Lärares datalogiska tänkande

En fallstudie om finlandssvenska grundskolelärares lärande inom ett Scratchprojekt

Julia Ljungman

Avhandling för pedagogie magisterexamen
Fakulteten för pedagogik och välfärdsstudier
Åbo Akademi
Vasa, 2018

Abstrakt

Författare	Årtal
Ljungman, Julia	2018
Arbetets titel	
Lärares datalogiska tänkande. En fallstudie om finlandssvenska grundskolelärares lärande inom ett Scratchprojekt.	
Opublicerad avhandling för magisterexamen i pedagogik.	Sidantal
Vasa: Åbo Akademi, Fakulteten för pedagogik och välfärdsstudier.	63

Referat

Syftet med den här avhandlingen är att studera finlandssvenska grundskolelärares datalogiska tänkande inom ramen för nätkursen *Programmering i grundskolan – vad, hur och varför?,* som har getts som fortbildningskurs för lärare våren 2016. Kursen behandlade bl.a. det visuella programmeringsspråket Scratch (läs mera om det i kapitel 4). Utgående från syftet och Brennans och Resnicks (2012) definition av datalogiskt tänkande formulerades två forskningsfrågor:

1. Vilka datalogiska begrepp har lärarna använt sig av i sina Scratchprojekt i en fortbildningskurs om programmering?
2. Vilka datalogiska handlingsstrategier och perspektiv har lärarna nämnt att de har använt sig av i sina kommentarer till Scratchprojekten i en fortbildningskurs om programmering?

Forskningen är en kvalitativ fallstudie som bäddas in i teorin om praktikbaserad gemenskap, där lärandet ses som ett ökat deltagande i gemenskapen, i det här fallet i programmeringsgemenskapen. Datamaterialet består av 15 lärares Scratchprojekt och tillhörande kommentarer, som lärarna har skapat som en del av nätfortbildningskursen *Programmering i grundskolan – vad, hur och varför?*. Scratchprojektens koder och de tillhörande kommentarerna analyseras med hjälp av ett analysverktyg som utvecklades ur Brennans och Resnicks (2012) definition av datalogiskt tänkande. I Brennans och Resnicks definition delas det datalogiska tänkandet upp i tre dimensioner: begrepp, handlingsstrategier och perspektiv. Dessa tre dimensioner har även flera underkategorier (se bilaga 1).

Resultaten visar att de flesta nätkurslärare har använt sig av alla eller nästan alla sju datalogiska begrepp som en programmerare ska kunna använda sig av och förstå enligt Brennan och Resnick (2012). De datalogiska begrepp som oftast saknades i lärarnas projekt var *loop*, *parallellism* och *operator*. Det visar sig också att nätkurslärarna oftare spontant har kommenterat de mera konkreta datalogiska handlingsstrategierna *stegvis och via upprepning* samt *testa och felsöka*, än de mera abstrakta handlingsstrategierna *återanvända och remixa* samt *abstrahera och skapa modeller*. Det datalogiska perspektivet *att förenas* var lättare för lärarna att spontant beskriva i sina kommentarer, medan perspektiven *att uttrycka sig* och *att ifrågasätta* var svårare.

Sammanfattningsvis kan det konstateras att nätkurslärarna upplevde programmeringskursen som givande. Flera lärare uttryckte att det var roligt att programmera i Scratch och alla lärare lyckades göra ett fungerande Scratchprojekt. Det är svårt att uttala sig om specifika lärares datalogiska handlingsstrategier och perspektiv, eftersom kommentarerna som lärarna har skrivit inte styrdes på något vis och många kommentarer var därför korta. Trots det kan resultaten ses som

exempel på hur lärares datalogiska tänkande kan se ut efter en grundkurs i programmering. Resultaten belyser även hur lärare överlag kan tänka och känna samt vilka programmeringskunskaper lärare kan ha om Scratch.

Sökord
datalogiskt tänkande, computational thinking, programmering, programming, ohjelmointi, scratch

Innehåll

Figurer

Tabeller

1 Inledning

Denna avhandling behandlar lärares datalogiska tänkande. I det inledande kapitlet presenteras avhandlingens bakgrund och val av tema. Vidare redogörs för avhandlingens syfte och forskningsfrågor samt avhandlingens fortsatta upplägg. Avslutningsvis beskrivs nätfortbildningskursen *Programmering i grundskolan – vad, hur och varför?*, eftersom kursen fungerar som kontext till den här fallstudien.

1.1 Bakgrund och val av tema

Framtidskompetenser (eng. *21st century skills*) är ett samlingsnamn på olika kompetenser som är viktiga att behärska för att kunna anpassa sig och leva i ett samhälle, en värld och ett arbetsliv som ständigt förändras. Forskarna Voogt och Roblin (2012) har jämfört flera internationella modeller som beskriver framtidskompetenserna. De kom fram till att framtidskompetenserna kännetecknas av att de är *transversella*, alltså viktiga inom många olika områden. De är också *multidimensionella*, vilket betyder att de innefattar både kunskaper, färdigheter och attityder och de förknippas med *avancerade kognitiva funktioner och beteenden* som visar på att man behärskar komplexa problem och oförutsägbara situationer. Begreppet framtidskompetens innefattar oftast kompetenserna samarbete, kommunikation, digital litteracitet, medborgarskap, problemlösning, kreativitet, produktivitet och livslångt lärande. (Voogt & Roblin, 2012, s. 302, 309.)

Många kompetenser som förknippas med framtidskompetenserna är också viktiga färdigheter för att utveckla *datalogiskt tänkande*, som är ett av avhandlingens nyckelbegrepp. Datalogiskt tänkande handlar om att utveckla och automatisera förmågan att med en dators hjälp kunna lösa problem oberoende av ämnesområde. Ett utvecklat datalogiskt tänkande innebär att man både kan använda och skapa nya tekniska verktyg. Därför skapar det datalogiska tänkandet förståelse för att tekniska lösningar är mänskliga innovationer. För att uppnå datalogiskt tänkande måste flera kognitiva färdigheter utvecklas. (Wing, 2006, s. 33.)

Enligt Wing (2006, s. 33) är datalogiskt tänkande en förmåga som alla borde besitta och inte enbart datavetare. Att undervisa i hur man gör PowerPoint-presentationer eller hur man skriver i Word engagerar inte studerande att på djupet analysera programmen – vilket skulle behövas för att utveckla elevernas kreativa och kritiska tänkande. De flesta studerande vet ganska lite om datavetenskap och om hur man tänker systematiskt så att man på ett effektivt sätt kan lösa alla typer av problem. (Kafai & Burke, 2013, s. 62–63.) För att få flera barn och unga att bli intresserade av datavetenskap har man därför i flera länder infört datavetenskap i läroplanen på något sätt. Experter världen över har samlats i grupper och försökt göra det datalogiska tänkandet mera tillgängligt för undervisning i årskurserna F–12. En lösning som flera länder anammat är att göra programmering till en del av läroplanen. Det betyder inte att det är viktigt att undervisa i specifika programmeringsspråk, utan det viktiga är att undervisa om de begrepp och principer som programmeringsspråken förmedlar. (Kafai & Burke, 2013, s. 63.)

Alla elever i grundskolan kommer inte att bli datavetare som skriver koder och planerar system som påverkar det dagliga livet, men alla människor kommer i framtiden att använda sig av digitalteknik och därför behöver alla kunna granska programvarudesign på ett kritiskt och konstruktivt sätt. Förr sades det att läsning och skrivning gav en möjlighet att förstå och förändra världen, men idag sägs det istället att kodläsning och kodskrivning ger en möjlighet att förstå och förändra världen (Kafai & Burke, 2013, s. 65).

I Finland har man för första gången skrivit in ordet programmering i den nationella läroplanen Grunderna för läroplanen för den grundläggande utbildningen 2014 (här efter Glgu 2014). Genom det beslutet hoppas man på att antalet datavetare ska öka och att de finländska eleverna ska bli bättre förberedda inför framtiden. Programmering finns som en del av ämnet matematik för årskurs 1–9, slöjd för årskurs 3–9 och även under rubriken Digital kompetensen (K5) står det att elever i årskurs 3–6 ska bekanta sig med programmering för att förstå att tekniska funktioner beror på mänskliga lösningar. (Utbildningsstyrelsen, 2014.)

Glgu 2014 trädde i kraft i augusti 2016 i Finland och det innebär att klasslärarna och matematiklärarna sedan dess har varit skyldiga att undervisa grundskolans elever i programmering. För att få eleverna i grundskolan att bli intresserade av datavetenskap

och programmering behövs engagerade och kompetenta lärare. Frågan är om klasslärarna och matematiklärarna på fältet har kunskap och redskap för att undervisa i programmering? European Schoolnets rapport från 2015 visar att det finns brister hos lärare som undervisar i digital kompetens (European Schoolnet, 2015, s. 59–60). Även datavetaren Denning visar på lärares kunskapsbrister gällande datalogiskt tänkande (Denning, 2017, s. 34). Norrena (2013) har genomfört tre fallstudier om hur man främjar framtidskompetenser i finländska grundskolor i årskurs 1–6. Han kom bland annat fram till att utmaningarna med att främja framtidskompetenserna är lärares attityder och otillräckliga färdigheter (Norrena, 2013, s. 125).

Jag är själv femte årets klasslärarstuderande med bland annat matematik som kort biämne, och jag anser att jag saknar kunskap om vad programmering är och hur man konkret ska arbeta för att uppnå de programmeringsmål som Glgu 2014 ställer. Inom klasslärarutbildningen har vi inte berört programmering alls, eventuellt har vissa studerande kommit i kontakt med programmering på någon praktik eller via vikariat. Programmering är alltså ett nytt fenomen både i grundskolan och i klasslärarutbildningen.

1.2 Syfte och forskningsfrågor

Centret för livslångt lärande vid Åbo Akademi och yrkeshögskolan Novia (CLL) ordnar många utbildningar inom pedagogik och lärarfortbildning på svenska i Finland. Våren 2016 ordnades fortbildningen *Programmering i grundskolan – vad, hur och varför?* som nätkurs för första gången. I skrivande stund (januari 2018) har kursen ordnats sammanlagt tre gånger. Syftet med avhandlingen är att studera finlandssvenska grundskolelärares datalogiska tänkande inom ramen för denna nätkurs. Nätkursen behandlar bland annat det visuella programmeringsspråket Scratch (läs mera om det i kapitel 4). Utgående från avhandlingens syfte valdes kvalitativ undersökning som metod. Eftersom jag undersöker hur en specifik grupp lärare (fortbildningsdeltagarna) klarade av samma kursuppgift, blir undersökningen en fallstudie (Merriam, 1994).

Brennan och Resnick (2012) har definierat datalogiskt tänkande som tre olika dimensioner: begrepp, handlingsstrategier och perspektiv. Utgående från deras

indelning av datalogiskt tänkande har jag konkretiserat avhandlingens syfte i två forskningsfrågor:

1. Vilka datalogiska begrepp har lärarna använt sig av i sina Scratchprojekt i en fortbildningskurs om programmering?
2. Vilka datalogiska handlingsstrategier och perspektiv har lärarna nämnt att de har använt sig av i sina kommentarer till Scratchprojekten i en fortbildningskurs om programmering?

1.3 Avhandlingens upplägg

Avhandlingen är indelad i sju kapitel. I kapitel 1, *Inledning*, presenteras bakgrunden till avhandlingen, syftet och forskningsfrågorna, avhandlingens upplägg samt nätfortbildningskursen som utgör fallstudiens bakgrund. Därefter följer tre teorikapitel. I kapitel 2, *Datalogiskt tänkande – en framtidskompetens*, definieras och diskuteras termen datalogiskt tänkande. Därefter beskrivs programmering som ett verktyg för att uppnå datalogiskt tänkande och avslutningsvis redogörs för några länders införande av datalogiskt tänkande i deras läroplan. I kapitel 3, *Lärares datalogiska tänkande*, beskrivs olika metoder för hur man kan utveckla lärares datalogiska tänkande. Fortbildning som metod beskrivs mera ingående och sist i kapitlet redogörs för utmaningar och strategier som lärare möter och använder sig av i sin programmeringsundervisning. I kapitel 4, *Scratch – ett visuellt programmeringsspråk*, introduceras Scratch som program, dess funktioner och forskning som har gjorts om Scratch.

