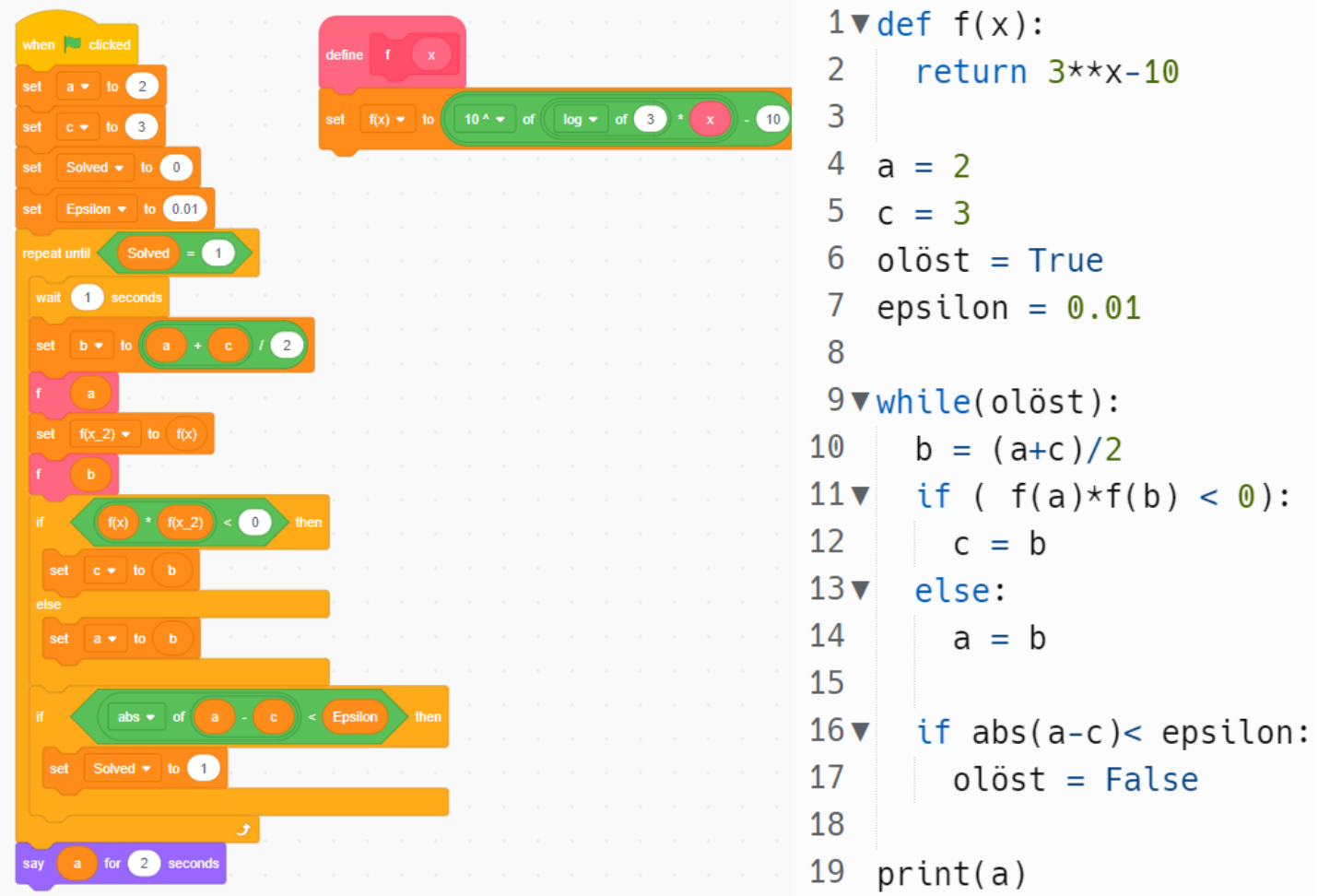
2.5.3 Grafisk programmeringsmiljö

Ett populärt sätt att introducera programmering är att göra det i en sådan miljö som garanterar att eleven inte kan göra syntaxfel. Syntaxfel uppstår i textbaserad kod då kodaren skriver in odefinierade funktioner, variabler eller kommandon som inte är en del av språkets syntax. Oftast är orsaken ett stavfel, något felplacerat tecken eller avsaknaden av ett sådant. Sådan kod exekverar inte alls, upphör att fungera och producerar oftast felmeddelanden som är utmanande att begripa, samt utgör en källa till frustration för nybörjare.

Grafiska programmeringsspråk eller även s.k. blockprogrammering kringgår detta problem genom att begränsa användaren till färdigt existerande kommandon, skrivna på block som pusslas ihop. Kodblocken är sorterade i kategorier, försedda med ikoner och färgkoder. Samtidigt är dessa fördefinierade funktioner ofta ganska kraftfulla kommandon, som i ett annat språk kunde utgöra ett stort delproblem i sig att lösa, innan man kan påbörja den egentliga uppgiften. Ett exempel på detta är funktionen ”*touching?*” i Scratch som är en fullständig implementation av kollisionsdetektering, något som få nybörjare hade kunnat implementera på egen hand.

En metaanalys (Hu m.fl., 2021) indikerar att oavsett ålder har visuella programmeringsmiljöer, däribland Scratch, en positiv effekt på elevernas prestationer. Några förklaringar på blockprogrammeringens fördelar är bland annat den enkla syntaxen och effektiverad tidsanvändning då mindre tid går åt till att reda ut triviala fel. Den kognitiva belastningen blir då lägre och uppmuntrar högre elevfokus, vilket bidrar med ytterligare en förklaring. Ännu en fördel som nämns med blockprogrammering är den visuella responsen som användaren får då koden exekveras, vilket uppfattas som mera konkreta resultat.

Nackdelen är, liksom med undervisningsrobotar, att om läraren vill betona matematiken begränsas möjligheterna av de existerande blocken och funktionerna (se Figur 1). De vanligaste blockprogrammeringsmiljöernas design riktas mot att hjälpa förståelsen av hur till exempel variabler används. Samtidigt tar de ett steg ifrån den matematiknära uppbyggnaden som textbaserade miljöer erbjuder.



Figur 1
Mittpunktsmetoden för att hitta nollställe, exakt samma algoritm implementerad i Scratch och Python. Notera att för att anpassa till de begränsningarna som scratch medför är potensfunktionen skriven med hjälp av tiopotens och logaritm. Booleska variabler skrivs som strängar och användardefinierade funktioner har inte returvariabler. Returvärdet sparas därför i en extern variabel med namnet f(x).

2.5.4 Textbaserad programmeringsmiljö

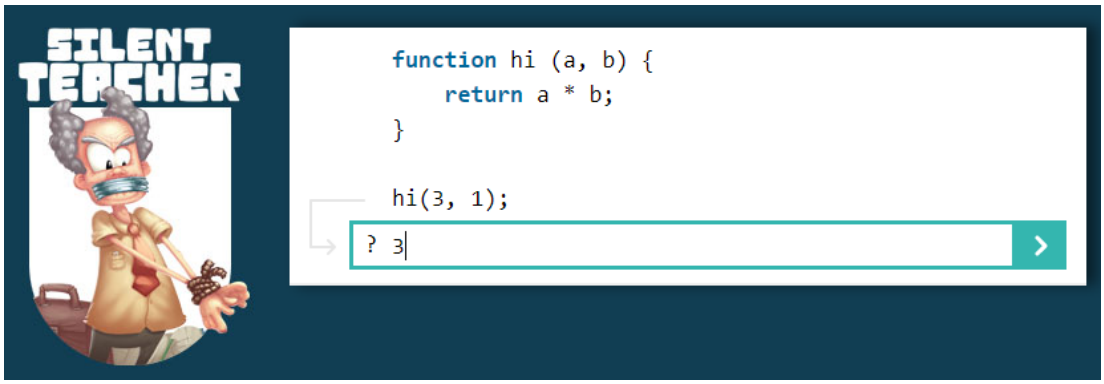
Under årens lopp har programmeringsspråk ständigt utvecklats för att bli mera nybörjarvänliga med hjälp av enklare syntax och en ökad mängd inbyggda funktioner. Ett tidigt språk som utvecklades år 1964 med nybörjare i åtanke var BASIC. En av skaparna (Kurtz, 1997) påpekade i ett öppet brev:" *It turned out that easy-to-learn and use was also a good idea for faculty members, staff members, and everyone else.*". Tack vare sitt låga krav på arbetsminne var BASIC och olika senare tillkomna varianter av det i flitig användning i nästan tre decennier. BASIC lever fortfarande

kvar än idag som ett alternativ för nybörjare, tillgängligt på de flesta stora plattformar genom olika program, som t.ex. Quite Basic, en online-plattform där man kan skriva och exekvera BASIC-program (McCracken, 2014). Så kallade dialekter (vidareutvecklade varianter) av BASIC, TI-BASIC och Casio BASIC, användes för att programmera de grafiska räknarna som användes under perioden innan datorn ersatte papper, penna och räknare i studentskrivningarna. Dessa dialekter var väldigt begränsade (då variabelnamn var endast en bokstav långa så var antalet variabler begränsade till 26) men kunde trots det användas för att automatisera långa uträkningar.

Andra efterkommande högnivåspråk, såsom C, C++, Java och särskilt Python har varit alltmer lämpade för att användas i undervisningssyfte. En av orsakerna är egentligen utvecklarmiljöerna, som har förbättrats med utökad visuell hjälp såsom färger för olika datatyper och variabler. Automatisk ifyllnad och förslag medan man skriver är ett annat tillägg som underlättar belastningen på arbetsminnet. Precis som Quite Basic finns det många webbaserade plattformar för dessa språk, vilket erbjuder en ingång med låg tröskel till programmeringens värld.

Matematiskt sett finns det många möjligheter med textbaserade programspråk. Matematikdelen av syntaxen är ofta likadan då två språk jämförs tack vare att matematiska operatorer nästan är universella. Tack vare detta så kan alla grundskolans uträkningar omformas till programkod.

Steget in i textbaserad programmering kan tas via interaktiva övningar. I bästa fall hittar en lärare något som tangerar det stoff hen undervisar, så att det kan kombineras till en fungerande helhet. Många appar och webbplattformar som tränar textbaserade språk har fördelen att de är självrättande och erbjuder därmed direkt återkoppling.



Figur 2

Webbaserade räkneövningar som både för fram matematiken och introducerar typisk kodsyntax.

Källa: https://silentteacher.toxicode.fr/ .

En annan metod för att jobba med kod som producerar intressanta resultat är att låta elever *remixa,* dvs. vidareutveckla redan fungerande kod. Läraren distribuerar programkod som eleverna ska modifiera för att förändra programmets beteende genom att byta ut några rader, eller lägga till något. Nya läromedel, såsom den finlandsvenska serien Tangent publicerad av Schildts & Söderströms följer till exempel den här strukturen.