I kapitel 5, *Metod*, redogörs för den empiriska undersökningen. Först beskrivs studiens syfte och forskningsfrågor, den praktikbaserade gemenskapen som utgångspunkt för analys av lärares lärande, den kvalitativa fallstudien, dokument som datainsamlingsmetod och bearbetningen samt analysen av data. Avslutningsvis presenteras studiens tillförlitlighet, trovärdighet och etiska aspekter. I kapitel 6, *Resultat*, redovisas resultaten av den empiriska undersökningen. Först beskrivs nätkurslärarnas användning av datalogiska begrepp. Begreppsanvändningen illustreras med skärmdumpar ur lärarnas Scratchkoder. Sedan redovisas nätkurslärarnas

användning av datalogiska handlingsstrategier och datalogiska perspektiv. Dessa exemplifieras med citat ur lärarnas kommentarer som hör ihop med Scratchprojekten. I kapitel 7, *Diskussion*, diskuteras studiens resultat i förhållande till tidigare forskning. Metodens fördelar och nackdelar diskuteras också och slutligen ges förslag på fortsatt forskning.

1.4 Nätfortbildningskursen *Programmering i grundskolan – vad, hur och varför?*

Våren 2016 ordnades för första gången nätfortbildningskursen *Programmering i grundskolan – vad, hur och varför?*. Under åtta veckors tid vägledde kurshållarna lärarna i programmeringens grunder via text, videoklipp och länkar på nätkursens hemsida. Till varje vecka hörde även en till två kursuppgifter. Kursmaterialet fanns alltså tillgängligt på nätet och lärarna och kurshållarna kunde träffas via chatt i ett gemensamt nätforum. Alla kursuppgifter laddades upp på kursbloggen. Målet med nätkursen var att lärarna efter avslutad kurs skulle känna till programvarans betydelse i dagens samhälle, veta hur de kan använda programmering som en del av undervisningen i olika ämnen och att skapa en egen plan för hur programmering kan införas i den egna skolan.

I den här studien analyserade jag kursuppgift nummer 5 och 6. I kursuppgift nummer 5 skulle lärarna skapa ett spel, en simulation av ett fenomen, en frågesport eller något liknande i de blockbaserade programmeringsspråken Scratch eller Scratch Jr. Lärarnas projekt skulle kunna kopplas till den nya läroplanen. I instruktionerna för kursuppgift 5 tipsade kurshållarna om att man kan remixa (återanvända helt eller delvis) ett annat projekt och de uppmuntrade även lärarna att fråga efter hjälp via e-post eller det gemensamma nätforumet direkt om frågor uppstod. I kursuppgift nummer 6 skulle lärarna skriva en rapport om sitt Scratch- eller Scratch Jr projekt. I rapporten skulle det ingå kommentarer om arbetsprocessen, det färdiga projektet samt en länk till Scratchprojektet eller skärmdumpar från Scratch Jr projektet. I min avhandling har jag valt att analysera lärarnas Scratchprojekt, eftersom jag kommer åt hela deras projekt via den insatta länken.

Det är svårt att veta hur många som egentligen deltar i en nätkurs, eftersom många kursdeltagare inte gör kursuppgifterna utan endast vill komma åt kursmaterialet (L. Mannila, personlig kommunikation, 6 februari 2018). I den nätkurs som ordnades på våren 2016 gjorde 39 kursdeltagare den första kursuppgiften, medan endast 23 kursdeltagare slutförde kursuppgift 5 och 6 (15 Scratchprojekt och 8 Scratch Jr projekt).

2 Datalogiskt tänkande - en framtidskompetens

I det andra kapitlet diskuteras termen *datalogiskt tänkande* ingående och jag klargör vilken definition jag använder mig av i den här avhandlingen. Ett verktyg för att utveckla datalogiskt tänkande är programmering. Programmering kommer att beskrivas som begrepp och olika programmeringsmiljöer och andra verktyg som fungerar som inkörsport för barn och unga i programmeringsvärlden presenteras. Programmeringsdelen i den finländska läroplanen Glgu 2014 beskrivs och jämförs med andra länders läroplaner, för att skapa en förståelse för hur viktigt datalogiskt tänkande är i framtidens skola.

2.1 Definition och diskussion om *datalogiskt tänkande*

Datalogiskt tänkande är en term som Papert skapade 1996, men som blev populär 2006 i och med Wings banbrytande artikel om vikten av att införa datavetenskap i skolan för alla. Det saknas en entydig definition på datalogiskt tänkande och hur man ska utvärdera barns och ungas utveckling av datalogiskt tänkande (t.ex. Brennan & Resnick, 2012, s. 1; Heintz, Mannila & Färnqvist, 2016, s. 2; Román-González, Pérez-González & Jiménez-Fernández, 2017, s. 678). Termen datalogiskt tänkande syftar på den tankeprocess som finns när man formulerar och arbetar med problem och lösningar på problem så att lösningarna kan representeras i en form som en dator kan förstå (Heintz m.fl., 2016, s. 2).

Wing (2006, s. 33) lyfter fram att det idag är lika viktigt för barn att lära sig tänka datalogiskt som det är för dem att lära sig att läsa, skriva och räkna. Hon påpekar att utvecklingen av datalogiskt tänkande inkluderar en utveckling av många olika kognitiva färdigheter. Både Papert och Wing anser att datalogiskt tänkande är mera än att kunna använda digitala verktyg. En dator behöver ändå inte vara med i alla steg av tankeprocessen. (Mannila, Dagiene, Demo, Grgurina, Mirolo, Rolandsson & Settle, 2014, s. 2.)

Det finns flera olika definitioner av datalogiskt tänkande som olika organisationer har skapat. De globala organisationerna International Society for Technology in Education (ISTE[1]) och Computer Science Teachers Association (CSTA[2]) (2011) betonar att datalogiskt tänkande är en problemlösningsprocess. Den brittiska organisationen Computing At School (CAS[3]) (2015) anser att datalogiskt tänkande ger eleverna en inblick i datavetenskapen, men att det viktigaste är att elevernas tänkande och problemlösningsfärdigheter utvecklas i alla ämnen och genom hela livet. ISTE (2016) menar att datalogiskt tänkande utvecklar elevernas strategier för hur de kan förstå och lösa problem med hjälp av teknik. Barr och Stephenson (2011, s. 51) skriver att datalogiskt tänkande är en förmåga att lösa problem med hjälp av en dator. Förmågan ska kunna automatiseras och användas i alla ämnen. Eleverna ska både använda och skapa digitala verktyg genom att lära sig använda en uppsättning av begrepp.

I min avhandling har jag valt att använda Brennans och Resnicks (2012) definition av *datalogiskt tänkande* (se tabell 1). Samma definition ligger även som grund för mitt analysverktyg. Brennan och Resnick (2012, s. 2–3) har studerat och analyserat många olika unga människors projekt och aktiviteter i Scratch[4]. Genom sina studier har de kommit fram till tre dimensioner som bygger upp det datalogiska tänkandet. Dimensionen datalogiska begrepp (eng. *computational concepts*) innefattar de begrepp som en programmerare måste förstå och kunna använda sig av när hen programmerar. Dimensionen datalogiska handlingsstrategier (eng. *computational practices*) innefattar de strategier som en programmerare utvecklar över tid och som hjälper hen att agera rätt i programmeringsprocessen. Dimensionen datalogiska perspektiv (eng. *computational perspectives*) innefattar olika perspektiv och synvinklar som en programmerare lär sig att kritiskt granska olika digitala fenomen från. Alla tre dimensioner har flera underkategorier. Jag förklarar de olika underkategorierna mera ingående i bilaga 1, eftersom underkategorierna bara beskriver dimensionerna mera

[1] *International Society for Technology in Education* (ISTE) hjälper lärare världen över att använda teknik i skolan för att lösa svåra problem.
[2] *Computer Science Teachers Association* (CSTA) hjälper och stöder lärare över hela världen att undervisa i datavetenskap i årskurserna F–12.
[3] *Computing At School* (CAS) är en organisation som förespråkar undervisning i databehandling för alla elever i skolan i Storbritannien.
[4] Scratch är ett gratis visuellt programmeringsspråk där alla kan skapa sina egna interaktiva medium i form av t.ex. berättelser, spel, animationer och simulationer. Läs mera om Scratch i kapitel 4.

ingående och underkategoriernas innebörd även kommer fram i mitt resultatkapitel. Tabell 1 ger en översikt över Brennans och Resnicks definition av datalogiskt tänkande.

Tabell 1. *Sammanfattning av Brennans och Resnicks (2012) definition av datalogiskt tänkande*

DATALOGISKT TÄNKANDE			
Dimension	**Datalogiska begrepp**	**Datalogiska handlingsstrategier**	**Datalogiska perspektiv**
Förklaring	Begrepp som en programmerare använder.	Handlingsstrategier som behövs när man programmerar.	Perspektiv som utvecklas när man lär sig programmera som t.ex. självförståelse, relation till andra och till tekniken som omger en.
Underkategorier	Sekvens Loop Händelse Parallellism Operatorer Variabel Data	Gå stegvis och upprepa Testa och felsöka Återanvända och remixa Abstrahera och skapa modeller	Att få uttrycka sig Att förenas Att ifrågasätta

Alla förslag på definitioner av termen *datalogiskt tänkande* lyfter fram förmågan att formulera ett problem som kan lösas med hjälp av en dator (i alla fall i något skede av utvecklingen). För att kunna göra det krävs det flera olika färdigheter. Dessa är färdigheten att samla in data, färdigheten att bryta ner ett problem i mindre delar, färdigheten att hitta mönster, färdigheten att tänka algoritmiskt och skapa algoritmer för att automatisera lösningen av problemet, färdigheten att logiskt organisera och analysera data, färdigheten att framställa data genom abstraktioner som modeller och simuleringar, färdigheten att testa, felsöka och evaluera för att effektivera och korrigera lösningen och färdigheten att generalisera lösningen för att kunna återanvända och remixa delar eller hela lösningen när det gäller andra problem. (Barr & Stephenson, 2011; CAS, 2015; Grover & Pea, 2013; ISTE, 2016; ISTE & CSTA, 2011.)

De olika färdigheterna som lyfts fram i definitionen av datalogiskt tänkande har en nära koppling till andra förmågor. Förmågor som stöder de datalogiska tänkandets utveckling är förmågan att reflektera, förmågan att ifrågasätta, förmågan att planera, förmågan att programmera och förmågan att använda rätt vokabulär. Det finns även attityder som är förbundna med och stöder utvecklingen av datalogiskt tänkande. Dessa attityder är kopplade till vanan att handskas med komplexitet, vanan att arbeta uthålligt med svåra och stora problem, vanan att tolerera tvetydiga problem, vanan att handskas med problem med öppna slut, vanan att kommunicera och samarbeta för att

uppnå ett mål eller en lösning och vanan att vara medvetenhet om sina egna styrkor och svagheter. Oberoende av ämne ska eleverna bli mera innovativa, kreativa och bättre på att upptäcka saker. (Barr & Stephenson, 2011; CAS, 2015; Grover & Pea, 2013; ISTE, 2016; ISTE & CSTA, 2011.)

Termen *datalogiskt tänkande* har även fått kritik. Denning (2009) anser att datalogiskt tänkande har en lång historia inom datavetenskapen som sträcker sig tillbaka till 1950-talet, då det kallades för algoritmiskt tänkande. Därför anser han att man idag bara ompaketerat färdigheter, förmågor och attityder och försöker sälja dem på nytt. Hemmendinger (2010, s. 4, 6) är lite inne på samma spår, eftersom han anser att begreppet består av flera kompetenser som alltid har funnits: problemlösningskompetens, matematisk kompetens, kompetens att resonera vetenskapligt och kompetens att använda modeller. Han är också kritisk till termen *datalogiskt tänkande* och anser att det låter nedvärderande och som om datavetare vill lära människor att tänka rätt. Denning (2017, s. 37) är kritisk till att somliga kallar det för datalogiskt tänkande så fort någon utför en uppgift med hjälp av en dator.

Att undervisa i datalogiskt tänkande är främst klasslärarnas uppgift i årskurserna 1–6 och matematiklärarnas uppgift i årkurs 7–9 i Finland, men studier världen över har visat att många lärare inte kan besvara dessa tre väsentliga frågor: Vad är datalogiskt tänkande? Hur kan man mäta studerandes förmåga att använda datalogiskt tänkande? Är datalogiskt tänkande bra för alla? Denning påpekar själv att hans kritik inte riktar sig mot att få in datavetenskapen i skolan, utan att kritiken är riktad mot svaga definitioner av begreppet datalogiskt tänkande och mot ogrundade påståenden som förs fram av entusiaster. (Denning, 2017, s. 34.)

Sammanfattningsvis kan man säga att datalogiskt tänkande är problemlösning, kritiskt tänkande, samarbete, kreativitet och kommunikation kombinerat med datoranvändning (Heintz & Mannila, u.å.). Datavetenskap är en egen vetenskaplig disciplin som studerar datorer och datorsystem. Datalogiskt tänkande är olika tankeprocesser som involveras när man löser problem. Datalogiskt tänkande är mycket mera än programmering, men programmering är ett viktigt verktyg för att utveckla datalogiskt tänkande. (Voogt, Fisser, Good, Mishra & Yadav, 2015.)

2.2 Programmering – ett verktyg för att utveckla datalogiskt tänkande

Forskning visar att programmeringsfärdigheter utvecklar det datalogiska tänkandet (Lye & Koh, 2014, s. 53; Mannila m.fl., 2014, s. 17). Därför har man i Finland valt att sätta in programmering i den nationella läroplanen Glgu 2014 bland annat som en del av ämnet matematik (Utbildningsstyrelsen, 2014). Målet med att införa datalogiskt tänkande i skolans lägre årskurser är att få elever att gå från att endast använda olika digitala verktyg till att själva skapa dem. Man hoppas också på att få flera unga att välja att studera datavetenskap, speciellt flickor och personer av olika etnisk bakgrund genom att ge barn och unga chans att bekanta sig med programmering redan i en tidig ålder (European Schoolnet, 2015, s. 14).

Enligt Grover och Pea (2013, s. 40) är programmering en grundläggande färdighet och ett nyckelverktyg för datavetenskapen som stöder olika kognitiva färdigheter som syns i ett utvecklat datalogiskt tänkande. Enligt Henriksson (u.å.) betyder programmering i databehandlingssammanhang att man skriver instruktioner för en dator så att datorn kan arbeta.

Programmering är inte samma sak som *kodning*, utan programmering ska ses som en problemlösningsprocess som bland annat kräver analys av problemet, uppdelning av problemet i mindre delar, planering av en lösning i form av en sekvens och till slut skrivande av kod. Kodning kan alltså ses som det sista steget i den mångfacetterade programmeringsprocessen och som en konkret lösning på ett problem. Alla delar av programmeringsprocessen är viktiga och det är när man beaktar hela programmeringsprocessen som man kan säga att programmering är ett verktyg som utvecklar det datalogiska tänkandet. (Mannila m.fl., 2014, s. 4.)

Programmeringsmiljöer

Flera programmeringsmiljöer för nybörjare använder sig av trestegsmodellen *Use-Modify-Create* (ungefär använda-modifiera-skapa). Trestegsmodellen stöttar och hjälper nybörjaren att gå från att endast använda ett program eller en modell som redan existerar till att modifiera den och till slut själv skapa egna program och modeller. När man programmerar och skapar något själv används de tre viktigaste färdigheterna för

datalogiskt tänkande: förmåga att abstrahera, förmåga att automatisera och förmåga att analysera. (Lee m.fl., 2011, s. 35.)

Alla programmeringsmiljöer har ett eget *programmeringsspråk* eller *programspråk*, som är det språk som alla sekvenser ska skrivas på i just den programmeringsmiljön för att en dator ska kunna tolka, förstå och verkställa den uppgiften som den är instruerad att göra. En sekvens som är skriven på ett programmeringsspråk kallas för en *kod*. Ett programmeringsspråk skiljer sig från de vanliga språken, eftersom det varken finns något utrymme för flexibilitet eller uttryckskraft (Henriksson & Nordström, u.å.). Det betyder att det inte finns något utrymme för misstag och fel när man skriver en kod.

Det finns två olika typer av programmeringsspråk som skiljer sig från varandra eftersom en kod matas in på olika sätt. När man använder *textbaserade programmeringsspråk* matas koden in manuellt i form av text. När man å andra sidan använder *visuella/grafiska programmeringsspråk* matas koden in manuellt genom att man drar och släpper (Drag-and-Drop) olika block som representerar olika programmeringsuttryck. Ibland talar man även om blockbaserad programmering. Det är oftast omöjligt att sammanlänka två block som inte passar ihop, och därför lämpar sig visuella programmeringsspråk bättre för barn, eftersom de inte kräver någon kunskap om programmeringssyntax. (Lye & Koh, 2014, s. 53.)

När man väljer programmeringsmiljöer för barn ska man tänka på att de ska ha låg tröskel och högt i tak. Med *låg tröskel* menas det att det ska vara lätt för nybörjare att skapa fungerande program i programmeringsmiljön. Med *högt i tak* menas att programmeringsmiljön samtidigt ska vara tillräckligt utmanande och kraftfull för att tillfredsställa de mera avancerade programmerarna. (Grover & Pea, 2013, s. 40.) *Rika programmeringsmiljöer* ska stötta användaren, möjliggöra återanvändning och remixning av andras program samt vara systematiska och hållbara. Rika programmeringsmiljöer ska ge användaren möjlighet att utvecklas från användare till skapare. (Lee m.fl., 2011, s. 35.)

Det finns många olika verktyg för programmeringsundervisning. I klassen kan man programmera utan datateknologiskapparatur (analog programmering) genom att till

exempel programmera en kamrat. Ett annat alternativ är att programmera robotar som till exempel LEGO Mindstorm, Sphero, Bee-Bots och Blue-Bots. Man kan även programmera med applikationer som Lightbot och Fix the Factory. Det finns också olika programmeringsmiljöer som man kan använda sig av som till exempel Alice, Greenfot, Scratch och Scratch Jr. Det mest avancerade alternativet är att programmera i textbaserade programmeringsspråk som till exempel Java och Python.

2.3 Datalogiskt tänkande i några olika länders läroplaner

År 2014 fick grundskolan i Finland (åk 1–9) en ny läroplan Glgu 2014 som både fokuserar på färdigheter i informations- och kommunikationsteknologi (IKT) och färdigheter som kopplas till datalogiskt tänkande. I Finland valde man att kalla det datalogiska tänkandet för programmering när man införde begreppet i Glgu 2014. Programmering finns nämnt för alla årskurser från 1 till 9 i matematikdelen, men också för årskurs 3–9 i slöjddelen samt för årskurs 3–6 i den allmänna delen under rubriken Digital kompetens (K5). Programmering är inte ett separat ämne i den finländska läroplanen, utan en färdighet som ska integreras specifikt i matematiken, men även i andra ämnen. (Utbildningsstyrelsen, 2014.)

Datavetenskap och programmering har i och med Glgu 2014 blivit en obligatorisk del för alla elever i grundskolan i Finland, eftersom man kommit fram till att datalogiskt tänkande är viktigt för alla och inte enbart för datavetare. Det datalogiska tänkandet är en viktig kompetens att besitta oberoende av vad eleverna kommer att studera i framtiden. (Mannila m.fl., 2014, s. 1.) Jag har valt att inte gå djupare in på läroplanens innehåll, eftersom det inte är relevant för syftet med min avhandling. Du kan läsa mera om programmering i läroplanen och hur programmering rekommenderas att implementeras i grundskolan på EDU:s [5] hemsidor.

[5] Programmering i Glgu 2014 http://www.edu.fi/planera/grundlaggande_utbildning/matematik/lp2016_-_stodmaterial_i_matematik/programmering och Programmering i skolan, lärandeprogression http://www.edu.fi/hitta_material/it_i_skolan/programmering/larandeprogression

Även i andra länder än Finland har man infört datalogiskt tänkande i skolors läroplaner. I läroplanssammanhang används flera olika begrepp som har ungefär samma betydelse, till exempel *programmering*, *databehandling*, *kodning*, *datorprogrammering*, *algoritmisk problemlösning* och *datalogiskt tänkande*. I Europa undersöktes 21 länder[6] år 2015 angående hur de har infört datalogiskt tänkande i sina läroplaner. Rapporten sammanställdes av European Schoolnet, som består av 34 europeiska utbildningsministerier. När undersökningen gjordes hade 16 länder redan integrerat programmering i sin läroplan på nationell, regional eller skolnivå. Finland och regionen Flandern i Belgien planerade då att integrera programmering i sina läroplaner, medan regionen Vallonien i Belgien, Nederländerna och Norge inte hade några planer på att göra det. (European Schoolnet, 2015.) I USA ligger kontrollen och ansvaret för utbildning främst hos delstaterna och dess lokala regeringar. Det gör att läroplanen är olika i olika delar av landet. Obama tog 2016 initiativ till att införa Computer Science for All, alltså datavetenskap för alla från daghem till andra stadiet. (Smith, 2016.)

Mannila m.fl. (2014, s. 9–10) beskriver flera länders nationella läroplaner som formella initiativ att lära elever datalogiskt tänkande, men nämner även att det finns en mängd informella initiativ som också lär eleverna datalogiskt tänkande. Som exempel på informella initiativ nämns klubbverksamhet, olika tävlingar, uppsökande verksamhet och läger. Barnen kan också lära sig av varandra, av föräldrar och syskon, på museum, på bibliotek, genom någon hobby, på Internet och så vidare. (Wing, 2008, s. 3721).

[6] Europeiska länder som deltog i undersökningen 2015 är Österrike, regionen Flanderna i Belgien, regionen Vallonien i Belgien, Bulgarien, Tjeckien, Danmark, Estland, Finland, Frankrike, Ungern, Irland, Israel, Litauen, Malta, Nederländerna, Norge, Polen, Portugal, Slovakien, Spanien och Storbritannien.

3 Lärares datalogiska tänkande

I det tredje kapitlet beskrivs olika metoder genom vilka man har försökt utveckla lärares datalogiska tänkande. Fortbildning är en metod som presenteras mera ingående, eftersom kontexten i min fallstudie utgörs av en programmeringsfortbildning. Avslutningsvis redovisas forskning om lärares upplevelser av programmering i skolan.

Resurser är avgörande för att man ska lyckas göra en systematisk pedagogisk förändring i grundskolan när man inför datalogiskt tänkande. Resurser behövs för att informera utbildningspolitiker om vikten av att införa det datalogiska tänkandet i utbildningen, att skaffa utrustning för det och att erbjuda lärare utbildning inom det. Först måste lärarna förstå vad datalogiskt tänkande innebär för dem, för att sedan kunna implementera idéerna om datalogiskt tänkande hos sina elever i praktiken. (Barr & Stephenson, 2011, s. 53.)

För att förbereda och inspirera lärare för det datalogiska tänkandets intåg i skolorna krävs olika slags aktiviteter och material framhåller Barr och Stephenson (2011, s. 53–54). Främst borde man främja den professionella utvecklingen genom fortbildningar, workshops, konferenser, sommarkurser och tutorinlärning (där en i kollegiet är expert på just datalogiskt tänkande och sen lär de övriga kollegorna). Kursmaterial, modeller, simuleringar, webbsidor och sociala medier med inspiration och självstudiemöjligheter för lärare är andra aktiviteter som förbereder lärare för att kunna undervisa i datalogiskt tänkande. Även Mannila m.fl. (2014, s. 25) poängterar vikten av professionellt utvecklande kurser med hög kvalitet och motiverande och engagerande material för lärare.

3.1 Fortbildning för lärare

Finländska lärare har rätt till fortbildning varje år (FSL, 2017). I många länder finns det brist på resurser och experter för fortbildning av lärares datalogiska tänkande. Det är viktigt att lärarna får praktisk övning som åtminstone delvis ska ske ansikte mot

ansikte. I lärarutbildningar måste det göras utrymme för datavetenskapliga inslag, gärna i samband med de naturvetenskapliga ämnena, så att lärarstuderande kan specialisera sig på det. Lärare ska kunna undervisa i datavetenskap på en åldersanpassad nivå. Eleverna ska få planera, skapa och experimentera i miljöer som är viktiga för dem. (Bocconi, Chioccariello, Dettori, Ferrari & Engelhardt, 2016.)

Det finns fem faktorer som gör att fortbildningar lyckas utveckla lärares professionalism. För att lärares undervisning ska förbättras och för att deras elever ska lära sig mera måste fortbildningens längd för det första vara tillräcklig. Man rekommenderar åtminstone 50 timmar och gärna flera sessioner utspridda över en längre tid. För det andra behöver lärare stöd över en längre tid för att kunna implementera nya undervisningsmetoder. För det tredje måste fortbildningen ha aktiva inlärningsmetoder så att läraren faktiskt lär sig något konkret. För det fjärde borde man väva in den pedagogiska innehållskunskapen i diskussioner om specifika ämnen och hur man tacklar barns olika förutfattade meningar, inlärningsstilar och intressen. För det femte måste kommunikationen mellan bildningsnämnder, skolledare och lärare ständigt vara öppen, och lärare måste känna att de får stöd uppifrån. (Menekse, 2015.)

Det har visat sig att samarbete mellan högre utbildningsinstitutioner och lokala skolorganisationer saknas helt eller är väldigt litet. Det betyder att fortbildningar för datalärare inte utvecklas optimalt, eftersom fältet och experterna inte samarbetar. Det saknas även fortbildningsutbildare, eftersom människor med rätt kunskap hellre jobbar inom industrin där lönen är högre. De som ger fortbildning jobbar inom en viss fakultet och har inte kontakt med andra ämnen som t.ex. naturvetenskap eller matematik. (Menekse, 2015.)