2.6 Tidigare undersökningar om lärarnas förhållningssätt till och beredskap att undervisa programmering

Finlandssvenska klasslärare uppgav att programmering var viktigt och spelar en växande roll i det moderna samhällets arbetsmarknad. De tyckte att det borde undervisas i grundskolan, och majoriteten var bekväma med tanken på att själva undervisa det. En stötesten som pekades ut då undersökningen gjordes var att det rådde brist på material. (Kallio-Kujala, 2017)

Undersökningen av Kallio-Kujala pekade även ut att det fanns klasslärare som upplevde programmeringsundervisningen som problematisk. Andelen som var negativt inställd var ändå förhållandevis liten, alltid mindre än 25% oavsett hur

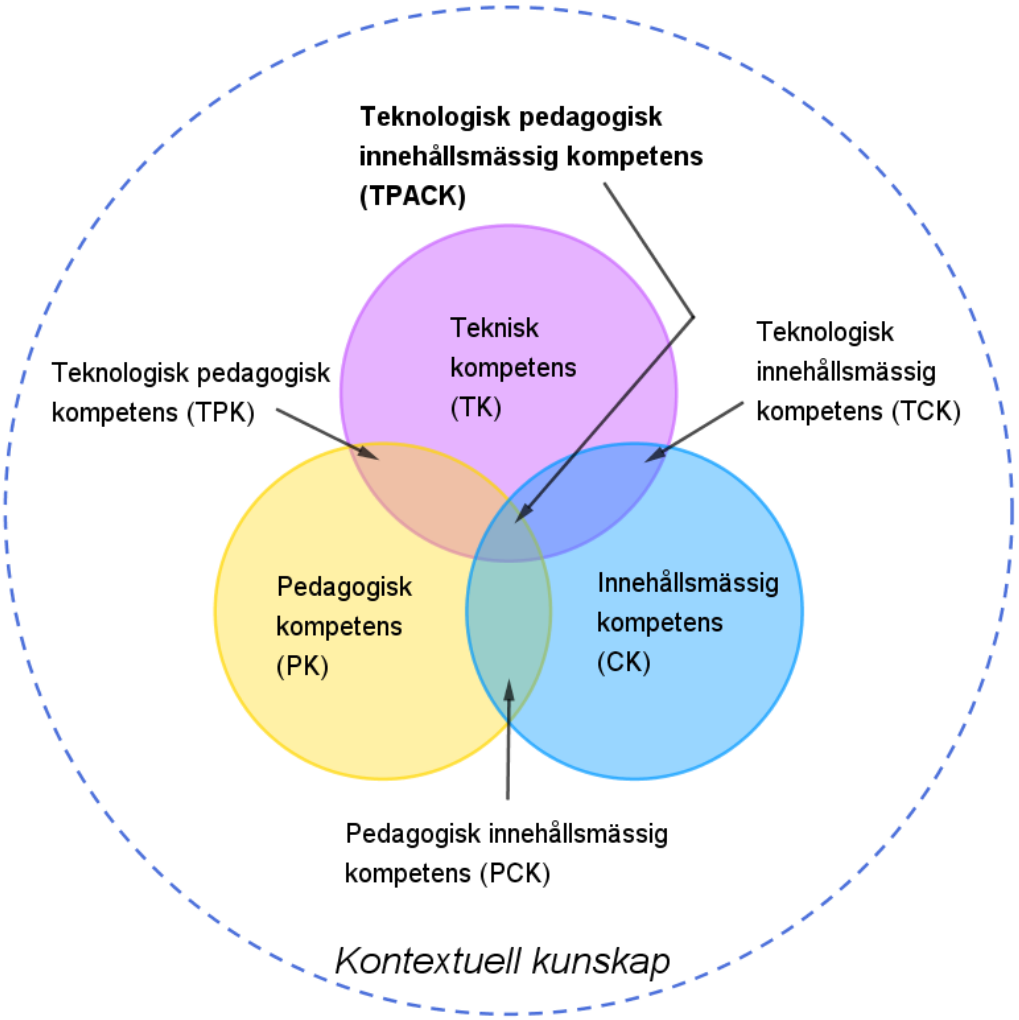
frågorna var vinklade. Då klasslärarna beskrev sina känslor var det oftast med positivt laddade ord, men över hälften beskrev också att de upplevde någon form av osäkerhet. Resultatet hade å andra sidan möjligtvis förskjutits om undersökningen skulle ha gjorts senare efter att flera hade fått erfarenhet av att undervisa programmering.

Det råder ganska stor spridning i hur många timmar finska ämneslärare uppger sig undervisa programmering i årskurserna 7–9. Över en fjärdedel uppgav sig i en enkät gjord 2020 använda 0–5 timmar per läsår, och 18% uppgav över 30 timmar (Kolu, 2020). Enkäten i fråga sändes däremot främst till skolornas IT-ansvariga och IKT-tutorer, vilket syntes genom en synnerligen genomgående positivitet gentemot programmeringsundervisning.

3 Teori

3.1 Studiens övergripande ramverk (TPACK)

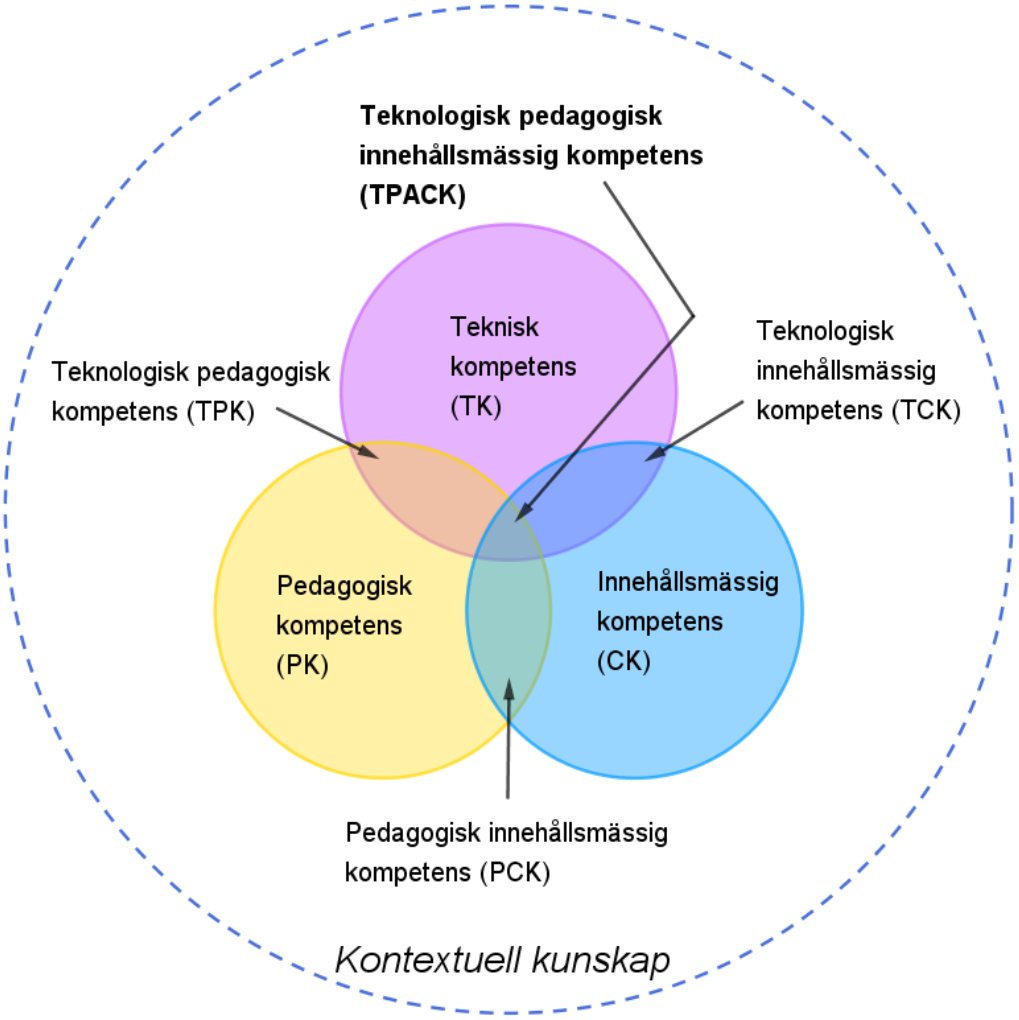
TPACK (även tidigare känd som TPCK) är ett ramverk som beskriver hur lärares förmåga att undervisa effektivt med hjälp av teknologi bygger på tre olika kompetensområden: pedagogisk, teknologisk och innehållsmässig kompetens. TPACK kan ses som en utvidgning av ”Pedagogical Content Knowledge”, eller PCK som konstruerades av Lee Shulman, till att också inkludera den teknologiska kunskapsaspekten. (Koehler & Mishra, 2009)



Figur 3
TPACK-modellen där de olika kompetenserna visualiseras genom områden formade som slutna cirklar. I cirklarnas snitt finns den överlappande kunskap som tillhör flera delområden

Koehler & Mishra motiverar behovet av att införa den tekniska kompetensen i ramverket med att teknologin skapar utmaningar genom att vara mångsidig och föränderlig jämfört med traditionell utrustning. (Koehler & Mishra, 2009) För det första är ett klassrum aldrig fruset i tiden, för maskiner och system byts ständigt ut av varierande orsaker. För det andra är inte läraren alltid låst i ett enda klassrum. Bara inom samma skolbyggnad kan utrustningen variera från klassrum till klassrum, för att inte tala om de skräddarsydda lösningar som kan hittas i övriga utrymmen, så som gymnastiksalar, auditorium, centralradio eller kopieringsrum.

Byter läraren arbetsplats så varierar inte bara utrustningen, för också det övriga sammanhanget ändras också. Till denna övriga kontextuella kunskap representerad av den omgivande cirkeln i



Figur 3 hör lärarens kännedom om övriga saker, till exempel hur

utbildningsanordnaren fungerar på organisationsnivå, ner till kännedom om elever på individnivå. (Mishra, 2019)

3.2 Pedagogisk innehållsmässig kompetens (PCK)

De två delområdena i PCK är pedagogisk och ämnesspecifik kompetens. God undervisning i ett ämne, förklarar Shulman, uppstår i mötet mellan lärarens allmänna kunskap om hur god undervisning ordnas och lärarens kunskap i ämnet som undervisas. Delvis anpassar man undervisningen till undervisningsämnet, och dels anpassas ämnet till undervisningen. Det kan handla om hur ämnet kan delas upp i logiska helheter, i form av välstrukturerade moment, lektioner och sekvenser. En annan förutsättning för att lärandet ska lyckas är att läraren kan förklara och välja mellan att illustrera begrepp med analogier, bilder, exempel eller demonstrationer. Om en representationsform misslyckas kan den överges tillfälligt och ersättas med en annan. Utöver förmågan att dela upp undervisningen i hanterbara bitar behövs en lyhördhet och kunskap om vanliga missföreställningar för att kunna korrigera dem i tid. (Shulman, 1986)

En annan punkt som lyfts fram i samband med PCK är vikten av att knyta band mellan kunskap från olika områden (Koehler & Mishra, 2009). Exempelvis kan det vara bra att kunna belysa både likheter och olikheter i hur variabler och funktioner hanteras och fungerar i matematiken jämfört med datavetenskapliga sammanhang.

$$x = x + 1$$

```
x = x + 1
```

Ekvation 1
En ekvation som saknar lösning eller ett lagrat värde som inkrementeras? En lärare bör kunna redogöra den fundamentala skillnaden som råder mellan ekvationslösning och programkod, för att undvika missförstånd.

3.3 Teknisk kompetens (TK)

Teknisk kompetens i undervisningssamband är inte bara digital kompetens i den form som omnämns i läroplanen. Förutom att det egna arbetet ska kunna utföras smidigt måste läraren kunna fatta beslut om hur och när teknologin ska utnyttjas i undervisningen, men också när den inte tillför något. Med en djupare förståelse av teknologin kan man både lära sig nytt, och ge instruktioner för att hjälpa andra komma vidare med sin problemlösning. Teknologilandskapets föränderliga natur gör dessutom att läraren måste hålla sin kunskap uppdaterad kontinuerligt för att inte använda utdaterade program, metoder eller tjänster inom några år. För att besitta sann teknisk kompetens behöver läraren befinna sig i ständig utveckling och inte sträva efter att uppnå något sluttillstånd, som en evolutionsprocess utan slut. (Koehler & Mishra, 2009)

Att göra en skillnad mellan teknisk kompetens och innehållsmässig kompetens är inte så lätt, då de i programmeringssammanhang överlappar varandra. Programspråk utvecklas till exempel och plattformar förnyas. En lärare som för 10 år sedan lärde sig använda Python 2.7 kan till exempel inte återanvända gammal kod från den tiden i nyare Pythonversioner utan att stöta på flera syntaxfel. Samtidigt är exempelvis felsökning både länkat till kunskapen i ämnet och en teknisk kompetens.

4 Metod

4.1 Val av metod

För att få en bra lägesbild av de Finlandssvenska matematiklärarnas situation behövdes en datainsamlingsmetod som kunden nå många deltagare samtidigt som den måsta vara kompatibel med coronapandemins rådande restriktioner år 2021. För att kunna generalisera några potentiella resultat måste jag även se till att data var insamlat ifrån ett representativt sampel. (Bryman, 2012). Eftersom en elektronisk enkät skulle ge mig en stor räckvidd och möjlighet att samla in data från alla berörda delar av landet blev det mitt val.

En viktig del av en enkätundersökning är att kontakta målgruppen utan att förbise någon delmängd av den och på så vis introducera bias.

4.1.1 Bortfallsfel

De flesta enkätstudier utförs på ett så kallat *bekvämlighetssampel* och innehåller ett sampelfel som uppstår då svar uteblir. Man får endast svar ifrån personer som har varit bekväma med tanken på att besvara frågorna och inte heller avbrutit ifyllningsprocessen innan svaret skickats in. Skälen till att en potentiell respondent låter bli att svara eller inte nås kan vara många. Oftast lider frivilliga undersökningar till någon grad av denna brist. Den grupp som väljer att inte svara representeras inte och man bör då vara försiktig med att generalisera resultaten. (Bryman, 2012)

Ifall jag inte hade valt att kontakta lärarna direkt, och i stället bett rektorerna att vidarebefordra mejlet hade svaren kunnat beaktas som ett så kallat *snöbollssampel*. Ett sådant sampel uppstår då en undergrupp av populationen, eller personer som är sammanlänkade med populationen (i mitt fall rektorerna) ombeds att understöda

forskaren att komma i kontakt med övriga ur populationen. Populationen kan inte representeras kvantitativt av ett sådant sampel, då intressenivå och bekvämligheten hos personen som vidarebefordrar påverkar urvalsprocessen. (Bryman, 2012) Dessutom finns risken att antalet svar minskar drastiskt om denna extra länk i kedjan ofta bryter kontakten till de slutliga mottagarna.

På grund av att det har blivit så lätt att genomföra enkätstudier har det noterats en negativ trend hos svarsprocenten i enkätstudier. Lärare är inget undantag och därför valdes tiden för utförandet med omtanke, för att motverka en potentiell bias mot lärare som upplever sig ha mycket överloppstid. För att motverka respondenternas passivitet har forskare ökat sina insatser genom bland annat utlottning av priser samt flera utskick. Dylika belöningar utlovades inte med denna studie, och därmed kan svarsprocenten ha påverkats. Eftersom svarsprocenten varierar drastiskt beroende på målgruppen finns det ingen allmän rekommendation om hur hög svarsprocent man borde eftersträva. I stället kan man jämföra med liknande studier. (Bryman, 2012)

4.2 Genomförande av studien

Frågorna utformades målgruppen i åtanke, Finlandssvenska ämneslärare i matematik som undervisar i årskurs 7–9. Ett utkast av enkäten testades på en grupp lärare, varvid det konstaterades att trots att frågorna var relevanta och studien intresserade, så var antalet frågor för stort. För att minimera svinnet i form av lärare som inte finner tid att besvara en lång enkät skars en del av frågorna bort, så att den förväntade svarstiden sjönk från över 15 minuter till 10. Frågorna grupperades under följande rubriker och användes för att besvara forskningsfrågorna: *bakgrund och erfarenhet [F1, F2]; stöd, utrustning och material [F3]; fortbildningar [F1]; åsiktsfrågor [F2]* samt *ämneslärargruppens strategi [F2, F3].*

Valet av plattform föll på *Microsoft Forms* eftersom svarsmiljön är användarvänlig oavsett om deltagaren fyller i sina svar med en mobil enhet eller datorns webbläsare. Deltagarna behövde inte logga in med något användarkonto för att svara och behöll anonymitet. De enda metadata som sparades var start- och insändningstiden. Eftersom

Svenskfinland är litet och det således går att identifiera personer utgående från kommun samt födelseår presenteras de i skilda tabeller och som intervall.

För att nå så många ur målgruppen som möjligt valde jag att kontakta dem direkt genom deras arbetsmejladress. Kontakt genom fortbildningar hade inte fångat upp lärare som inte deltar i fortbildningar medan telefonkontakt hade gjort det svårare att hitta kontaktuppgifter och mera tidskrävande, dyrare samt fridstörande. Med tanke på målgruppens storlek (uppskattningsvis n < 200 utgående från storleken på den tabell jag gjorde samt antalet återstående skolor vars information inte räckte till) fanns det ingen orsak att göra en stickprovsundersökning. Kontaktinformationsinsamlingen och utskicket var möjliga att genomföra inom några dagar.

E-postadresserna till lärarna fanns oftast tillgängliga genom skolornas officiella webbplatser, men i några fall då den informationen inte fanns tillgänglig kontaktades i stället skolans rektor. I andra fall gick adressen att gissa sig fram till med hjälp av pålitlig information från tredje part, till exempel namn som dök upp i artiklar där lärare presenterades med namn och arbetsplats. Eftersom felaktiga mejladresser automatiskt besvaras med ett felmeddelande och adresserna alltid genereras utgående från för- och efternamn gick det att pröva sig fram tills meddelandet levererades. I de två fallen som detta skedde ombads även mottagaren vidarebefordra mejlet till övriga inom kollegiet, och således nå dem, något som å andra sidan gjorde att det blev omöjligt att få en exakt svarsprocent (se nästa avsnitt).

För att minimera risken att spärras av spamfilter eller liknande säkerhetsåtgärder sändes inte alla meddelanden ut samtidigt, utan spreds ut över loppet av flera dagar. De första sändes den 30.5.2021 och de sista 3.6.2021. Förhoppningen var att då detta var den sista skolveckan i terminen skulle vitsord redan vara fastställda, och att lärarna då skulle finna tillräckligt mycket ledig tid att besvara formuläret. Majoriteten av svaren (41) är registrerade före 8.6.2021, inom en vecka efter att det sista sändes. Ytterligare några svar (8) dök upp under sommarlovet. Det sista svaret anlände 3.8.2021. Risken med att sända ut formuläret så sent i terminen var att en del mottagare potentiellt aldrig öppnade sina inkorgar under tidsperioden. Således uppstod en bias i urvalet med tanke på att personer som antingen ställt in aviseringar eller kontrollerar sin inkorg oftare med större sannolikhet upptäckte meddelandet.

Svaren exporterades som en exceltabell, och undersöktes sedan i *Excel*. I fritt formulerade svar märktes nyckelord ut med färgkoder för att underlätta analysen. Öppna svar ifrån olika frågor kunde innehålla flera olika argument, och de vanliga kvantiserades i frekvenstabeller. Korrelation mellan sådana variabler som kunde ha någon tänkbar koppling undersöktes genom att svaren i flervalsfrågorna konverterades till tal. En fråga med fyra olika rangordnade svarsalternativ överfördes till värdemängden [0,1,2,3], medan ett ja/nej alternativ fick värdemängden [0,1]. Excel beräknar korrelationskoefficienten med hjälp av följande formel, även känd som Pearsons korrelationskoefficient:

$$Correl(X,Y) = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sqrt{\sum (x-\bar{x})^2 \sum (y-\bar{y})^2}}$$

Ekvation 2
Formeln för korrelationskoefficient, där $\bar{x}$ och $\bar{y}$ är medeltalet av en kolumn medan x och y är värden i kolumnen
Källa: https://support.microsoft.com/en-us/office/correl-function

4.2.1 Reliabilitet och etiska överväganden

Svaren var 49 till antalet. Eftersom respondenterna inte ombads verifiera sin identitet, och då det inte med säkerhet går att avgöra hur ofta en mottagare vidarebefordrade mejlet till sina kolleger, går det inte att få en exakt svarsprocent. En tabell med all information upprätthölls under processens gång och innehöll till slut 170 namn och mejladresser. Av dessa kunde minst 9 inte nås, antingen på grund av att personen pensionerats eller bytt arbetsplats och mejlen stängts. Om mottagaren slutat använda adressen eller temporärt varit tjänstledig, men inkorgen fortfarande funnits kvar, så har det inte funnits någon orsak för den mottagande e-postservern att skicka felmeddelande. Trots att även nyligen pensionerade lärare mycket väl hade kunnat ha värdefulla synpunkter och erfarenhet av programmeringsundervisning, valde jag att inte uppsöka dessa med alternativa metoder. Med uppskattningen att högst 161 har

nåtts av meddelandet (antalet mejl som inte returnerades pga. felaktig adress) ligger svarsprocenten på minst 30 %.

Jämförelsevis hade en liknande enkätundersökning (Kolu, 2020) en svarsprocent på 25,7% då målgruppen var IT-ansvariga och IKT-tutorer i årskurserna 7–9. Kallio-Kujalas enkät riktad till klasslärare fick ett mycket högre antal svar (91) men sändes också ut till 190 skolor, dvs. ungefär lika många skolor som det finns matematiklärare i Svenskfinland, så även i det fallet bedömer jag att svarsprocenten var lägre.

I mejlet som sändes ut försäkrades deltagarna om att inga uppgifter de uppger kan användas för att identifiera dem. Detta är viktigt för att tillåta så ärliga svar som möjligt och ge alla deltagare en kanal där de kan uttrycka precis vad de anser om något. Dessutom sparades inte namn alls då enkäten gick att fylla i utan identifiering. Deltagarna informerades om att enkäten bland annat skulle innehålla frågor om deras erfarenheter av att undervisa programmering i årskurserna 7–9. Informationen som har samlats in används endast till denna avhandling och allt som inte är publicerat här kommer att raderas. Slutligen har alla lärare deltagit av sin egen vilja, då deltagarna är i vuxen ålder och inte har utsatts för någon form av påtryckning till att svara. Således är de olika kraven för god forskningsetik uppfyllda. (Stukát, 2011)

Eftersom Svenskfinland är litet publiceras inte deltagarantalet på kommunnivå. För att visa att respondenter från hela målgruppen är inkluderade, och således håller en hög reliabilitet, nöjer jag mig med att publicera den informationen på landskapsnivå.

4.3 Respondenterna

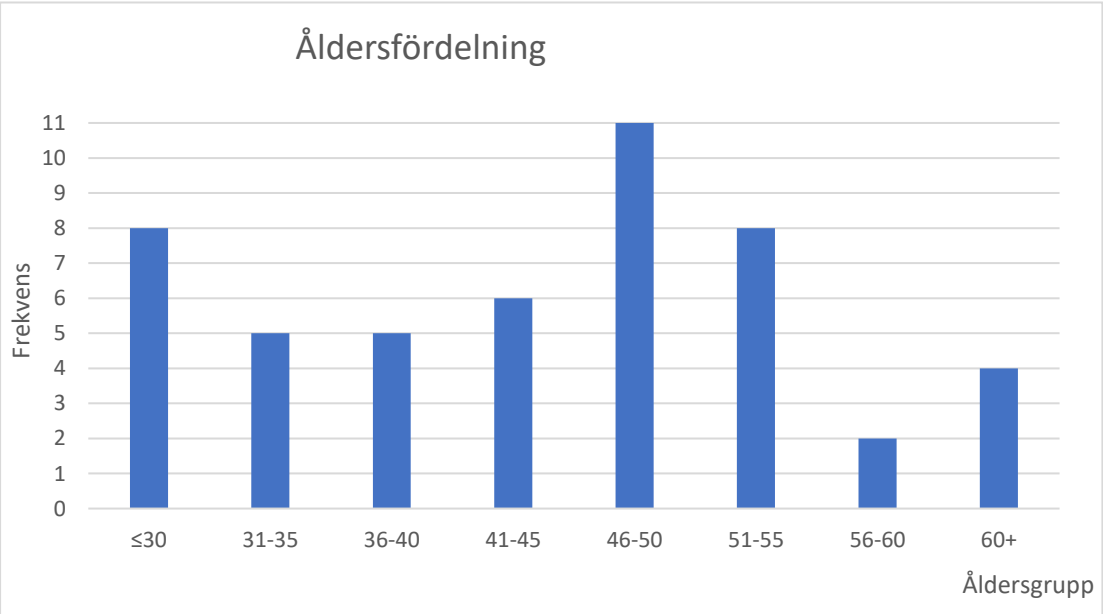
Tabell 1 visar antalet lärare vars kontaktuppgifter fanns tillgängliga landskapsvis, sammanställt baserat på vilken kommun respondenten uppgav sig vara anställd. Undersökningens målgrupp är hela Svenskfinlands matematiklärarkår i årskurserna 7–9, men det är det värt att notera att respons från några kommuner har uteblivit både medelstora och små. Samtidigt hade det varit svårt att uppnå fullständig täckning, då sannolikheten att någon i varje kommun med bara en handfull lärare svarar är liten.