3.2 Forskning om lärares programmeringsfärdigheter

Lärares attityder, kunskaper och värderingar påverkar elevernas värderingar och kunskapsutveckling i datavetenskap (Munson, Moskal, Harriger, Lauriski-Karriker & Heersink, 2011). Därför är det viktigt att lärare har goda attityder till och goda kunskaper i datavetenskap.

Faktorer som utmanar lärares programmeringsundervisning

Lärare beskriver flera utmaningar som finns när det kommer till att undervisa i programmering. Det finns utmaningar med innehållet. Lärare beskriver att de har för lite kunskap och att de saknar programmeringserfarenhet. Det finns också pedagogiska utmaningar. Eleverna kan lätt göra något annat på datorn än att programmera. Alla problem som uppstår undre en lektion kräver individuella lösningar och det är svårt för lärarna att differentiera undervisningen. Lärare ser även bedömningen som en utmaning, eftersom det inte finns något bedömningsverktyg och det är svårt att veta om eleverna förstår de datalogiska begreppen eller om det är en kamrat som har hjälpt dem. (Sentance & Csizmadia, 2017; Yadav, Gretter, Hambrusch & Sands, 2016.)

Det kommer även fram att datalärare är dåligt förberedda. Det finns inga färdiga program som utbildar datalärare och lärarna upplever även tidsbrist. Datalärare känner sig isolerade, eftersom det ofta finns bara en datalärare per skola, vilket gör att det inte finns några stödgrupper och ingen man kan utbyta idéer med. Därför borde man forma stödgrupper utanför skolan, så att datalärarna får känna att de tillhör en grupp. Lärarna beskriver även tekniska utmaningar. Lärarna önskar sig därför en lokal IT-stödgrupp, snabbare uppdatering av programvaror och nya datorer. Dessutom upplever lärare att det stora utbudet av material på Internet är svårt att kvalitetsbestämma. (Sentance & Csizmadia, 2017; Yadav m.fl., 2016.)

Faktorer som underlättar lärares programmeringsundervisning

Lärare nämner flera faktorer som gör programmeringsundervisningen mera fungerande. Vissa lärare nämner analogprogrammering, vilket betyder att eleverna programmerar utan en dator. Lärare upplever också att ett gott samarbete mellan eleverna underlättar programmeringen. Då kan eleverna programmera parvis eller så kan de fungera som varandras mentorer, alltså kompis mentorskap. Ett utvecklat datalogiskt tänkande är en annan sak som lärare upplever att gör programmeringen lättare i klassen, alltså att eleverna kan lösa problem genom att dela upp ett problem i mindre delar och sedan skriva algoritmer. Programmeringsuppgifter med stöd och en rik kontext, till exempel programmeringsuppgifter som kan finnas i den riktiga världen är en annan sak som lärarna upplever att fungerar. (Sentance & Csizmadia, 2017.)

Det har visat sig att lärarstuderande som har fått gå en naturvetenskapligkurs med inslag av robotprogrammering har fått bättre självförtroende att undervisa med robotar och att deras naturvetenskapliga kunskaper och deras förmåga till datalogiskt tänkande har förbättrats (Jaipal-Jamani & Angeli, 2017). Efter att IT-lärare hade gått en kurs med inslag av Scratch, såg man att deras självförtroende gällande nästan alla programmeringsuppgifter hade förbättrats. Man såg även att deras negativa attityder till programmering hade minskat. (Yukselturk & Altiok, 2017.) Även lågstadielärare, som deltog i en 54 timmar lång kurs om Scratch på distans, lärde sig både om datalogiskt tänkande och Scratch. Alla deltagare hade ändå inte lärt sig lika mycket. (Marcelino, Pessoa, Vieira, Salvador & Mendes, 2018.)

Även lärare som har deltagit i kontextrika berättande workshoppar där de har fått pröva på analoga programmeringsmetoder, har blivit mera inspirerade samt fått bättre självförtroende och ökad kunskap om datalogiska begrepp (Curzon, Mcowan, Plant & Meagher, 2014). En annan studie visar också att t.o.m. en dags workshop har fördjupat lärares kunskaper om det datalogiska tänkandets innehåll, pedagogiken kring det, deras teknologikunskaper och deras självförtroende gällande begrepp och praktik (Bower m.fl., 2017). En workshop på två veckor lyckades inte förbättra lärares attityder till programmering (men de hade redan goda attityder från början). Workshoppen lyckades ändå förbättra lärares programmeringsfärdigheter. (Munson, Moskal, Harriger, Lauriski-Karriker & Heersink, 2011.)

Det har också visat sig att datalärare som har blivit handledda, har förbättrat sina didaktiska och datavetenskapliga kunskaper och deras känsla av isolering har minskat (Margolis, Ryoo & Goode, 2017).

4 Scratch – ett visuellt programmeringsspråk

I det fjärde kapitlet introduceras det visuella programmeringsspråket Scratch så att det ska bli lättare för läsaren att förstå metod- och resultatkapitlet. Först beskrivs bakgrunden till Scratch, sedan de olika programmeringsblocken som finns i Scratch och till sist presenteras forskning om Scratch.

4.1 Basfakta om Scratch

Scratch är ett visuellt, även kallat grafiskt, blockbaserat programmeringsspråk. Syftet är att lära barn och unga eller nybörjarprogrammerare att programmera genom att kombinera block samtidigt som de får jobba med personliga meningsfulla projekt tillsammans med andra Scratchanvändare på Scratchs hemsida. De block som ska kombineras har olika färger beroende på deras funktion. Det är omöjligt att kombinera två block som saknar betydelse, vilket gör att nybörjarprogrammerare inte behöver fundera över onödig syntax (jämför textbaserade programmeringsspråk) när de skapar sina första programkoder som kallas för *skript*. Ett skript är ett slags manus, eller en sekvens av instruktioner, som berättar för datorn vad den ska göra. I Scratch drar, släpper och kombinerar man färggranna block så att ett skript bildas. De grafiska 2D-figurerna som man väcker till liv när man programmerar kallas för *sprajts*. Sprajtsen rör sig på en bakgrund som kallas för *scen*. En person som programmerar i Scratch jobbar med ett *projekt*. Ett projekt kan till exempel vara en animerad berättelse, ett spel eller ett simulerat fenomen. Scratch riktar sig främst till 8–16-åringar. (Maloney, Resnick, Rusk, Silverman & Eastmond, 2010.)

Scratch utvecklades av en liten forskargrupp vid Lifelong Kindergarten Group vid MIT Media Lab i USA. Det är ett program och en miljö som började planeras 2003, men som lanserades först 2007. År 2014 lanserades Scratch 2.0 som är en bättre version av Scratch. Scratch är alltså ett gratisprogram som finns tillgängligt för alla på Internet (https://scratch.mit.edu/) och som är översatt till nästan 50 olika språk. År 2009 lanserades även ett nätforum för lärare, Scratch-Ed (http://scratched.gse.harvard.edu/), där de får dela med sig av tips, erfarenheter och

lektionsplaner (Resnick m.fl., 2009). Från början användes Scratch endast i informella lärsituationer, men idag används det även i skolor. (Maloney m.fl., 2010.)

Idén med Scratch är att göra programmeringen mera meningsfull och motiverande för barn och unga. Därför kan olika slags media så som bilder, ljud och musik importeras så att alla kan göra sina projekt personliga. En annan viktig aspekt med Scratch är den sociala kontexten. På Scratchs hemsida kan man dela sina projekt, få och ge feedback och lära sig av andras projekt. Scratch uppmuntrar till samarbete med andra Scratchanvändare. (Maloney m.fl., 2010.)

4.2 Blocktyper i Scratch

Det finna fyra olika typer av block i Scratch (se tabell 2). *Kommandoblocken* har olika färger och kännetecknas av att de har en urgröpning på toppen och en utbuktning på botten, så de passar ihop som pusselbitar. Genom att kombinera kommandoblock bildas en sekvens av kommandon som kallas för en *stack*. Ett *funktionsblock* kan också ha olika färger och de kännetecknas av att de saknar urgröpningar och utbuktningar. Funktionsblocken ser ovala ut och de anger ett värde. Ett *triggerblock* är brunt och har en rundad topp och triggar (utlöser) kommandot i klyftan när den triggande händelsen sker. *Kontrollblocken* är gula och har en öppning som håller fast den inplacerade kommandosekvensen. (Maloney m.fl., 2010.)

Tabell 2. *Blocktyper i Scratch*

	*Sprajt* är en grafiska 2D-figur som man programmerar i Scratch.
	*Kommandoblocken* har en urgröpning på toppen och en utbuktning på botten, så de passar ihop som pusselbitar. Genom att kombinera kommandoblock bildas en sekvens av kommandon som kallas för *stack*.
	*Funktionsblocken* saknar urgröpningar och utbuktningar. Funktionsblocken anger ett värde.
	*Triggerblocken* är bruna med rundad topp och triggar (utlöser) sekvensen som sitter fast under blocket när den triggande händelsen sker.

	*Kontrollblocken* är gula och har en klammer som håller fast den inplacerade kommandosekvensen.

Att ta isär stacks (serier av kommandon) är lätt. Genom att ta tag i blocket som är högst upp i en stack kan man flytta hela stacken. Genom att ta tag i ett block som är i mitten av en stack tar man bort det blocket och alla block under det. (Maloney m.fl., 2010, s. 9.)

4.3 Forskning om Scratch

I studier har man jämfört grupper som har gått en kurs i Scratch före de ska lära sig att programmera i ett textbaserat programmeringsspråk, med grupper som inte har någon programmeringserfarenhet. Det har visat sig att gruppen som har gått en Scratch-kurs lärde sig snabbare, de hade färre svårigheter och de fick en högre kognitiv förståelse för nästan alla datalogiska begrepp. De som hade gått en Scratchkurs anmälde sig oftare till kurser i datavetenskap, de hade högre motivation och bättre självförtroende gällande programmering. Efter avslutad kurs hade båda grupperna ändå samma färdigheter. (Armoni, Meerbaum-Salant & Ben-Ari, 2015.) Man har även undersökt hur effektivt Scratch är för att lära sig programmera över tid. Resultaten visar att Scratchanvändarnas sociala färdigheter ökar, men att deras tekniska kunnande inte ökar över tid. Det kan bero på att duktiga programmerare slutar programmera i Scratch. I medeltal programmerar användarna tre månader i Scratch före de slutar. (Scaffidi & Chambers, 2012.)

En studie visar att elever i årskurs 5 och 6 i Spanien blev bättre på att förstå och använda datalogiska begrepp och datalogiska handlingsstrategier efter att de hade programmerat i Scratch. De sade att Scratch var motiverande och roligt, och att de tyckte att det var bra att jobba med projektet för det var användbart och gjorde lärandet aktivt. (Sáez-López, Roman-González & Vázquez-Cano, 2016.) Grundskolelever lärde sig datalogiska begrepp när de fick designa spel i Scratch. Det styrker påståendet att grafiska programmeringsspråk som är anpassade för elevernas ålder hjälper elever att utveckla datalogisk begreppskunskap. Processen att designa och testa på de andras

spel ledde till en fortsatt designprocess. Interaktionen mellan elever och lärare, samt elever och elever var viktig för den fortsatta designprocessen, eftersom de både delade spel, idéer, begrepp och strategier. (Baytak & Land, 2011.)

5 Metod

I det femte kapitlet redogörs för studiens metodologiska tillvägagångssätt. Först beskrivs studiens syfte och forskningsfrågor. Sedan bäddas studien in i Wengers (1998) teorier om lärande som praktikbaserad gemenskap. Därefter beskrivs fallstudien som metod, datainsamlingsprocessen samt bearbetningen och analysen av data. Avslutningsvis presenteras studiens tillförlitlighet, trovärdighet och etiska aspekter.

5.1 Syfte och forskningsfrågor

Syftet med studien är att studera finlandssvenska grundskolelärares datalogiska tänkande inom ramen för nätkursen *Programmering i grundskolan – vad, hur och varför?*, som ges som fortbildningskurs för lärare. Syftet konkretiseras i följande forskningsfrågor:

1. Vilka datalogiska begrepp har lärarna använt sig av i sina Scratchprojekt (kursuppgift 5) i en fortbildningskurs om programmering?
2. Vilka datalogiska handlingsstrategier och perspektiv nämner lärarna att de har använt sig av i sina kommentarer till Scratchprojekten (kursuppgift 6) i en fortbildningskurs om programmering?

5.2 Kvalitativ fallstudie

Syftet med en kvalitativ forskningsstrategi är att på djupet förstå innebörden av en viss företeelse eller upplevelse. Den kvalitativa forskaren är det främsta instrumentet för datainsamlingen och dataanalysen och hen är mera intresserad av processer än av resultat. Därför säger man att den kvalitativa forskaren har ett tolkande synsätt. (Bryman, 2011, s. 40–41; Merriam, 1994, s. 30–33.) I den här studien ville jag förstå hur utvecklat lärares datalogiska tänkande är efter en grundkurs i programmering.

Praktikbaserad gemenskap som utgångspunkt för analys av lärares lärande

Imsen (2006, s. 307, 332) beskriver Vygotskijs och Säljös sociokulturella syn på lärande. De ser lärande som en process som alltid sker i samspel med den sociala omgivningen. Den sociokulturella synen är intresserad av hur den sociala gemenskapen, kulturen och språket lägger grunden för utveckling och lärande.

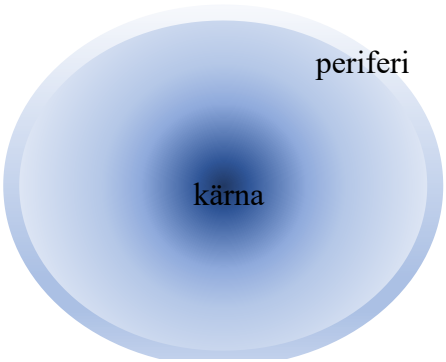
Wenger (1998, s. 3) har studerat lärande i större kontexter och har skapat en social teori om lärande som heter *Communities of Practice*, men som har översatts till svenska med termen *praktikbaserad gemenskap* (Eklöf, 2005). I Wengers (1998) teori om lärande ses lärande som något som utgår från ett socialt deltagande i en gemenskap. Alla människor tillhör många olika praktikbaserade gemenskaper samtidigt, t.ex. i hemmet, skolan och via hobbyer. En praktikbaserad gemenskap behöver inte nödvändigtvis vara rumsligt och tidsmässigt sammanhållen. Det betyder att även distansutbildningar är praktikbaserade gemenskaper. I min studie kunde nätkursen, kursuppgifterna och kursbloggen ses som en praktikbaserad gemenskap, där nätkurslärarna tillsammans formade ny kunskap genom att de tog del av material och i praktiken utförde uppgifter och redovisade dem för varandra.

Enligt Wenger (1998, s. 5) finns det fyra delar som utgör en förutsättning för lärande i den praktikbaserade gemenskapen: innebörd, gemenskap, praktik och identitet. Med *innebörd* (eng. *meaning*) menar Wenger att lärande ger erfarenheter som förändrar och utvecklar förmågor, vilket gör att man upplever lärandet som meningsfullt. För att lära sig ska man delta aktivt i den praktikbaserade gemenskapen och det gör att man lyckas bygga in sin kunskap i artefakter. I den här studien deltog lärarna i nätkursgemenskapen när de tog del av materialet på nätkursens hemsida. Samtidigt byggde de in sin kunskap i artefakter när de utförde kursuppgifter och rapporterade dem i kursbloggen, så att alla fick ta del av allas kursuppgifter. De utvecklade även nya verktyg som hjälpte dem att tala om programmering när nya programmeringsbegrepp blev en del av deras vokabulär.

Med *gemenskap* (eng. *community*) menar Wenger (1998, s. 5) att lärande skapar tillhörighet. För att kunna lära sig måste man veta hur man kommunicerar i gemenskapen och hur man kan ge och få hjälp. Alla deltagare inom en praktikbaserad gemenskap måste inte interagera med alla andra deltagare. Nätkursens hemsida,

nätforum och kursbloggen utgjorde grunden för lärarnas programmeringsgemenskap i den här studien. När lärarna tog del av kursmaterialet och gjorde kursuppgifterna utvecklade de ett gemensamt sätt att kommunicera på och de använde sig av en specifik vokabulär när de beskrev sina Scratchprojekt. Lärarna kunde även chatta med kurshållarna vid behov.

Med *praktik* (eng. *practice*) menar Wenger (1998, s. 5) att en person lär sig samtidigt när hen gör saker. En handling kan inte ske utan en tanke, vilket betyder att hela personen deltar (fysiskt och psykiskt) när hen lär sig. I en praktikbaserad gemenskap finns det nybörjare som deltar i gemenskapen mera i periferin (utkanten av cirkeln) och experter som deltar i gemenskapens kärna (cirkelns mittpunkt) och allt där emellan (se figur 1). I den här nätkursen hade lärarna olika mycket kunskap om programmering i början av kursen, men man kan anta att alla befann sig någonstans i periferin av programmeringsgemenskapen. Kurshållarna befann sig närmare kärnan eftersom de hade mera kunskap om programmering. Alla människor är samtidigt medlemmar av flera praktikbaserade gemenskaper (via t.ex. jobbet och hobbyer). Den praktiska delen av lärande innebär att personerna som deltar i samma gemenskap utvecklar ett ömsesidigt engagemang, en förståelse för och en vilja att vara lik resten av verksamheten. Alla lärare på nätkursen engagerade sig i Scratch i kursuppgift 5 och de lärde sig hur Scratch fungerade. De försökte lära sig mera om blockbaserad programmering i Scratch så att de bättre skulle passa in i programmeringsgemenskapen.



Figur 1. Olika slags deltagande i en praktikbaserad gemenskap.

Med *identitet* (eng. *identity*) menar Wenger (1998, s. 5) att varje ny sak som en person lär sig omformar hens identitet. Ju mera programmeringskunskap och -färdigheter som

lärarna tillägnade sig i den här studien, desto mera förknippade de sig själva med programmeringsgemenskapen. Lärarnas identitet förändrades och de började identifiera sig med andra som programmerar.

Sammanfattningsvis kan man säga att man inte kan skilja kunskap från praktik, för det går inte att kunna utan att göra, vilket gör lärande till ett socialt fenomen. I nätkursen kunde man därför se lärandet som ett ökat deltagande i den gemensamma programmeringspraktiken. Jag hade ingen aning om vad som rörde sig i lärarnas huvud när de programmerade i Scratch, men jag såg deras ökade aktivitet i programmeringsgemenskapen som produkter (Scratchprojekten), och det var dem jag analyserade i den här studien.

Fallstudien som forskningsmetod

Fallstudier används då forskaren vill få djupgående kunskap om en viss situation. Fokus ligger på att upptäcka mera än att bevisa. Fallstudier lämpar sig på områden som kräver förståelse, innan man kan förbättra praktiken. (Merriam, 1994, s. 9.) I den här studien undersökte jag lärares datalogiska tänkande efter en programmeringsfortbildningskurs. Alla lärare hade deltagit i samma fortbildning på nätet och jag undersökte i vilken grad deras datalogiska tänkande syns i kursuppgift 5 och 6.

Kvalitativt inriktade fallstudier har enligt Merriam (1994, s. 25–27) fyra grundläggande egenskaper. Den första egenskapen är att fallstudier är *partikularistiska*. Det betyder att en fallstudie fokuserar på en viss situation, en händelse, ett fenomen eller en person, även kallat ett avgränsat system. Denscombe (2016 s. 92) kallar det för en (eller få) undersökningsenheter. I min studie undersökte jag en specifik grupp lärares Scratchprojekt och tillhörande kommentarer. Alla lärare hade gått samma nätkurs om grunderna i programmering samtidigt, och alla har gjort kursuppgift 5 och 6. Nätkursen var min kontext och deltagarnas slutuppgift var mitt fall.

Den andra egenskapen som fallstudier har enligt Merriam (1994, s. 25–27) är att de är *deskriptiva*. Det betyder att beskrivningen av företeelsen som har undersökts ska vara omfattande och tät. Den tredje egenskapen som fallstudier har är att de är *heuristiska*.

Det betyder att undersökningens resultat ska förbättra läsarens förståelse av företeelsen genom en ingående och kritisk förklaring. I resultatdelen beskriver jag lärarnas datalogiska tänkande ingående och i diskussionsdelen återkopplar jag till tidigare forskning om datalogiskt tänkande. Det ger förhoppningsvis läsaren en djupare förståelse för nätkurslärarnas datalogiska tänkande.

Den fjärde egenskapen som Merriam (1994, s. 25–27) beskriver att fallstudier har är att de är *induktiva*. Det betyder att resultaten endast finns i den information som forskaren har tillgång till. I den här studien utgick jag endast ifrån Scratchprojekten och de tillhörande kommentarerna. Om jag skulle ha samlat in data även på andra sätt så skulle jag med större säkerhet och utförlighet kunnat beskriva nätkurslärarnas datalogiska tänkande.

Denscombe (2016, s. 94) belyser vikten av att fallet ska finnas i sin naturliga miljö och att fallet inte är ett experiment, där forskaren målmedvetet påverkar variabler. Jag kunde inte alls påverka nätkurslärarnas Scratchprojekt eller deras kommentarer, eftersom jag fick tillgång till projekten och kommentarerna först två år efter att kursen hade ordnats. Denscombe (2016, s. 93), men även Patel och Davidson (2011, s. 57) och Merriam (1994, s. 24) påpekar att det inte finns några specifika metoder för datainsamling och dataanalys. Fallstudien tillåter och uppmuntrar till att forskaren använder flera olika slags data, för att på det viset ge en så exakt bild som möjligt av företeelsen. I den här studien analyserade jag både Scratchprojekten och de tillhörande kommentarerna. Merriam (1994, s. 22) uppmuntrar till att resultaten ska presenteras kvalitativt i form av ord och bilder hellre än med siffror. Resultaten redovisades i den här studien med skärmdumpar från Scratchskoderna och citat från kommentarerna. Det finns även tabeller som ger en översikt över fördelningen av data.

Frågan om man kan generalisera fallstudier är viktig. Varje fall är unikt, men varje fall ingår även i en bredare kategori. Hur mycket man kan generalisera inom kategorin beror på hur många gemensamma nämnare de olika fallen delar. När forskaren redovisar resultaten måste hen ge tillräckligt med detaljer om fallet så att läsaren själv kan göra en välavvägd bedömning av huruvida resultatet är generaliserbart för andra fall inom samma kategori eller inte. (Denscombe, 2016, s. 100–102, Patel & Davidson, 2011, s. 57.) Min studies resultat kan inte generaliseras nämnvärt. Samma

nätfortbildningskurs har dock ordnats två gånger till efter våren 2016, så eventuellt skulle mina resultat kunna jämföras med de nya nätkurslärarnas Scratchprojekt och kommentarer. Min studie ger ändå exempel på tankar, handlingssätt, kunskaper och funderingar som även andra lärare kan ha om programmering i grundskolan.

5.3 Dokument som datainsamlingsmetod

Ofta förknippas en viss datainsamlingsmetod med en viss forskningsansats, men kopplingen är inte vattentät. Alla datainsamlingsmetoder har sina starka och svaga sidor. Att välja datainsamlingsmetod handlar om att välja den metoden som är bäst lämpad för att få svar på forskningsfrågorna. Fallstudier uppmuntrar till att använda olika slags data. (Denscombe, 2016, s. 93, 103.) I min studie består datamaterialet av personliga, virtuella och visuella dokument, där ingen utomstående har kunnat styra produktionen av dokumenten. Att datamaterialet består av dokument betyder att materialet går att läsa och att det finns bevarat och tillgängligt för analys. Dokument och arkivmaterial kan inte påverkas av forskarens värderingar och uppfattningar. Det gör dokumenten till ett icke-reaktivt material och det förhöjer trovärdigheten hos data. (Bryman, 2011, s. 488–494, 499–500.)

Jag fick tillgång till den här studiens datamaterial via kurshållaren för fortbildningskursen *Programmering i grundskolan – vad, hur och varför?*. Samma kurshållare hjälpte mig även att skicka e-post till fortbildningsdeltagarna (lärarna) och frågade dem om deras samtycke för att medverka i min studie. Fortbildningskursen ordnades våren 2016 och i januari 2018 blev fortbildningsdeltagarna tillfrågade om de ville medverka i min studie via ett e-postmeddelande. Via kurshållaren fick jag även tillgång till nätkursens hemsida, där allt kursmaterial fanns, och till deras kursblogg, dit alla kursuppgifter hade laddats upp. I min studie bestod datamaterialet alltså av en produkt som lärarna själva hade skapat utgående från nätkursens kursuppgifter 5 och 6. Du kan läsa mera om nätkursen i kapitel 1.4.