Likaledes är en hög svarsprocent från en liten kommun inte heller oväntat. Skillnaden mellan stora och små kommuner är ytterst liten. Svarsprocenten i kommuner med 5 eller färre kontaktade lärare är 31% (19 av 61) medan andelen svarande i större kommuner (med 6 eller flera kontaktade) är 30% (30 av 100).

Tabell 1
Antal respondenter från olika landskap med svenskspråkiga grundskolor 7-9

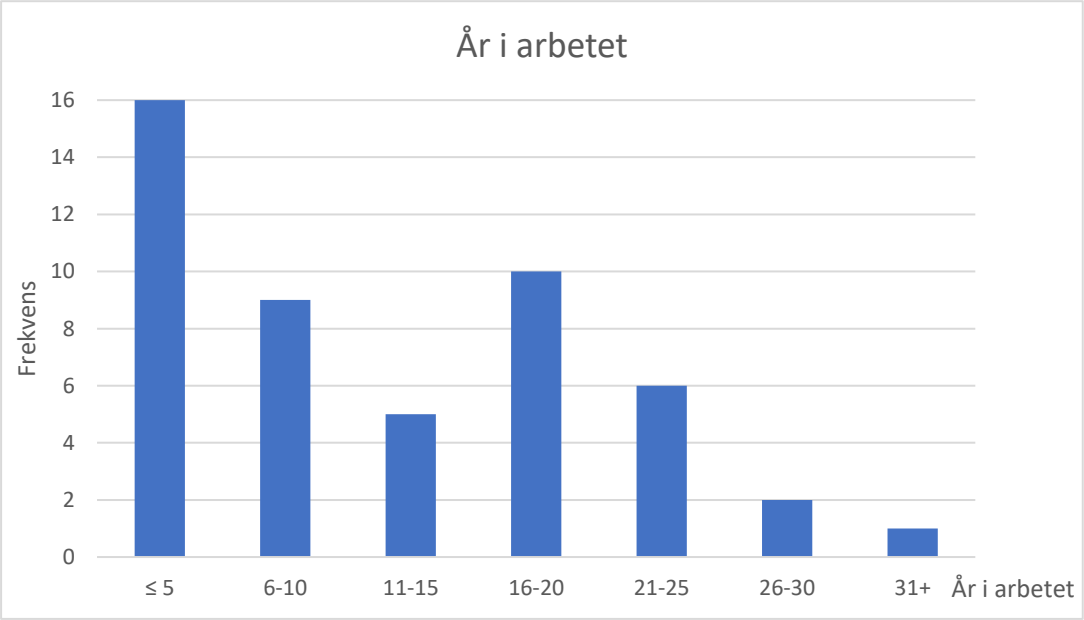
Landskap	Antal svar	Antal kontaktade	Svarsprocent
Nyland	24	80	30,0%
Egentliga Finland	6	19	31,6%
Österbotten	16	49	32,7%
Övriga (Satakunta, Birkaland, Norra Österbotten, Kymmenedalen)	3	13	23,1%
Totalt	49	161	30,4 %

Åldersfördelningen (Figur 4) är uppskattad baserat på uppgivna födelseårtal och stämmer endast vid årsskiftet 2021–2022. Trots det ges en bra översikt över respondenternas åldrar. 32,6% av respondenterna var under 40 år, den största andelen på 36,7% mellan 40 och 49 samt 30,6% i åldern 50 eller äldre. I jämförelse med den



senaste undersökningen om lärare och rektorer i Finland (Utbildningsstyrelsen, 2019) som uppdateras vart tredje år finns det inga större avvikelser då motsvarande procenttal var 30,8% 33,5% och respektive 35,7%.

Det går att urskilja ett aningen högre deltagarantal i denna undersökning hos den yngsta åldersgruppen som var 30 eller yngre, samt inom intervallet 46 – 55. På samma sätt upptäcks ett högt deltagarantal för lärare med färre än sex år av yrkeserfarenhet (Figur 5) samt en höjning i frekvensen för de som passerat 15 år i arbetet (under antagandet att frekvensen är strängt avtagande på grund av karriärbyten).



Figur 5
Yrkeserfarenheten som antal år I arbetet

5 Resultat

5.1 Kunskapsnivå och självsäkerhet

I gruppen lärare som är yngre än medeltalet (44 år) har 12 av 22 (55%) mellan 0 och 5 studiepoäng (eller 3v) i programmering, IT eller datavetenskap. Motsvarande andel för åldrarna 44 eller äldre är 15 av 27 (56%), så det går inte att urskilja någon förändring i mängden akademiskt förvärvad kunskap genom denna grova indelning av åldersgrupperna.

Nästan hälften av alla respondenter uppger att de till viss mån eller helt och hållet är självlärda. Dessa självlärda beskrev oftast sina kunskaper som tillräckliga eller rigida. De som ansåg sig ha lösryckt kunskap uppgav oftast att den kunskapen var förvärvad genom fortbildning (11 av 16). Respondenterna utvärderade sin kunskap och Tabell 2 visar svaren tillsammans med statistik över deras fortbildningsdeltagande.

Tabell 2
Respondenternas självbedömning vad beträffar kunskaper I programmering

Påstående	Antal	Har deltagit i fortbildning	Antal
Jag känner att jag inte alls kan programmera	3	Ja	1
		Nej	2
Jag har lösryckt kunskap, men känner att det inte räcker	16	Ja	13
		Nej	3
Mina kunskaper täcker behoven i åk 7–9	25	Ja	17
		Nej	8
Jag har en rigid kunskap om programmering och kan även handleda elever som vill fördjupa sig.	5	Ja	3
		Nej	2

Tabellen ovan avslöjar att bara 1 av 3 som inte alls känner att de kan programmera har deltagit i fortbildning. Av respondentens korta beskrivning att döma var den fortbildningen i fråga inte så relevant. Å andra sidan är antalet respondenter i kategorin som inte känner att de kan programmera för lågt för att några allmänna slutsatser ska kunna dras.

De respondenter som uppgav sig ha deltagit i fortbildning(ar) ombads beskriva sina erfarenheter med fria formuleringar och välja ett alternativ som de höll med om mest. De fria formuleringarna vittnade om en stor mångfald mellan fortbildningarna, *Lego Mindstorms, Arduino, Micro:Bit, Scratch, Python* är bara några verktyg/miljöer som nämndes. Kurserna hade både ordnats via nätet och genom närtillfällen. Flera av de som uppgav sig vara nöjda med fortbildningen hade deltagit i flera, och uppgav det genom att räkna upp dem i sina svar. Tabell 3 visar att av de som har deltagit i fortbildningar är ungefär två tredjedelar nöjda med resultaten.

Tabell 3
Hur lärorik anser Du att fortbildningen var för dig?

Jag fick inte ut något av den	2	5,9%
Inte så givande	10	29,4%
Ganska givande	15	44,1%
Mycket givande	7	20,6%
Inte deltagit	15	-

5.2 Lärarnas attityd till att undervisa programmering

Många av frågorna som ställdes i enkäten tangerar förhållningssättet till att undervisa programmering. Några av dessa frågor besvarades med fri formulering som har tolkats

och kvantifierats, medan andra frågor direkt presenterade några alternativ på en skala. Möjligheten gavs att efteråt motivera påståenden i fråga 18–20 i fritt formulerad text, där flera argument både för och emot programmering i matematikundervisning framlades av respondenterna.

Tre av lärarna hade ännu inte undervisat någon lektion i programmering, antingen för att en kollega hade skött det, eller av annan okänd orsak. Orsaken kan däremot inte vara att lärarna var på sitt första år och hade planerat in programmering senare, eftersom enkäten besvarades under sen vårtermin eller tidigt sommarlov.

Frågor som anknyter till förhållningssätt.

7. Hurdana har Dina erfarenheter kring undervisning i programmering varit? Jämför t.ex. med normal matematikundervisning.

   Fri formulering

17. Hur viktigt anser du det är att programmering undervisas för alla, och inte bara erbjuds som tillval?
    (1) Mycket viktigt
    (2) Viktigt
    (3) Inte så viktigt
    (4) Inte alls viktigt
18. Vilket påstående beskriver din åsikt bäst?
    (1) Programmering bör undervisas som en del av matematiken.
    (2) Har ingen åsikt i frågan.
    (3) Programmering bör undervisas som ett skilt ämne.
19. Gynnas matematikundervisningen av att innehålla programmering?
    (1) Ja
    (2) Nej
    (3) Vet inte
20. Gynnas programmeringen av att vara en del av matematiken?
    (1) Ja
    (2) Nej
    (3) Vet inte

21. Valfri motivering kring de tre föregående frågorna

    Fri formulering

”Svårt” är ett av de ord som används främst i svaren på frågan om hurudana erfarenheter lärarna har av att undervisa programmering. Svårigheterna har uppstått redan i planeringsskedet, tex. om de har försökt väva in övrig matematik i lektionsplanen. Materialbrist och otillräcklig erfarenhet pekas ut som orsaker till att det inte är lätt att hitta rätt svårighetsgrad. En annan orsak till att respondenterna inte har lyckats att anpassa nivån, är att elevernas förkunskaper och erfarenhet har varit mera utspridd än i vanlig matematikundervisning. Några kämpar med att motivera eleverna medan andra har svårt med att hinna stöda alla elever som behöver hjälp under lektionerna. Så här skrev respondent nummer 49:

> *Det görs lite när det finns tid, har inte varit lika prioriterat som övrig matematikundervisning.*
>
> *Själva undervisningen i programmering har varit ganska kaosartad: gammal teknikutrustning till förfogande, svårt att få elevernas uppmärksamhet då man försöker visa något/lära ut, en aning knepigt att få eleverna motiverade till varför det är viktigt.*

Följande frekvenstabell räknar antalet respondenter som i antingen fråga 7 eller 21 nämner någon av dessa olika känslor, erfarenheter, eller synonymer till dem. Alla 49 respondenter hade skrivit minst ett ord i någondera av textrutorna. Till positiv erfarenhet har jag räknat beskrivningar av lyckade lektioner, eller situationer beskrivna med positivt laddade ord, såsom ”intressant”, ”roligt”, ”motiverande” eller ”bra omväxling”. Likväl förekom negativt laddade ord, och jag väljer här att dela upp beskrivningarna av mindre lyckade lektioner i tre grupper: ”nivåskillnader”, ”svårt” och ”tekniska utmaningar” (som ganska naturligt faller in i TPACK-teorins tre olika delområden PK, CK och TK[2]). En del svar placerades i flera är en grupp.