I min studie valde jag att använda Scratchprojekten och de tillhörande kommentarerna som mitt datamaterial, eftersom jag kom åt Scratchprojekten via den insatta länken på kursbloggen. Det gjorde att jag kunde pröva projekten och se projektens koder. Jag

gjorde ett så kallat bekvämlighetsurval, vilket betyder att jag tog emot allt datamaterial som jag fick för att spara tid (Denscombe, 2016, s. 77). Av 15 Scratchprojekt som fanns rapporterade på kursbloggen, analyserade jag 13 projekts koder för att få reda på vilka datalogiska begrepp nätkurslärarna har använt sig av i kursuppgift 5 (forskningsfråga 1). Orsaken till att jag exkluderade två projekt från forskningsfråga 1, var att ett projekts direktlänk inte fungerade och ett annat projekts kod inte syntes.

Jag analyserade alla 15 inlägg på kursbloggen som beskrev hur det var att jobba med Scratchprojekten. Det betyder att jag analyserade 15 lärares svar på kursuppgift 6, för att få reda på vilka datalogiska handlingsstrategier och perspektiv nätkurslärarna har använt sig av när de programmerade i Scratch (forskningsfråga 2). Kommentarernas längd varierade väldigt mycket.

5.4 Bearbetning och analys av data

Enligt Denscombe (2016, s. 342) är syftet med dataanalysen att förbättra sin förståelse för datamaterialet. Genom att i detalj undersöka datamaterialet kan man beskriva beståndsdelarna, förklara funktionerna eller tolka betydelsen av datamaterialet. Syftet med min analys var att beskriva nätkurslärarnas datalogiska tänkande. Jag började min analysprocess med att undersöka allt material som fanns på nätkursens hemsida. Efter det läste jag kursdeltagarnas egna kommentarer om sina Scratchprojekt och prövade deras projekt. Parallellt med genomgången av kommentarerna och projekten sammanställde jag ett dokument med länkar till kommentarerna och projekten samt skrev ner mina egna spontana tankar om deras Scratchprojekt, för att inte låta mina egna förutfattade meningar styra analysarbetet.

Analysverktyget

För att kunna analysera kursdeltagarnas datalogiska tänkande skapade jag ett analysverktyg utgående från Brennans och Resnicks (2012) definition av datalogiskt tänkande (se bilaga 2). Brennan och Resnick spjälker upp det datalogiska tänkandet i tre dimensioner: begrepp, handlingsstrategier och perspektiv. Med dimensionen *datalogiska begrepp* menas begrepp som en programmerare kommer att stöta på och lära sig använda när hen programmerar. I modellen beskrivs sju olika begrepp:

sekvens, loop, händelse, parallellism, villkorssats, operator och data. Med dimensionen *datalogiska handlingsstrategier* förstås de handlingsstrategier som en programmerare utvecklar över tid när hen blir en duktigare programmerare. I modellen beskrivs fyra olika handlingsstrategier: att gå stegvis och upprepa, att testa och felsöka, att återanvända och remixa och att abstrahera och skapa modeller. Med dimensionen *datalogiska perspektiv* menas de nya perspektiv som en programmerare utvecklar i och med att hens programmeringsfärdigheter utvecklas. I modellen beskrivs tre olika perspektiv: att få uttrycka sig, att förenas och att ifrågasätta. I bilaga 1 kan du läsa mera ingående om hur underkategorierna inom de olika dimensionerna definieras.

Analysprocessen

Jag använde mig av en Exceltabell för att strukturera upp min analys av Scratchprojekten och de tillhörande kommentarerna (se bilaga 2 för exempel). Först matade jag in mitt analysverktyg i Exceltabellen, så att varje rad motsvarade en underkategori av datalogiskt tänkande, till exempel rad 1: sekvens, rad 2: loop. Sedan matade jag in namnet på varje projekt, så att varje kolumn motsvarade ett projekt, t.ex. kolumn 1: projekt 1, kolumn 2: projekt 2. Sedan analyserade jag systematisk alla Scratchprojekt.

Jag började med att analysera vilka datalogiska begrepp jag kunde hitta i varje projekt. Jag gick alltså enskilt igenom varje Scratchprojekts kod och fyllde samtidigt i Exceltabellen huruvida projektet innehöll en sekvens, loop, händelse o.s.v. eller inte och gav sedan ett exempel på en sekvens, loop, händelse o.s.v. (se bilaga 2). Den här delen av analysen utgjorde basen för att jag skulle kunna besvara min första forskningsfråga. I resultatkapitlet beskriver jag varje underkategori kvantitativt, genom att ange antal lärare som har använt ett begrepp i sin kod, och kvalitativt, genom att beskriva underkategorins innebörd samt exemplifiera den med skärmdumpar från Scratchprojektens koder.

När jag hade studerat alla Scratchkoder angående vilka datalogiska begrepp jag kunde hitta i vilken kod fortsatte jag med att analysera Scratchprojektens tillhörande kommentarer. På samma vis gick jag då enskilt igenom alla kommentarer som hörde till ett projekt och fyllde samtidigt i Exceltabellen med citat som exemplifierade olika datalogiska handlingsstrategier och perspektiv som nätkurslärarna hade använt sig av

(se bilaga 2 för exempel). Den här delen av analysen ligger till grund för svaret på min andra forskningsfråga. I resultatkapitlet beskriver jag varje underkategori kvantitativt, genom att ange antal lärare som har skrivit en kommentar som passade in i en underkategori (handelsstrategier/perspektiv), och kvalitativt, genom att beskriva underkategorins innebörd och exemplifiera den med citat från lärarnas kommentarer.

Jag har endast analyserat det datamaterial som jag har haft tillgång till. Kommentarernas längd varierade. Det är därför omöjligt att uttala sig om nätkurslärarnas datalogiska handlingsstrategier och perspektiv som hade väldigt korta kommentarer. Kompletterande datainsamling i form av intervjuer, enkäter, observationer, skärminspelningar och ljudupptagningar skulle ha gett en bredare bild av nätkurslärarnas datalogiska tänkande.

5.5 Tillförlitlighet, trovärdighet och etiska aspekter

Det är svårt att bedöma den kvalitativa forskningens trovärdighet enligt samma kriterier som man bedömer den kvantitativa forskningen. Det är omöjligt att upprepa exakt samma kvalitativa studie, eftersom den sociala inramningen aldrig kan bli den samma och tiderna förändras. Den andra orsaken är att det finns en risk att den kvalitativa forskaren blir väldigt nära involverad i datainsamlingen och dataanalysen, och då är det svårt för en annan forskare att producera identiskt data och identiska slutsatser. Men det finns ändå ett behov av att verifiera kvalitativ forskning. (Denscombe, 2016, s. 409.)

Reliabiliteten kallas enligt Denscombe (2016, s. 411) i kvalitativa sammanhang för tillförlitlighet eller pålitlighet. Tillförlitligheten brukar delas in i en extern och en intern del enligt Bryman (2011, s. 352). Extern tillförlitlighet handlar om i vilken mån studien kan upprepas med samma resultat. Det är svårt i en kvalitativ studie eftersom det är i princip omöjligt att återskapa sociala miljöer, situationer och beteenden. Min studie hade eventuellt kunnat upprepas med liknande resultat om man skulle analysera de två andra nätfortbildningskurserna med samma namn som gått efter år 2016. Intern tillförlitlighet handlar om att medlemmarna i ett forskarlag kommer överens om hur de tolkar det som de ser och hör. Detta är enklare om forskaren är ensam, samtidigt

som forskaren då bara har ett perspektiv ur vilket hen tolkar data. Jag valde att tolka mina resultat med hjälp av mitt analysverktyg som jag utvecklade ur Brennans och Resnicks (2012) definition på datalogiskt tänkande. Analysverktyget är tydligt beskrivet och underkategorierna är exemplifierade, vilket gör min tolkning synlig.

Validiteten kallas enligt Denscombe (2016, s. 410) i kvalitativa sammanhang för trovärdighet, eftersom man i kvalitativ forskning inte kan bevisa att man objektivt sett har exakt och träffsäkert data. Trovärdigheten handlar alltså om huruvida forskaren mäter det hen anser sig mäta. Bryman (2011, s. 532) delar in trovärdigheten i en extern och en intern del. Extern trovärdighet handlar om i vilken mån forskaren kan generalisera resultatet till andra sociala miljöer och situationer. I kvalitativa studier är den externa trovärdigheten ofta ett problem, eftersom man ofta använder sig av fallstudier och ett begränsat urval. Min studie är en fallstudie vilket gör att resultaten inte går att generalisera på alla finlandssvenska lärare, men vissa tankar som nätkurslärarna har haft kan nog finnas även hos andra lärare. Intern trovärdighet betyder att forskarens observationer och de teoretiska idéer som hen utvecklar stämmer överens. I kvalitativa studier brukar den interna trovärdigheten vara en styrka, eftersom forskaren på djupet har studerat något en längre tid.

Det finns inget lätt sätt att ta reda på hur tillförlitlig och trovärdig en kvalitativ studie är. För forskarens del handlar det om att synliggöra processen, besluten och resultaten på ett sådant sätt att läsaren själv får bedöma studiens tillförlitlighet och trovärdighet (Denscombe, 2016, s. 412–413). Jag har i mitt metodkapitel beskrivit mitt tillvägagångssätt så tydligt och transparent som möjligt. I resultatkapitlet är resultaten utförligt beskrivna och exemplifierade. Därför anser jag att studien har en hög tillförlitlighet och trovärdighet.

Det finns fyra olika etiska principer som borde uppfyllas för att en studie ska vara etiskt korrekt utförd enligt Bryman (2011, s. 131–132). *Informationskravet* betyder att forskaren ska informera deltagarna om studiens syfte och moment, att medverkan är frivillig och att de får hoppa av när som helst. *Samtyckeskravet* betyder att deltagarna själva får bestämma om de deltar eller inte deltar i studien. *Konfidentialitetskravet* betyder att forskaren har skyldighet att skydda informanternas identiteter och att inga

personuppgifter får komma ut. *Nyttjandekravet* betyder att deltagarnas uppgifter och svar endast får användas för forskningsändamål.

I samband med att nätkursen *Programmering i grundskolan – vad, hur och varför?* ordnades för första gången år 2016, hade kurshållarna inte frågat om nätkurslärarna ville delta i en studie. Därför hjälpte den ena kurshållaren mig att i januari 2018 skicka ett e-postmeddelande åt nätkurslärarna där min studie, det frivilliga deltagandet, konfidentialitetskravet och nyttjandekravet beskrevs. Om någon av nätkurslärarna motsatte sig att delta i studien skulle de svara nej på e-postmeddelandet. En nätkurslärare svarade nej på e-postmeddelandet och exkluderades därför. Jag var alltså inte med när datamaterialet samlades in, men jag har fått tillgång till det senare. Jag vet personligen inte heller vem alla nätkurslärarna är, eftersom flera har lämnat in sina kursuppgifter under pseudonymer. I resultatkapitlet nämns inga namn eller pseudonymer, utan jag refererar till Scratchprojekten som projekt nummer 1, 2, 3 och så vidare. Därför anser jag att studien är etiskt korrekt utförd.

6 Resultat

I det sjätte kapitlet redovisas först Scratchprojektens olika typer och teman. Därefter redovisas studiens resultat i två delar utgående från de två forskningsfrågorna. Syftet med avhandlingen var att studera finlandssvenska grundskolelärares datalogiska tänkande inom ramen för nätkursen *Programmering i grundskolan – vad, hur och varför?*, som har getts som fortbildningskurs för lärare. I den första delen av resultatkapitlet redogörs för vilka datalogiska begrepp som nätkurslärarana har använt sig av i sina Scratchprojekt. I den andra delen av resultatkapitlet redogörs för vilka datalogiska handlingsstrategier och perspektiv som nätkurslärarna har nämnt att de har använt sig av i sina kommentarer när de har programmerat i Scratch. Avslutningsvis sammanfattas resultaten.

Lärarnas Scratchprojekt är utformade både som spel, test och presentationer (se tabell 3). De flesta lärare gjorde någon form av spel eller test med temat matematik. Dessa projekt berör multiplikationstabellerna, delbarhetsreglerna, binära tal, geometriska figurer, de fyra räknesätten, tiokompisar och primtal. Två lärare satsade på projekt med temat språk och ordkunskap. En lärare satsade på ett naturvetenskapligt tema och gjorde en frågesport om grundämnen. En annan lärare gjorde ett bollspel, en tredje ett morsdagsspel och en fjärde en presentation om läroplanen.

Tabell 3. *Olika typer av Scratchprojekt och deras teman*

Scratchprojektens: TYP → TEMA ↓	Spel	Test	Presentation
Matematik	5	3	-
Språk	-	2	-
Naturvetenskap	1	-	-
Övrigt	3	-	1

Både spelen och testen är interaktiva projekt där den som prövar på projekten själv måste medverka. I spelprojekten finns element som slumpning av frågor, olika nivåer, tidsramar eller poängmål och förmågan att ”dö”. I projekten som är utformade som test finns också element som poängräkning och slumpning av frågor, men testen har ett slut. Projektet som är en presentation har inga egentliga element utan när projektet har startats byts texten på skärmen automatiskt ut.