[2] En likadan indelning hade varit intressant att göra för de positiva erfarenheterna, men det går tyvärr inte att göra i analysskedet eftersom respondenterna inte spontant väljer att prata om t.ex. hur bra tekniken har fungerat.

Tabell 4
Frekvenstabell, hur många av respondenter hade valt att skriva om en sådan erfarenhet eller förmedlade en sådan känsla. Några svar har innehållit flera sorters erfarenheter

Känsla eller erfarenhet	Antal	Andel
Positiv erfarenhet	16	32,7%
Stora nivåskillnader mellan elever	20	40,8%
Svårt/utmanande	18	36,7%
Tekniska utmaningar	8	16,3%
Uttrycker osäkerhet kring antalet undervisningstimmar	17	34,7%

Ett anmärkningsvärt fenomen är att över en tredjedel av respondenterna i fråga 7 eller 21 uttrycker att det inte finns självklara krav på hur många lektioner som ska användas till programmering (n = 17). Respondenten har då i sitt svar antytt att hen inte undervisar fler än enstaka lektioner (betonat i citatet nedan). Övriga orsaker som anges är bland annat att den egna kunskapen inte har varit tillräcklig för fördjupning, eller att tiden har varit så knapp att avsnittet har bortprioriterats. Respondent nummer 7 har ett svar som sammanfattar de olika tankarna som framlagts:

> *Programmering och matematik är två skilda ämnen. De skall inte blandas ihop.* ***I detta läge kan man själv välja att programmera väldigt litet och inte ge bort så många lektioner av matematiken för programmering.*** *Om programmering var ett eget ämne kunde lärare som studerat ämnet undervisa i det och en viss tid (t.ex. 1 åvt) skulle vara dedikerad för programmering.*

På fråga nummer 17 om hur viktigt det är att programmering undervisas för alla har respondenterna svarat på följande vis:

Tabell 5
Hur viktigt anser du det är att programmering undervisas för alla, och inte bara erbjuds som tillval?

Mycket viktigt	7	14,3%
Viktigt	20	40,8%

Inte så viktigt	20	40,8%
Inte alls viktigt	2	4,1%

Svaren avslöjar att oavsett om programmering skulle vara en del av matematiken eller inte, så är det en betydande andel som inte ser någon större poäng med att det undervisas för alla. Detta står i kontrast mot de svar som Kallio-Kujala (2017) erhöll från klasslärare som beskrev sin förhållning till att undervisa programmering. Då var 74,5% positivt ställda till tanken att undervisa det och 83,5% tyckte att det var viktigt att lära ut.

Fråga 18–20 besvarades på följande sätt:

Tabell 6
Vilket påstående beskriver din åsikt bäst?

Programmering bör undervisas som en del av matematiken	13	26,5%
Programmering bör undervisas som ett skilt ämne	21	42,9%
Har ingen åsikt i frågan	15	30,6%

Tabell 7
Gynnas matematikundervisningen av att innehålla programmering?

Ja	20	40,8%
Nej	15	30,6%
Vet inte	14	28,6%

Tabell 8
Gynnas programmeringen av att vara en del av matematiken?

Ja	17	34,7%
Nej	12	24,5%
Vet inte	20	40,8%

Både lärare som upplever sig vara starka eller svaga i sina programmeringskunskaper (enkätfråga 5) påpekar i den valfria motiveringen att det inte kan garanteras att elever med olika lärare erbjuds samma kvalitets undervisning då inte alla lärare har samma goda utgångsläge jämfört med matematikens övriga temaområden. Något som också förespråkas är att inte matematikämnet ensamt ska axla ansvaret för att lära ut programmering, utan att det borde tas upp också i övriga ämnen. Främst fysik och slöjd pekas ut, men även övriga ämnen räknas upp.

5.3 Undervisningsmiljö och läromedel

Trots att läroplanen inte nämner hurudan miljö man ska arbeta i, eller vilka programspråk som borde användas fanns det lärare som nämnde att de hade fått uppfattningen av att de *måste* arbeta textbaserat. Andra funderade på ifall övriga program, till exempel kalkylark och *Geogebra*, kunde räknas som programmering. Onekligen finns det väldigt många alternativ att välja mellan, något som gör att elever kan ha vitt skilda erfarenheter av programmering, och därför ställdes frågan ifall lärarna anser att programspråk eller teman borde styras av läroplanen.

Tabell 9
Borde läroplanen styra vilket programspråk och vilka teman som tas upp?

Ja	15	30,6%
Nej	32	65,3%
Vet inte	2	4,1%

Förespråkarna av ökad styrning påpekar bland annat att det hade gjort fortbildningssatsningar och läroböcker mera fokuserade på ett enda språk, och därmed kanske höjt kvalitet och kvantitet. De som ställer sig skeptiska till mera styrning kommenterar däremot att de är nöjda med den rådande flexibiliteten. Ett annat

motargument var att ett utvalt språk kan föråldras eftersom läroplanen inte byts ut så ofta.

Hur mycket används de olika miljöerna då? I fråga 24–26 ombads respondenten uppskatta hur stor andel av programmeringsundervisningen som sker i icke-digital, grafisk (blockbaserad) och textbaserad miljö. Tyvärr måste en betydande andel av svaren, 15 av 49, måste filtreras bort[3]. Svar uteslöts eftersom summan av de uppskattade procenttalen inte var 100. Av de svar som återstår framgår att cirka 4,1% av programmeringsundervisningen sker utan digital miljö, 44,3% i visuella eller blockbaserade och återstående 51,6% i textbaserade miljöer.

Användning av Student's t-test är motiverat då standardavvikelserna (och därmed också varianserna) mellan grupperna är av samma storlek, grupperna är oberoende och svaren inom grupperna ungefärligen är normalfördelade runt medelvärdet. Det går det att hitta en stark länk mellan mängden textbaserad miljö och vad respondenterna har uppgett om sitt läromedel. Korrelationen mellan det faktum att lärarens lärobok innehåller programmering (värdet 0 eller 1) och andelen tid som spenderas på textbaserad miljö (0 till 100) är stark, 0,48. Det visar sig att lärare vars läroböcker innehåller programmeringsavsnitt ägnas i medeltal 65,6% av programmeringsundervisningen åt textbaserad miljö, medan de som inte har ny lärobok endast uppger sig spendera 32,5% i medeltal. Ett tvåsidigt t-test mellan gruppen som inte hade lärobok (n=16, m=32,5%, sa 34,3), och gruppen som hade lärobok (n=15, m=65,6%, sa 30,9) resulterade i ett t(29)-värde på 2,826 med p-värdet 0,0042. Testet gav ett högt t-värde, som indikerar att det finns en skillnad mellan de två grupperna, tillsammans med det låga p-värdet ($p < 0,05$), som beskriver sannolikheten att resultatet har uppstått av en slump är väldigt liten. Det går att dra en slutsats om att läroboken antingen styr lärarna, eller att lärare som är intresserade av att undervisa mera textbaserat oftare har valt att byta till ett sådant läromedel. (Rice, 2007)

Detta indikerar att de uppdaterade läroböckerna motiverar lärarna att övergå till mycket mera textbaserad än grafisk programmering. Alternativt har lärare som önskar undervisa mycket textbaserat valt att byta till nyare bokserier.

[3] Resultaten förändras inte heller märkbart även om bortfiltrerade svar inkluderas. Andelarna blir i så fall 7,4%, 44,8% samt 47,8%

Det märks att övergången till läromedel som innehåller avsnitt med programmering har framskridit långsamt. Över hälften av respondenterna (Tabell 10 Ifall läroboken Du använder dig av har innehållit material / avsnitt om programmering, har detta varit användbart?) uppgav att läroboken de använder sig av inte innehåller programmering. Trots detta är det den gruppen som är mest nöjd med sitt existerande material.

Tabell 10
Ifall läroboken Du använder dig av har innehållit material / avsnitt om programmering, har detta varit användbart?

Är lärobokens material användbart?	Antal svar	Svarsandel	Har Du all utrustning och allt material som krävs?	Antal	Andel
Ja	13	26,50 %	Ja	9	69,2%
			Nej	4	31,8%
Nej	8	16,30 %	Ja	5	62,5%
			Nej	3	37,5%
Läroboken innehåller inte programmering	28	57,10 %	Ja	22	78,6%
			Nej 6	6	21,4%

5.4 Övriga resultat

I detta avsnitt presenteras data som delvis tangerar forskningsfråga 1 och 3. Dessa aspekter kan utgöra värdefulla komponenter för god programmeringsundervisning. Jag var intresserad av om lärarna ansåg sig ha goda tekniska förutsättningar. Jag är också intresserad av hur mycket kollegialt samarbete som förekommit samt vilka strategier man har utarbetat.

5.4.1 Tekniska utmaningar och materialbrister

Ett kritiskt moment i undervisningen, som lärare dessvärre inte har så mycket kontroll över att styra, är utrustningen i klassen. Valet av utrustning görs ofta av företag som kommunen valt att samarbeta med. Förutom att en lärare ska behärska klassrummets utrustning och skolans digitala infrastruktur, är dessutom läraren elevernas primära stödkälla vid tekniska problem under lektionstid. Om eleverna inte använder samma hårdvara som läraren, vilket enkäten avslöjar ofta är fallet, uppstår ofta ytterligare utmaningar. Till exempel kan datorer av annan modell och märke än lärarens ha andra tangentbord, operativsystem och olika versioner av samma programvara. Ibland har eleverna inte ens tillgång till en traditionell dator med fysiska tangentbord, utan en pekplatta av något slag.

Då respondenterna skrev om sina erfarenheter var det endast åtta svar som antydde att tekniken hade gett upphov till problem i undervisningen. På basen av det går det att dra slutsatsen att matematiklärarnas tekniska kompetensnivå (TK) och kvaliteten på deras utrustning är så pass hög att detta oftast inte utgör ett hinder i undervisningen.

Dock avslöjar svaren på frågorna 11, 12 och 13 att de som upplever att det råder materialbrist inte tycker att eleverna har tillgång till ändamålsenlig hårdvara. Datasalar och klassuppsättningar av bärbara datorer är ibland bokade då läraren önskar använda dem, eller räcker inte till så att alla elever kan arbeta på en egen enhet. Några påpekar att den önskade programmeringsmiljön inte går att installera utan administratörrättigheter i en skola där eleverna har personliga lånedatorer. Trots att hårdvaran är tillgänglig kan det till exempel krävas att läraren tillsammans med skolans IT-ansvariga reder ut hur de kan gå till väga för att få tillgång till önskad programvara.

5.4.2 Kollegialt samarbete och strategier

En klar majoritet, 37 av de 49 respondenterna (76%) har uppgett sig få stöd av sina kolleger till någon grad. Bara sju ansåg att de inte hade fått stöd alls och de återstående

fem uppgav att de inte har behövt hjälp. Stödet uppstår ofta i planeringsskedet om ämneskolleger samplanerar periodens eller läsårets tidsallokering. I det skedet finns det också tid för att fundera på ifall de vill använda programmering som ett verktyg då andra teman behandlas, eller om det ska behandlas avsides som ett skilt, isolerat kapitel. Några gör upp gemensamma lärstigar för att få till stånd en progression genom årskurserna, och andra passar på att införa praktisk användning i fysikundervisningen.

Bara skilt från övrig matematik	15	30,6%
I slutet av period, eller ifall överloppstid uppstår	8	16,3%
Först skilt och integreras senare i övrig undervisning	3	6,1%
Bara integrerat i undervisningen	11	22,4%
Ingen samplanering	8	16,3%
Har inte undervisat det / tomt svar	4	8,2%

De svar som har kategoriserats som ”ingen samplanering” innebär att alla i kollegiet behandlar det på eget sätt, då de finner det lämpligt.