6.1 Nätkurslärarnas användning av datalogiska begrepp

Genom att studera nätkurslärarnas Scratchkoder kan man se vilka olika block de har använt sig av när de har programmerat sina projekt. Block av olika färger och former har olika funktioner i Scratch och de olika blocken kan därför sägas representera olika begrepp. Den här delen av resultatkapitlet är strukturerat enligt Brennans och Resnicks (2012) sju olika datalogiska begrepp som en person med ett utvecklat datalogiskt tänkande använder sig av och förstår. Dessa begrepp är: *sekvens*, *loop*, *händelse*, *parallellism*, *villkorssats*, *operator* och *data*. I den här studien har jag analyserat 13 nätkurslärares Scratchprojekts innehåll av datalogiska begrepp.

I 6 av 13 Scratchprojekt återfinns alla sju datalogiska begrepp (se tabell 4). Det betyder att 6 nätkurslärare har använt sig av alla sju datalogiska begrepp. I 6 andra projekt används alla datalogiska begrepp förutom ett. Av dem saknas begreppet *loop* i 2 projekt, begreppet *parallellism* i 3 projekt och begreppet *operator* i 1 projekt. I ett projekt används endast 2 datalogiska begrepp (*händelse* och *sekvens*). Det betyder ändå att majoriteten av nätkurslärarna har använt sig av nästan alla datalogiska begrepp, förutom en deltagare som knappt fick ihop en presentation genom att använda två begrepp. Tabell 4 visar vilka datalogiska begrepp som saknas i nätkurslärarnas Scratchprojekt.

Tabell 4. *Datalogiska begrepp som saknas i nätkurslärarnas Scratchprojekt*

Scratchprojekt → Datalogiska begrepp ↓	1	2	3	4	5	6	7	8	9	10	11	12	13
Sekvens													
Loop					Nej			Nej					Nej
Händelse													
Parallellism		Nej					Nej					Nej	Nej
Villkorssats													Nej
Operator	Nej												Nej
Data													Nej

I Scratchkoden ser begreppet *händelse* ut som bruna triggerblock som har en rundning på toppen (se figur 2 för exempel). Vissa händelse-block har en grön flagga på sig och på andra står det *när denna sprajt klickas på*. Alla nätkurslärare har lyckats programmera den gröna flaggan så att projektet startar. Det betyder att alla

nätkurslärare har använt sig av åtminstone ett händelse-block. Vissa lärare har även använt sig av händelser som startas när man klickar på en specifik sprajt[7]. Alla lärares projekt innehåller också åtminstone en *sekvens*, alltså en serie av instruktioner (se figur 2 för exempel). I koden ser sekvenser ut som kommandoblock som sitter ihop med utbuktningar på botten och urgröpningar på toppen. Man kan jämföra sekvenser med pusselbitar som har kombinerats. De instruktioner som finns i en sekvens börjar automatiskt efter varandra. Figur 2 visar exempel på hur en händelse och en sekvens kan se ut i lärarnas Scratchkoder.

Figur 2. Exempel på händelser och sekvenser i ett Scratchprojekt (projekt 4).

På bilden ses två olika händelse-block som startar en ny händelse: det bruna blocket med grön flagga och det bruna blocket med texten *när denna sprajt klickas på*. På bilden finns även två olika sekvenser: *gå 10 steg*, *studsa vid kanten* och *spela ljudet pop*, *ändra poäng med 1*.

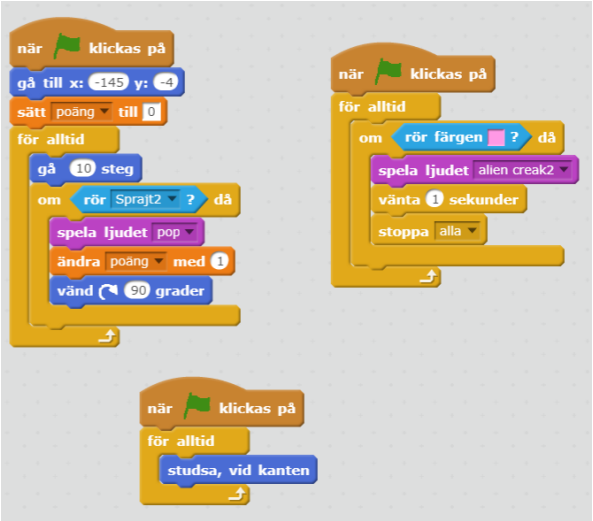
I koden ser *loopar* ut som gula kontrollblock med en klammer (se figur 3 för exempel). Inne i klammern finns det kommando eller den sekvens som man vill upprepa. I de Scratchprojekt som har loopar finns loopkommandon som *upprepa för alltid*, *repetera tills*, *repeterar 3 gånger*, *repeterar tills 5 poäng*, *repeterar tills återstående tid = 0* och *visa för alltid*. Det saknas loopar i 3 projekt, vilket betyder att tre lärare har valt att inte upprepa ett kommando eller en sekvens. Figur 3 visar exempel på hur loopar kan se ut i lärarnas Scratchprojekt.

[7] *Sprajt* är en 2D-figur i Scratch som man programmerar.

Figur 3. Exempel på en loop i ett Scratchprojekt (projekt 6).

På bilden ses ett gult loop-block som ser ut som en klammer med texten *repetera tills*. Inne i klammern finns den sekvens som man vill upprepa och i det här fallet upprepas sekvens tills den återstående tiden är=0.

I koden ser *parallellism* ut som flera händelse-block som har samma startsignal (se figur 4 för exempel). Parallellism innebär alltså att två eller flera sekvenser sker samtidigt, alltså parallellt. För att sekvenser ska ske samtidigt startas de samtidigt, t.ex. genom att händelse-blocket med grön flagga har använts flera gånger i samma projekt. Parallellism saknas i 4 projekt. Det betyder att sekvenserna i dessa projekt alltid sker enskilt. Figur 4 visar exempel på hur parallellism kan se ut i lärarnas Scratchprojekt.



Figur 4. Exempel på parallellism i ett Scratchprojekt (projekt 1).

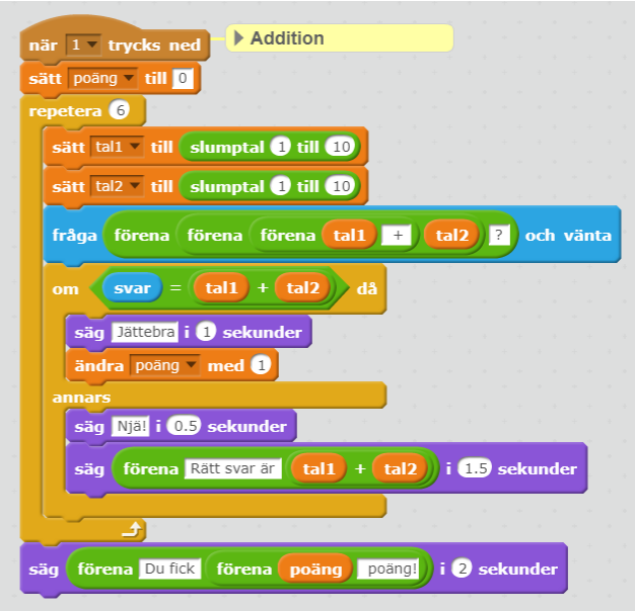
På bilden ser parallellism ut som tre olika bruna händelse-block med grön flagga, vilket betyder att alla tre sekvenser som finns under händelseblocken startar och sker samtidigt när användaren har tryckt på den gröna flaggan.

I koden ser *villkorssatser* ut som gula kontrollblock med en eller två klamrar (se figur 5 för exempel). Att använda sig av en villkorssats gör det möjligt att ange olika alternativ för vad nästa instruktion i en sekvens ska vara. Den sekvens som hör ihop med det alternativ (villkor) som uppfylls startar. På engelska talar man om if-satser. Exempel på villkorssatser som finns i Scratchprojekten är kontrollblocket *om ... då*. Sedan kan man ännu utveckla villkorssatsen med villkoret *om ... då ... annars*. Villkorssatser saknas i 2 av projekten. Det betyder att det bara finns ett alternativ för vad nästa steg i spelet eller testet kan vara i de projekten. Figur 5 visar exempel på hur villkorssatser ser ut i lärarnas Scratchprojekt.

Figur 5. Exempel på en villkorssats i ett Scratchprojekt (projekt 8).

På bilden ses villkorssatsen som ett gult kontrollblock med två klamrar. I det här projektet ska man svara på frågan: *Vad heter katt på spanska?* Om man svarar rätt spelas bl.a. meow-ljudet upp och poängen ändras, men om man svarar fel spelas chomp-ljudet upp och katten säger att det rätta svaret är *gato*.

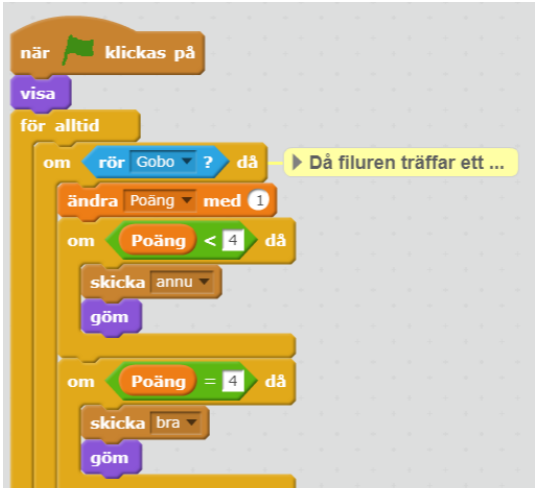
I koden ser *operatorer* ut som gröna funktionsblock utan urgröpningar och utbuktningar (se figur 6 för exempel). En operator är ett block som möjliggör slumpmässiga val ur ett avgränsat område så att användaren kan jämföra eller räkna med olika tal varje gång spelet eller testet används. I Scratch kan man även förena och jämföra ord med varandra. Exempel på operatorer som har använts i Scratchprojekten är *antal kvar <frågor per rond*, *poäng = 10* och *slumptal 1–10*. Operatorer saknas i 2 projekt. De projekt som saknar operatorer har inte programfunktioner som möjliggör matematiska operationer och jämförelser av olika tal och ord. Figur 6 visar exempel på hur operatorer kan se ut i lärarnas Scratchprojekt.



Figur 6. Exempel på olika operatorer i Scratchprojekten (projekt 11).

På bilden ser operatorer ser ut som gröna funktionsblock. I det här projektet används operatorer för att slumpa tal mellan 1 och 10 och för att förena två tal.

I koden ser *data* ut som orangea funktionsblock eller kommandoblock med eller utan urgröpningar och utbuktningar (se figur 7 för exempel). Data är information och det finns olika sätt att lagra informationen på, till exempel som variabler eller som listor. En funktion som datablocken kan ha i Scratchprojekten är att räkna poäng. Alla projekt utom 1 har använt sig av något sätt att lagra data på. Figur 7 visar exempel på hur data kan se ut i lärarnas Scratchprojekt.



Figur 7. Exempel på data i ett Scratchprojekten (projekt 9).

På bilden ser data ut som orangea funktions- och kommandoblock. I det här projektet är poäng en variabel som ändras med 1 då den träffar rätt sprajt. Poängvariabeln används också i samband med en operator och då jämförs variabelns värde med ett tal. Beroende på variabelns värde gäller olika villkor.

Sammanfattningsvis kan man säga att de flesta nätkurslärarna har använt sig av alla eller nästan alla begrepp som en programmerare använder sig av enligt Brennan och Resnick (2012) (se tabell 4). En lärare har endast använt två typer av begrepp och hens projekt är väldigt enkelt och intetsägande. Hens programmeringsfärdigheter är klart sämre än de andras färdigheter, vilket även kom fram i kommentarerna om projektet.

6.2 Nätkurslärarnas användning av datalogiska handlingsstrategier och perspektiv

Genom att studera nätkurslärarnas spontana kommentarer till sina Scratchprojekt får man en bild av hur lärarna upplevde programmeringen i Scratch. Lärarnas kommentarer var fritt formulerade och styrdes inte av något eller någon, och därför benämner jag dem spontana kommentarer. Utgående från Brennans och Resnicks (2012) definition av *datalogiskt tänkande*, analyserade och kategoriserade jag nätkurslärarnas kommentarer i två dimensioner: som datalogiska handlingsstrategier och datalogiska perspektiv. Jag analyserade 15 nätkurslärares kommentarer. Alla lärare har gett en kommentar, men kommentarernas omfattning varierar mycket. Vissa har reflekterat djupare kring programmeringen i Scratch än andra, men resultatet exemplifierar ändå olika typer av strategier, perspektiv och reaktioner som programmering i Scratch kan ge och lära en.