Tabell 11
Berätta kort om Er strategi kring programmeringen. Tas det t.ex. upp som ett skilt tema eller integreras det med resten av undervisningen?

6 Slutsatser och diskussion

6.1 Kunskap

Denna undersökning använder sig av data från matematiklärare som undervisar i årskurserna 7–9. Av den orsaken bör det påpekas att respondenterna kan ha varierande bakgrund. Alla respondenter har inte nödvändigtvis behörighet, och matematik behöver inte ha varit deras huvudämne. Andra har kanske gått via klasslärarutbildningen och skaffat ämnesbehörighet därifrån. För tillfället råder en situation där den intresserade erbjuds många möjligheter att utveckla sitt programmeringskunnande, medan det inte ställs krav på den ointresserade.

Min förhandsuppfattning var att yngre lärare skulle vara bättre förberedda för uppdraget att undervisa programmering. En av de äldre respondenterna reflekterar denna förhandsuppfattning:

> *Tycker att det är fruktansvärt jobbigt då man helt utan studier i ämnet och erfarenhet bara "tvingades" till det. Jag hör till den generationen som helt undgått ämnet kodning.*

Generationen som helt har undgått ämnet är förvånansvärt stor. I programmeringsavseende har antagligen lärarutbildningarna inte förändrats märkbart. Enkätsvaren avslöjar varken en ökning eller sänkning i vare sig upplevda kunskaper eller antalet avlagda kurser i datavetenskap och programmering. För att komma till denna slutsats har den yngre halvan av respondenterna jämförts med de äldre.

Orsaken till detta är antagligen följande: programmering är än så länge inte ett officiellt skolämne, medan fysik eller kemi ofta paras ihop med matematiklärartjänster. Därför finns det inte heller mycket utrymme för att möblera om i studieplanerna som utformas

på basen av skolornas behov. Blivande lärare som önskar ha behörighet i andra naturvetenskapliga ämnen har helt enkelt inte plats för flera kurser. Å andra sidan, om programmering är en del av skolmatematiken, räcker då kunskaperna från en matematikexamen till för att förverkliga läroplanens mål?

Man kan kanske säga att det i denna aspekt finns en klyfta i kopplingen mellan universitets- och grundskolematematiken så länge som programmering inte får status som ett skilt ämne. En möjlig lösning hade varit att låta grundkurser i programmering räknas till godo som matematikkurser, åtminstone för lärarstuderande. I liknande anda undervisas redan renodlad matematik i datavetenskapens logikkurser.

I matematikkurserna används förvisso diverse programvara som jag skulle klassa som *slutanvändarprogrammering,* se 2.3 (Blackwell, 2002), och visst bidrar det till att utveckla studerandenas algoritmiska tänkande. Däremot är det inte i min mening tillräckligt för att bilda den breda kunskapsbas finska ämneslärare är vana att stöda sig mot, och denna osäkerhet visar sig då lärarna utvärderar sina egna kunskaper.

Nästan 40% av respondenterna har uppgett att de inte känner att deras kunskaper räcker till för att undervisa årskurs 7–9. Av dessa har en stor andel deltagit i fortbildningar. De flesta har också fått någon form av stöd ifrån kollegiet, till exempel samplanering, materialdelning, eller kunskap de har plockat upp under årens lopp. En lärare ur denna grupp räknar upp sina ansträngningar:

> *”Några få kurser på universitetet, en fortbildning. Jag skulle gärna fördjupa mej mera, men tiden har inte funnits. Det känns också tokigt att vara tvungen att göra det på egen tid (dvs kvällar och helger).”*

Samtidigt beskriver denna osäkra grupp inte lika ofta sina kunskaper som *självförvärvade* som den övriga 60%-en, gruppen som anser sina kunskaper räcka till för grundskolans behov. Det är inte överraskande att många lär sig programmera på egen hand. På motsvarande sätt träder en del mjukvaruutvecklare in i arbetslivet på basen av sina självförvärvade färdigheter, utan att skaffa en examen. Enkätsvaren antyder att gruppen med högre självvärdering överskrider antalet med akademiskt

förvärvade kunskaper, så det syns att en del helt på egen hand har utvecklat sina färdigheter.

Det finns en nackdel med att ha fullständigt självförvärvade kunskaper. I vanliga fall har lärare på förhand en uppfattning av hur en lektion i olika ämnen kan vara strukturerad om de har deltagit i sådan undervisning som elev. Om de däremot saknar goda (eller dåliga) förebilder kan det vara svårt att gissa vad som kommer fungera bra, och vilka moment som kräver mycket tid. Mycket av den egna undervisningen kan då bli pedagogiska försök och misstag. Jag tror att dialog, samarbete och stöd inom kollegiet är viktigt här. Det är viktigt att dela med sig av både sina goda och dåliga erfarenheter. Man kan till och med åhöra varandras lektioner om möjligheten finns. Ytterligare en möjlighet är att delta i diskussionsforum för lärare på sociala medier, där alla kan dela idéer, lektionsplaner och kodexempel samt övningar.

De icke-positiva erfarenheter och känslor som lärarna skildrade var nästan uteslutande anknutna till antingen pedagogiska utmaningar (PK) eller innehållsmässiga (CK). Trots att många skolor uppenbarligen ännu inte hade ett högt antal datorer per elev, verkar majoriteten ha klarat sig med den utrustning som finns till hands.

Endast 4 respondenter nämnde spontant någon form av teknisk utmaning (TK), men om svaren på en annan fråga om materialtillgångar räknas med, kan antalet utökas till 8. Dessa ytterligare 4 fall av tekniska problem är däremot inte en följd av lärarens tekniska kompetens, utan en direkt följd av otillräcklig utrustning i skolorna, till exempel pekplattor utan tangentbord eller svårigheter med att boka tid till datasalen.

Frågan kring fortbildningarnas roll återstår. Av svaren att döma går det inte att peka ut någon viss sorts fortbildning som har varit extra populär. Många av dem som har beskrivit fortbildningarna som givande uppgav att de har deltagit i flera olika. De övriga som uppger att fortbildningen inte varit givande har i regel bara deltagit i en enda. I dessa fall har de ofta valt att inte berätta något om denna fortbildning. Fortbildningarnas mångfald verkar återspegla programmeringens breda möjligheter att närma sig från olika infallsvinklar, och för de som hittills inte varit nöjda gäller kanske att delta i några till, tills de upptäcker ett tema eller arbetssätt som väcker intresset och nyfikenheten.

6.2 Attityd

Hur förehöll sig lärarna till att undervisa programmering? De spontana upplevelser eller åsikter respondenterna väljer att beskriva avslöjar mycket om vad de har för attityd (svaren främst ifrån fråga 7, men även övriga fria formuleringar har beaktats). En del utmaningar som respondenterna räknade upp vittnade om att det fanns någon komponent i deras pedagogiska ämnesspecifika kompetens som inte räckte till, att deras vanliga metoder som fungerar för övrig matematikundervisning exempelvis inte kan tillämpas. Samtidigt är undervisningen ofta mer beroende av hur väl tekniken fungerar än övrig matematikundervisning:

> *Utmanande. Den pedagogiska biten är mycket svårare. Dessutom uppstår andra problem t.ex. tekniska som gör det knepigare. Att installera program på datorn, behörigheter, otillräcklig utrustning osv.*

En del erkände att de kände sig osäkra på olika sätt, ett fenomen som också en tidigare undersökning (Kallio-Kujala, 2017) avslöjade hos klasslärarna. Matematiklärarna är vana vid att vara experter, som kan hitta många olika representationsformer för att förklara begrepp. Utan bred programmeringserfarenhet är läraren plötsligt inte längre experten, och elever lägger märke till osäkerhet. Om läraren plötsligt visar tecken på osäkerhet sprider sig i värsta fall den känslan vidare till elever. Den ställvis rådande materialbristen gör att läraren inte alltid kan stödja sig på materialets förklaringar och ofta måste kunna hitta på egna.

I vissa fall har läraren till exempel påpekat att det är viktigt att eleverna får ta del av ämnet, men att de egna kunskaperna är bristfälliga. Detta verkar beskriva många lärares situation, att viljan finns men känslan av att inte räcka till dominerar diskursen. Många är osäkra eller upprörda gällande antalet undervisningstimmar, och tycker tiden inte räcker till eller konstaterar att de bara spenderar någon timme per läsår på programmering. Denna situation kan eventuellt bli bättre med tiden allt efter att nya elever har ökad programmeringserfarenhet ifrån årskurserna 1–6. Med ett sådant

elevunderlag kan programmeringen tidigare integreras i övrig undervisning och spara tid.

Det förekom svar som var direkt negativt inställda till programmering, svar som antydde att det inte alls tjänade något syfte att undervisa programmering. Å andra sidan fanns det aningen flera svar där respondenterna lyfte fram positiva erfarenheter. Många berättade om att deras elever tyckte det var roligt, kändes nyttigt eller intressant att programmera. Både de med positiva och negativa erfarenheter kunde beskriva undervisningen och planeringen som svår, utmanande eller jobbig. Det påpekades att man måste vara väl förberedd. Kanske denna ökade arbetsbörda kan vara en orsak till frustration.

Några fria formuleringar antydde att några av lärarna antingen lät bli att undervisa programmering, eller höll någon enstaka lektion per läsår. Detta oroväckande resultat dök även upp i en tidigare undersökning (Kolu, 2020) och det kan konstateras att fenomenet antagligen ger upphov till stora skillnader i kunskap, både regionalt och från klass till klass inom samma skola. Det bör samtidigt påpekas att så länge programmering finns utskrivet som ett mål i läroplanen, och bedömningskriterier för årskurs 9 är publicerade borde alla göra så gott de kan med elevernas rätt till samma utbildning i åtanke.

Att lyckas motivera eleverna beskrevs ofta som en utmaning. Många svar tydde på att lärare väljer att ta upp programmeringen under tidpunkter som kan anses vara ofördelaktiga (slutet av terminer efter att betygen är skrivna, eller som utfyllnad då det råkar finnas lite tid). I så fall kommunicerar läraren samtidigt åt eleverna att deras insats inte påverkar betyg, i värsta fall att det inte finns krav eller förväntningar. Det kan finnas skäl att överväga hur undervisningstimmarna ska fördelas och vad som behöver prioriteras. Förmågan att motivera elever hör främst till den pedagogiska kompetensen. Ett trick för att bibehålla elevernas motivation är att inte ge för svåra uppgifter, då känslan av att lyckas med en uppgift kan vara en stark drivkraft (Barton, 2018). Samtidigt rapporterar många lärare om elever som överträffar klasskamraterna och kräver fördjupning.

Endast en tredjedel av lärarna väver in programmering med övrig undervisning. Resten behandlar det som ett separat tema. Tabell 6, Tabell 7 och Tabell 8 avslöjar att det existerar en klar splittring i målgruppen. Förvisso skulle majoriteten föredra att

programmeringen fick status som ett separat skolämne i stället för att vara del av matematiken, å andra sidan tycker också majoriteten att matematiken gynnas av att inkludera den. En orsak till att många har undvikit att ta ställning till om programmering bör undervisas som del av matematiken i fråga 5 kan vara att de vill bevara den situation som för tillfället råder. Programmering undervisas nämligen både separat som tillval och integrerat[4] i många skolor. På så vis garanterar de att intresserade elever erbjuds fördjupning.

En betydande andel är osäkra på om de är nöjda eller missnöjda med situationen och förskjutningen från ”Ja” & ”nej” till ”Vet inte” mellan tabell 4 och 5 visar att många som uttryckte sig för matematikens del inte är säkra på ifall samspelet ur programmeringens synvinkel är vare sig positivt eller negativt.

6.3 Programmeringsmiljö

Det var förväntat att inte så många lärare skulle använda sig av *analog programmering*. Pseudokod är inte ett begrepp som dyker upp i den pedagogiskt inriktade litteraturen kring programmering, trots att planering av program (där pseudokod ofta används) nämns i styrdokument. Av den orsaken tror jag att pseudokod absolut har en plats i grundskolans undervisning. Det är ett bra verktyg för att åskådliggöra en algoritm innan alla är bekväma med syntaxen, särskilt om målet är att jobba med ett textbaserat programspråk där syntaxfel är vanligast.

Den jämna fördelningen mellan text- och blockprogrammering väcker inga frågor, för båda har sina för- och nackdelar. Blockprogrammeringen ger ett stadigt fotfäste för nybörjare och används flitigt i samband med programmering av diverse robotar, så det har sin plats i årskurserna 7–9. Textbaserad programmering står däremot närmare matematikens språk och har antagligen därför dominerat läroböckerna. Detta har enligt enkätens data i sin tur varit en avgörande faktor i matematiklärarnas val att undervisa textbaserade språk, oftast *Python*. Lokala läroplaner specificerar ibland att äldre elever

[4] Ett svarsalternativ som representerar detta, ”både skilt och som del av matematiken”, borde ha erbjudits i enkäten.

ska lära sig *Python* men många respondenter ställer sig skeptiska till alternativet att standardisera något språk på nationell nivå. Om man däremot vänder blicken till andra stadiet, så är *JavaScript* och *Python* de enda programspråk som går att använda i studentexamensmiljön *Abitti* (Studentexamensnämnden, 2021) och således nyttiga alternativ.

Det bör också nämnas att lärare har använt övrig programvara, såsom t.ex. *Microsoft Excel* eller *Geogebra* och ansett att det fyller programmeringskvoten. Dessa program ger användaren möjligheten att automatisera många uppgifter och innehåller miljöer för att utveckla skript, *Visual Basic* i *Excel* samt *Geogebra script*. Normal, skriptlös användning av programmen på sin höjd klassas som *slutanvändarprogrammering* (Blackwell, 2002) och ger kanske lite träning i det algoritmiska tänkandet. De kan användas för att implementera *villkor* samt träna användandet av variabler och färdigt definierade *funktioner*. Däremot är upprepning, *while*- eller *for-loopar*, inte lätta att representera.

Ytterligare en aningen begränsande faktor (som inte nämnts av respondenterna) är att programvaran som läraren väljer att jobba med bör följa lagar om dataskydd. En del nätbaserade resurser kräver att något konto registreras och kan visa sig spara användardata utanför EU. Då godkänns de inte när en s.k. *Data Privacy Assessment* görs. I praktiken resulterar denna säkerhetsbedömning oftast bara i fördröjningar innan program går att installera på elev- och lärarenheter.

6.4 Läroböcker och läroplanen

De äldre böckernas innehåll täcker fortfarande den nuvarande läroplanens övriga mål väl. Dessutom säljs de nya e-böckerna som licenser och en traditionell lärobok som återanvänds ca. 3 gånger eller fler blir en billigare investering i längden. Detta förklarar varför en förvånansvärt stor andel av respondenterna inte hade tagit någon ny lärobok som innehöll programmering i bruk (se Tabell 10). Samtidigt är denna grupp också de som oftast tyckte sig ha allt material de behövde. Då läroplanen nyss hade trätt i kraft hösten 2016 rådde det brist på färska, uppdaterade läromedel. En möjlig förklaring är

att lärarna under denna period tvingades utarbeta sitt eget material. De som var nöjda med sitt eget material såg antagligen ingen orsak att införskaffa nya läroböcker.

Det påpekas också att det finns vissa risker med att ha programmeringsmaterial i bokform. Varje tryckfel i något av kodningsexemplen kan resultera i att exemplet inte går att exekvera korrekt trots att en elev kopierar koden tecken för tecken[5]. Digitala plattformar erbjuder läroboksförfattaren möjligheten att rätta fel och det finns inte samma behov av att trycka nya upplagor.

Flera av respondenternas svar reflekterade min åsikt att den nuvarande formuleringen i läroplanen är vag, och att den därmed tvingar lärarna att fatta de flesta besluten om hur undervisningen ska förverkligas. Å andra sidan har diverse senare publikationer av undervisningsstyrelsen förtydligat en del oklarheter. Ifall läraren vill få till stånd en mera varierad undervisning är det alltid en fördel att vända sig till ytterligare källor, oavsett vilken bokserie som används. Extra material och variation i programmeringsmiljön kan rent av vara nödvändigt för att väcka intresse, och det är positivt att se att många lärare har utforskat alternativen.

6.5 Metoddiskussion

Det var lämpligt att samla in data för undersökningen med hjälp av en elektronisk enkät. Pandemiläget som rådde hade försvårat alla former av besök till skolor, och antalet intervjuer som behövs för att kunna representera hela Svenskfinland hade varit för stort. En enkät som kunde fyllas i under några lediga minuter av arbetsdagen erbjöd den låga tröskel som tillät alla som önskade att delta. Säkerligen har ändå ett bekvämlighetsurval uppstått. Trots det har spridningen ibland deltagarnas åsikter varierat mycket, så ingen potentiell bias kan ha varit för stor.

I jämförelse med liknande undersökningar utförda på jämförbara målgrupper (Kallio-Kujala 2017, Kolu 2020) verkar svarsprocenten vara bättre än man skulle förvänta sig.

[5] Exempelvis första upplagan av Kubik utgiven 2017 av Schildts & Söderströms har ett återkommande fel på sida 38 - både i teorin och exempel 10. Alla rader med *elif* och *else* saknar ":"-tecknet vilket resulterar i syntaxfel om koden testas genom att kopiera den tecken för tecken.

Möjligtvis stämde min teori om att lärare skulle ha mera överloppstid i slutet av läsåret efter att vitsorden hade fastslagits. Tidpunkten är också en sådan då lärare blickar tillbaka till det gångna läsåret och reflekterar kring vilka förbättringar som kan göras för det följande. Att också kontakta respondenterna direkt i stället för att be rektorer vidarebefordra budet ökade antagligen andelen mottagare en aning, och var i min mening värt mödan.

Mycket arbete kan inbesparas om enkäten som sänds ut är väl förberedd. I värsta fall går forskaren miste om viktigt data ifall formuleringar är otydliga. En brist med enkäten som uppdagades i ett senare skede var att många respondenter antagligen inte var säkra på vad som menades med ”programmering utan digital miljö (fråga 24)”. En kort definition på analoga miljöer samt exempel på samma sätt som i fråga 25 hade antagligen gjort att flera svar hade kunnat användas. Ytterligare ett misstag fanns i fråga 4, där svarsalternativet ”högst 5p/3v” borde ha varit ”mindre än 5p/3v” för att undvika överlappande intervall. Tack vare att alternativen ordnades i fallande ordning har de flesta ändå antagligen valt ”rätt svarsalternativ”.

Antalet svarsalternativ i enkäten begränsades ofta till 3 eller 4. Trots att det hade varit lockande att i varje fråga begränsa svaren till ett jämnt antal och således tvinga fram ett ställningstagande, hade i så fall inte den andel lärare som nöjer sig med att vara neutrala i frågan uppmärksammats. Utrymme för fritt formulerade kommentarer och svar i många frågor var fördelaktigt med tanke på att respondenternas svar inte begränsades till frågeställarens förhandsuppfattningar. Nackdelen med kvalitativa svar är att de ger upphov till mycket ytterligare arbete i behandlingsskedet.

7 Avslutning

Idag banar vi kanske väg för framtidens programmeringsundervisning där alla nyutexaminerade lärare, inte bara matematiklärarna, har erbjudits möjligheten att utveckla sitt datalogiska tänkande. Denna nya generation kommer kanske att hitta tillämpningarna inom de övriga ämnena som ligger och väntar på att bli upptäckta. Då kommer inte längre en liten grupp av lärarna ensamma axla detta ansvar. Tills dess är det tyvärr bara matematiklärarna som har någon erfarenhet av programmering, så det lär ännu räcka några år innan det kan dyka upp som ett verktyg i övriga skolämnen.

Ifall läsning eller skrivning endast tränades och användes under modersmålslektionerna, eller teknisk utrustning endast utnyttjades i datatillval hade vi inte heller utbildat medborgare med goda läskunskaper eller hög digital kompetens. Den nuvarande lösningen, där matematiklärarna bär ansvaret för programmering, är beklaglig men nödvändig för att få en mera djupgående samhällelig förändring till stånd. Alla medborgare behöver inte bli programmerare, men alla får chansen att upptäcka det och en grogrund för framtida lärande.

Samtidigt syns inga tecken på att programmering kommer att plockas bort från vare sig läroplanen eller matematikämnet då det spelar en central roll i undervisnings- och kulturministeriets utvecklingsprogram *Nylitteracitet* (Nationella Audiovisuella Institutet & Utbildningsstyrelsen, 2020). Denna satsning visar att viljan att fortsätta driva igenom denna förändring är stark. Av den orsaken bör lärarutbildningarna se över universitetsstudierna för såväl blivande klasslärare som ämneslärare för att nya lärare ska vara väl förberedda inför uppdraget.

7.1 Arvet från grundskolan

En tydlig progression inom grundskolan är att programmeringen introduceras som en komponent till spel och robotar, eller som verktyg för att skapa sagor och berättelser. I de högre årskurserna börjar elever få bekanta sig med textbaserad programmering och tanken på att använda det som ett nyttigt verktyg etableras. Hur borde då andra

stadiets utbildningar plocka upp stafettpinnen och stöda det fortsatta lärandet? I gymnasiets allmänbildande anda kunde programmeringen i mån av möjlighet gärna få tillämpas utanför några enstaka kurser, och gärna även användas som ett verktyg i flera ämnen än bara matematik. Tack vare att studentexamen nu är helt digitaliserad och *abitti*-miljön redan innehåller en *python* kompilator så existerar redan all nödvändig infrastruktur för att också ordna studentskrivningar med flera datavetenskapliga inslag. I lång matematik har simuleringsuppgifter till exempel förekommit.

Ökad exponering under skoltiden gör att flera unga har en bättre bild av vad programmering innebär och vilka dörrar det öppnar i arbetslivet, där fokuset alltmer går över till automatisering. Just av denna orsak har så många länder genomfört stora läroplansförändringar, för att utbilda konkurrenskraftiga medborgare.

7.2 Framtida forskningsförslag

Utbildningsstyrelsen publicerade förtydligande bedömningskriterier för slutbedömningen som togs i bruk läsåret 21–22, och kriteriet för vitsord 7 i M20 är att eleven både ska kunna läsa, testa och tolka program samt använda villkor och upprepning i den egna programmeringen. Det hade varit intressant att undersöka hur lärarna anpassar sig till dessa kriterier. I matematiken spelar prov och förhör ännu en central roll i bedömningen, men hur gör lärarna för att bedöma elevernas algoritmiska tänkande och färdigheter att tillämpa programmering? Vågar lärarna försöka sig på någon form av elektroniska prov, eller använder de sig av papper och pseudokod? Kommunicerar lärarna om hur avsnittet bedöms till eleverna? Gör eleverna projekt som lämnas in eller behandlas programmering så sent i terminen att alla vitsord är inskrivna och därmed orubbliga? Jag har personligen upplevt att det inte är lätt att utvärdera elever när det kommer till programmering och hade gärna tagit del av hur andra gör.

En annan intressant aspekt som inte undersöktes var hurudana källor lärarna skaffar idéer och material från. Med undervisningsspråk på svenska har Finlandssvenska lärare delvis den nackdelen att läroböcker ibland är översatta och därmed också görs

tillgängliga i ett senare skede. Å andra sidan kan vi ”importera” material från Sverige, som befinner sig i en liknande utvecklingsfas. Att döma av svaren så finns det en stor mängd material och idéer som redan är i användning, men man känner kanske bara till det egna lilla som man har exponerats för genom fortbildningar eller kollegiet.

8 Litteraturförteckning

Barnes, S. (2010). *User Friendly: A Short History of the Graphical User Interface*. Sacred Heart University Review, 16(4). Hämtad 2021, 30 augusti från https://digitalcommons.sacredheart.edu/shureview/vol16/iss1/4

Barton, C. (2018). *Hjärnan i matematikundervisningen.* (Upplaga 1) Natur & Kultur

Björklund, K. (2015, mars). Finland uppgraderar. *Origo*, hämtad 3.5.2019 från https://web.archive.org/web/20170711135007/http://tidningenorigo.se/finland-uppgraderar/

Blackwell, A. (2002). *What is Programming?* Hämtad 2022, 15 februari från http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.1345&rep=rep1&type=pdf

Bryman, A. (2012). *Social research methods (Upplaga 4)*. Oxford university press.

Department for education. (2013). *Computing (s. 178).* Hämtad 3 april från https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/425601/PRIMARY_national_curriculum.pdf

Emanuelsson, G. (1986). Hur mycket styr läroböckerna. *Nämnaren 13 (2/3)*, 85–87.

Fuentes Martinez, A. (2019). “Gender and prior knowledge factors in pupils’ beliefs about programming in mathematics”. Hämtad 19.9.2021 från https://doi.org/10.21125/iceri.2019.1373

Hansen, V & Zweng, M. (Red.) (1984). *Computers in Mathematics Education*. The National Council of Teachers of Mathematics

Hasanovic, A (2020). *TPACK - Teknisk, pedagogisk och innehållsmässig kompetens*. [Kandidatuppsats, Göteborgs universitet]. Gothenburg University Publications Electronic Archive. https://gupea.ub.gu.se/handle/2077/66542/

Hu, Y., Chen, C.-H., & Su, C.-Y. (2021). *Exploring the Effectiveness and Moderators of Block-Based Visual Programming on Student Learning: A Meta-Analysis. Journal of Educational Computing Research, 58(8), 1467–1493*. https://doi.org/10.1177/0735633120945935

Ifous. (2020). Programmering i skolan- Vad, hur, när och varför? hämtad 7.8.2021 från https://www.ifous.se/app/uploads/2020/11/202010-ifous-2020-5-f.pdf

Kallio-Kujala, P (2017). *Klasslärarnas beredskap och förhållningssätt till att undervisa i programmering.* [Opublicerad kandidatavhandling, Åbo Akademi]

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

Kolu, M. (2020). *Ohjelmoinnin opettaminen suomenkielisissä yläkouluissa.* [Pro gradu avhandling, Haaga-Helia ammattikorkeakoulu]

Korhonen, T, Salo, L, Sormunen, K. (2019), Making with Micro:bit: Teachers and Students Learning 21st Century Competences through the Innovation Process.

FabLearn Conference 2019, New York, United States, 09/03/2019. 120-123, https://doi.org/10.1145/3311890.3311906

Kurtz, T (1997). *An Open Letter from Thomas E. Kurtz.* https://web.archive.org/web/19970702201824/http://www.truebasic.com/TBI_TEK.html

Läroplan för grundskolan, förskoleklassen och fritidshemmet: Reviderad 2019. (2019). Skolverket. https://www.skolverket.se/getFile?file=4206

Mannila, L. (2017). *Att undervisa i programmering i skolan: varför, vad och hur?* (Upplaga 1). Studentlitteratur.

McCracken, H. (2014). Fifty Years of BASIC, the Programming Language That Made Computers Personal. Time magazine. Hämtad 2021, 1 oktober från https://time.com/69316/basic/

Mishra, P. (2019). *Considering Contextual Knowledge: The TPACK Diagram Gets an Upgrade.* https://doi.org/10.1080/21532974.2019.1588611

Nationella Audiovisuella Institutet, Utbildningsstyrelsen. (2020). *Programmeringskunnande*. Hämtad 26.4.2022 från https://uudetlukutaidot.fi/sv/programmeringskunnande/

OECD. (2020), "Material resources available at school", PISA 2018 Results (Volym 5): Effective Policies, Successful Schools, OECD Publishing. https://doi.org/10.1787/2a420765-en.

Rice, J. (2007) *Mathematical Statistics and Data Analysis*. (Upplaga 3). Belmont: Thomson Brooks/Cole

Shulman, L. (1986). *Those who understand: Knowledge growth in teaching.* Educational Researcher, 15(2), 4–14.

Skolverket. (2019). *Matematikundervisningen med digitala verktyg II, årskurs 7-9*. Hämtad 2022, 26 april från https://larportalen.skolverket.se/LarportalenAPI/api-v2/document/path/larportalen/material/inriktningar/1-matematik/Grundskola/438_matematikundervisningmeddigitalaverktygII_%C3%A5k7-9/del_01/1.%20Om%20programmering.pdf

Statsrådets förordning om ändring av 6 § i statsrådets förordning om riksomfattande mål för utbildningen enligt lagen om grundläggande utbildning och om timfördelning i den grundläggande utbildningen. (793/2018). Statsrådet. https://www.finlex.fi/sv/laki/alkup/2018/20180793

Stukát, S. (2011) *Att skriva examensarbete inom utbildningsvetenskap.* Lund: Studentlitteratur

Utbildningsstyrelsen. (2014). *Grunderna för läroplanen för den grundläggande utbildningen 2014.* Utbildningsstyrelsen.

Utbildningsstyrelsen. (2019). *Lärarna och rektorerna i Finland*. Hämtad 2021, 1 december från https://www.oph.fi/sv/statistik/lararna-och-rektorerna-i-finland

Utbildningsstyrelsen. (2020). *Bedömningskriterier för årskurs 9 vid slutbedömningen*. Hämtad 2022, 8 mars från https://www.oph.fi/sites/default/files/documents/Bed%

C3%B6mningskriterier%20f%C3%B6r%20slutbed%C3%B6mningen%20i%20den%20grundl%C3%A4ggande%20utbildningen%2031.12.2020_6.pdf

Utbildningsstyrelsen. (2021). *Programmeringsbegrepp*. Hämtad 2021, 19 september från https://www.oph.fi/sv/programmeringsbegrepp

Utbildningsstyrelsen. (2022). *Lektionsplaner*. Hämtad 2022, 4 februari från https://www.oph.fi/sv/lektionsplaner

Wing J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, *366*(1881), 3717–3725. Hämtad 2021, 21 december från https://doi.org/10.1098/rsta.2008.0118

9 Bilagor

9.1 Frågeformulär

Den svarandes bakgrund och erfarenhet

1. I vilken kommun arbetar du?

2. Vilket år är du född?

3. Hur många år har du undervisat matematik?

4. Hur många studiepoäng/-veckor av programmering, IT eller datavetenskap har du avlagt?

- Minst 60p / 40v
- Minst 25p / 15v
- Minst 5p / 3v
- Högst 5p / 3v[6]
- Vet inte

5. Välj alternativet som beskriver dig bäst, gällande din kunskap om programmering.

- Jag har en rigid kunskap om programmering och kan även handleda elever som vill fördjupa sig.
- Mina kunskaper täcker behoven i åk 7–9.
- Jag har lösryckt kunskap, men känner att det inte räcker.
- Jag känner att jag inte kan programmera alls.

6. Hur har Du förvärvat dessa programmeringskunskaper? Exempelvis fortbildning, kurser vid universitet, självlärd...

7. Hurdana har Dina erfarenheter kring undervisning i programmering varit? Jämför t.ex. med normal matematikundervisning.

Skolans stöd, utrustning och material

8. Hur mycket stöd har Du fått av undervisningsanordnaren för att kunna skaffa dig kunskaper och färdigheter i att undervisa programmering sedan det infördes i läroplanen?

- Jag har fått mycket stöd
- Jag har fått stöd
- Jag har fått lite stöd
- Jag har inte fått stöd
- Jag har inte behövt stöd

9. Hur mycket stöd har Du fått av dina kolleger för att kunna skaffa dig kunskaper och färdigheter i att undervisa programmering?

- Jag har fått mycket stöd
- Jag har fått stöd
- Jag har fått lite stöd
- Jag har inte fått stöd
- Jag har inte behövt stöd

10. Ifall läroboken Du använder dig av har innehållit material / avsnitt om programmering, har detta varit användbart?

- Ja
- Nej
- Läroboken innehåller inte programmering

[6] Det var meningen att skriva ”under”, i stället för ”högst”. Misstaget upptäcktes då redan några hade svarat och åtgärdades därför inte.

11. Hurudan hårdvara finns tillgänglig för eleverna i din skola? (Flera svar kan väljas)

- ☐ Skolan förser varje elev med en egen lånedator (dvs. maskin med Windows, Mac OS, Chrome OS eller Linux)
- ☐ Skolan förser varje elev med en pekplatta (Android eller iOS)
- ☐ Skolan har datasal(ar)
- ☐ Skolan har laptopar som kan lånas ut under lektioner
- ☐ Annat

12. Har Du all utrustning och allt material som krävs?

- ○ Ja
- ○ Nej

13. Övriga kommentarer om materialet Du har.

Fortbildningar

14. Har Du deltagit i fortbildning gällande undervisning i programmering?

- ○ Ja
- ○ Nej
- ○ Jag har inte erbjudits möjligheten.

15. Hur lärorik anser Du att fortbildningen var för dig?

- ○ Mycket givande
- ○ Ganska givande
- ○ Inte så givande
- ○ Jag fick inte ut något av den

16. Berätta kort om fortbildningen/fortbildningarna Du deltagit i.

Åsiktsfrågor

17. Hur viktigt anser du det är att programmering undervisas för alla, och inte bara erbjuds som tillval?

- ○ Mycket viktigt
- ○ Viktigt
- ○ Inte så viktigt
- ○ Inte alls viktigt

18. Vilket påstående beskriver din åsikt bäst?

- ○ Programmering bör undervisas som en del av matematiken.
- ○ Programmering bör undervisas som ett skilt ämne.
- ○ Har ingen åsikt i frågan.

19. Gynnas matematikundervisningen av att innehålla programmering?

- ○ Ja
- ○ Nej
- ○ Vet inte

20. Gynnas programmeringen av att vara en del av matematiken?

- ○ Ja
- ○ Nej
- ○ Vet inte

21. Valfri motivering kring de tre föregående frågorna

22. Borde läroplanen styra vilket programspråk som används och vilka teman som tas upp?

- ○ Ja
- ○ Nej
- ○ Annat

Ämneslärargruppens strategi

23. Berätta kort om Er strategi kring programmeringen. Tas det t.ex. upp som ett skilt tema eller integreras det med resten av undervisningen?

24. Hur stor andel av programmeringsundervisningen (%) spenderar Ni på programmering, men utan en digital miljö?

Besvaras med tal mellan 0 och 100

25. Hur stor andel av programmeringsundervisningen (%) spenderar Ni på en visuell programmeringsmiljö under hela åk 7–9? (Scratch, micro:bit, Construct, Lego Mindstorms)

Besvaras med tal mellan 0 och 100

26. Hur stor andel av programmeringsundervisningen (%) spenderar Ni på en textbaserad programmeringsmiljö? (Python, Java, C)

Besvaras med tal mellan 0 och 100

27. Ifall det ännu finns något som Du vill lyfta fram gällande undervisning i programmering eller enkäten kan du göra det här. Ordet är fritt!

28. Är du intresserad av att svara på flera frågor eller intervjuas vid ett senare tillfälle?[7]

- Ja
- Nej tack

29. Hur och när kan jag nå dig bäst? (t.ex. telefonnummer eller epost som läses även under sommaren)

[7] Detta hade jag utnyttjat ifall svarsprocenten hade varit lidande. 9 st. lämnade sina kontaktuppgifter och jag vill rikta ett varmt tack till dem.