Datalogiska handlingsstrategier

Med *datalogiska handlingsstrategier* menas handlingsstrategier som en programmerare utvecklar för att lyckas med sin programmering. Det finns fyra underkategorier till dimensionen: *stegvis och via upprepning*, *testa och felsöka*, *återanvända och remixa* samt *abstrahera och skapa modeller*. Som man kan se i tabell 5 kunde inte ett enda projekts kommentarer placeras in i alla fyra underkategorier av dimensionen datalogiska handlingsstrategier. Av 15 projektkommentarer kunde 2 kommentarer placeras in i tre underkategorier, 5 kommentarer placeras in i två underkategorier och 8 kommentarer placeras in i en underkategori. Tabell 5 visar vilka datalogiska handlingsstrategier som saknas i nätkurslärarnas spontana kommentarer.

Tabell 5. *Datalogiska handlingsstrategier som saknas i nätkurslärarnas kommentarer*

Scratchprojekt → Datalogiska handlings-strategier ↓	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Stegvis och via upprepning					Nej		Nej				Nej	Nej			
Testa och felsöka			Nej	Nej			Nej	Nej	Nej		Nej	Nej		Nej	
Återanvända och remixa	Nej	Nej	Nej	Nej	Nej	Nej		Nej	Nej	Nej			Nej		
Abstrahera och skapa modeller	Nej	Nej	Nej	Nej	Nej		Nej	Nej	Nej	Nej	Nej	Nej	Nej	Nej	Nej

Med underkategorin *stegvis och via upprepning* förstås att en programmerares handlingsstrategier är cykliska istället för linjära. En programmerare utvecklar en modell, testar den och utvecklar modellen vidare med hjälp av nya erfarenheter och idéer. Flera nätkurslärare beskrev sina programmeringsupplevelser cykliskt: *Uttrycket "VARFÖR blir det inte som jag vill!" sammanfattar ganska bra mitt arbetssätt. Försök och misstag* (projekt 4) och *Jag har provat på både Scratch och Scratch Jr. Till sist valde jag att göra ett spel i Scratch. Jag gillar Scratch, det går att använda bara de allra enklaste funktionerna och vartefter avancera lite* (projekt 11).

Många lärare tyckte att programmeringen var rolig och att de gick bättre ju mera tid de lade på den: *Det var nog jätteroligt fast det var svårt, och jag skulle gärna ha finslipat det ännu mycket mera* (projekt 8) och *Det gick långsamt i början men efter att man hållit på några timmar började man förstå hur man kombinerar de olika bitarna* (projekt 7). Flera lärare beskrev att de hela tiden kom på flera aspekter som de ville utveckla: *När jag trodde att allt var klart visade sig att flera saker behövde justeras* (projekt 6). I underkategorin *stegvis och via upprepning* kan 11 kommentarer placeras in.

Underkategorin *testa och felsöka* beskriver olika handlingsstrategier som är bra att använda när man stöter på problem. För att lösa problem som uppstod nämner nätkurslärare att de har frågat eller berättat för någon annan om problemen: *Några elever fick testa spelet och det behövdes. De hittade direkt några svaga punkter, tex kunde du få poäng hur många gånger som helst när du en gång löst ett tal* (projekt 5). Andra har sökt efter hjälp på internet: *Jag hade vissa problem kring drag-drop-funktionen men hittade genast tips i ett diskussionsforum* (projekt 10). Vissa beskriver att de pausat från programmerandet: *Jag fick ta en paus. När jag satte mig igen insåg*

jag att programmet kunde göras enklare (projekt 6) och *Jag kunde lägga till en ny nivå (level) med en ny bakgrund då bollen rörde sig snabbare men sedan bara slutade spelet vid en viss poäng. Så jag gav upp för tillfället* (projekt 1). I underkategorin *testa och felsöka* kan 7 kommentarer placeras in.

Underkategorin *att återanvända och remixa* är en handlingsstrategi som tar hjälp eller inspiration av program som redan finns i Scratch. Det gör man för att lyckas programmera något som är mera avancerat och mycket större än man skulle klara av att göra om man tvingades göra allting själv från början. Tre nätkurslärare valde att remixa ett projekt: *Jag valde att remixa och utveckla ett projekt som fanns under matematikexemplen i vårt kursmaterial* (projekt 11). Några nätkurslärare övervägde att remixa ett projekt, men valde ändå att göra egna projekt från grunden: *Jag bestämde mig direkt för att göra något inom matematik och kollade först några färdiga som jag ev kunde remixa. Valde ändå att försöka göra eget direkt från grunden* (projekt 15).

För varje Scratchprojekt som har publicerats finns det en knapp som heter *remix*. Det betyder att andra Scratchanvändare kan utgå ifrån och använda samma projekt eller delar av ett projekt om där finns något bra element. Varje Scratchprojekt har även ett eget *remix-träd* som visar originalprojekt som remixade projekt har byggt vidare på. Det ger synlighet åt alla upphovsmän. I underkategorin *återanvända och remixa* kan 5 kommentarer placeras in.

Underkategorin *att abstrahera och skapa modeller* är en grundläggande handlingsstrategi för all planering och problemlösning. När man kan abstrahera och skapa modeller lyckas man dela upp problem i mindre delar. För att kunna bygga stora, avancerade spel måste man först lära sig att bygga små modeller som man sedan kan kombinera till större helheter. En nätkurslärare konstaterade att vissa idéer var alldeles för stora och komplicerade att förverkliga: *Även om idén var mycket enkelt växte programmet snabbt och blev för stort och komplicerat* (projekt 6). Alla 15 nätkurslärare har egentligen klarat av att skapa modeller eftersom alla har skapat ett Scratchprojekt som fungerar. Ändå var det bara 1 kommentar som kunde placeras in i underkategorin *att abstrahera och skapa modeller*.

Sammanfattningsvis kan man säga att flera nätkurslärare har reflekterat över handlingsstrategierna *stegvis och via upprepning* samt *testa och felsöka* i sina spontana kommentarer, men att endast en lärares kommentar kunde placeras in under handlingsstrategin *abstrahera och skapa modeller*, fast alla lärare egentligen har lyckats göra det. Det verkar som att ju abstraktare handlingsstrategin är, desto svårare har lärarna haft att spontant kommentera den handlingsstrategin. Det här resultatet visar på att flera lärare har kommit in i programmeringspraktiken men att de ännu har svårt att uttrycka vad det är de gör.

Datalogiska perspektiv

Med *datalogiska perspektiv* menas de nya perspektiv som en programmerare utvecklar i och med att hens programmeringsfärdigheter utvecklas. Det finns tre underkategorier till den här dimensionen: *att uttrycka sig*, *att förenas* och *att ifrågasätta*. Som man kan se i tabell 6 kan 3 projektkommentarer av 15 placeras in i alla tre underkategorier. Av de resterande projektkommentarerna kan 6 kommentarer placeras in i två underkategorier, 4 kommentarer placeras in i en underkategori och 2 kommentarer kan inte placeras in i någon underkategori. Tabell 6 visar vilka perspektiv som saknas i nätkurslärarnas spontana kommentarer.

Tabell 6. *Datalogiska perspektiv som saknas i nätkurslärarnas kommentarer*

Scratchprojekt → Datalogiska perspektiv ↓	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Att uttrycka sig		Nej			Nej	Nej	Nej				Nej	Nej		Nej	Nej
Att förenas											Nej	Nej	Nej		Nej
Att ifrågasätta			Nej	Nej	Nej		Nej	Nej				Nej	Nej		Nej

Med underkategorin *att uttrycka sig* menar man att datalogiskt tänkande handlar om att själv få planera och skapa teknologi och inte bara konsumera den. Flera nätkurslärare beskrev glädjen och ivern över att programmera: *Sedan blev jag riktigt ivrig och ville göra ett spel som jag som liten spelade på vår första datamaskin på 80-talet. "Tennispelet" där man skulle fånga bollen med brädan och spela mot datorn* (projekt 1) och *Det var väldigt roligt att jobba med Scratch, sådär så att man hela dagen gick och tänkte på hur man skulle lösa programmeringskoden, de få timmarna man inte satt framför datorn för att det var så roligt/beroendeframkallande* (projekt 9). I underkategorin *att uttrycka sig* kan 7 kommentarer placeras in.

Med underkategorin *att förenas* menar man att datalogiskt tänkande och skapande av teknologi är en kreativ läroprocess som kräver sociala sammanhang. Att skapa tillsammans gör att man klarar av att göra mera avancerade saker än man skulle göra individuellt. Flera nätkurslärare beskriver att de fått hjälp online: *Jag använde hjälpmanualerna i programmet för att sätta in stjärnornas och mattans rörelser* (projekt 3), *Följde beskrivningen som Svenska Internetstiftelsens Barnhack hade gjort* (projekt 4) och ... *hittade genast tips i ett diskussionsforum* (projekt 10). En annan nätkurslärare nämner remixandet av projekt som hjälp för att lära sig: *Det är också väldigt bra att man kan se hur andra programmerat och plocka delar från andras projekt* (projekt 14). Vissa nätkurslärare bad de andra kursdeltagarna om hjälp för att förbättra sina egna projekt: *Testa gärna och ge förbättringsförslag. Det är lätt att snöa in i eget tankesätt* (projekt 5).

Några nätkurslärare beskriver en önskan om att få lära sig mera om programmering i Scratch: *Jag tror nog jag skulle vilja gå en kurs till, där man bara behandlar Scratch. Tack till handledarna för hjälpen* (projekt 8). Andra nätkurslärare tänker på undervisningen i programmering och önskar tätare samarbete mellan lärarna: *Som alltid då man börjar undervisa i någonting nytt kommer det naturligtvis att gå åt en massa energi och fantasi! till att skapa lektionsplaner, så här skulle det vara viktigt med samarbete lärare emellan* (projekt 9). I underkategorin *att förenas* kan 11 kommentarer placeras in.

Med underkategorin *att ifrågasätta* menar man att det datalogiska tänkandet ska göra att man känner sig berättigad att ställa frågor om och med teknologin. Vissa nätkurslärare kritiserar någon egenskap i Scratch: *När programfilen blev rätt stor fich jag ganska stora problem att hantera den i editorn. Allting gick mycket trögt när jag släpade och släppte blocken* (projekt 2), *Måste poängtera att bakgrunden med texterna och "välj räknesätt-rutan" inte är gjorda inne i Scratch utan i Photoshop. Jag tyckte bildredigeringen i Scratch var lite bökig* (projekt 11) och *Jag saknade möjligheten att välja typsnittets storlek från en meny* (projekt 10). En nätkurslärare kände frustration över hur noggrann och tydlig man måste vara när man programmerar: *Datorn är verkligen inte smart* (projekt 6).

En del nätkurslärare ställde sig kritiska till hur man ska få in programmering och Scratch i undervisningen: *Ser inte heller direkt hur man kan få in spelprogrammerandet i undervisningen, så att mattedelen får en betydande del av inlärningen* (projekt 9), *Scratch kan nog vara utmanande att använda i åk 1–6. Jag kan tänka mig att vissa elever med ett brinnande intresse nog klarar av Scratch. Men att få alla med på det här kanske inte går* (projekt 1) och *Scratch är säkert något som skulle uppskattas av många elever i åk 7–9. De som har tålamod får säkert tillstånd fina slutprodukter (spel, presentationer, ...). Men det finns säkert även elever som tröttnar efter några minuter och ger upp* (projekt 14). I underkategorin *att ifrågasätta* kan 7 kommentarer placeras in.

Sammanfattningsvis kan man konstatera att majoriteten av nätkurslärarna åtminstone har kunnat ge uttryck för ett datalogiskt perspektiv. Lättast har det varit för lärarna att spontant kommentera perspektivet *att förenas*. Hälften av projektkommentarerna kan även placeras in i perspektiven *att uttrycka sig* och *att ifrågasätta*. Det finns färre kommentarer i dessa perspektiv eftersom man måste ha tillräckligt med kunskap och erfarenhet för att kunna tänka kritiskt och reflektera över att dataanvändning är mera än konsumtion och för att våga ifrågasätta tekniska lösningar.

6.3 Sammanfattning av resultaten

I tabell 7 sammanfattas alla tre dimensioner av nätkurslärarnas datalogiska tänkande som kommer fram i deras Scratchprojekt och kommentarer. Lärarnas datalogiska tänkande analyserades i enlighet med Brennans och Resnicks (2012) definition av datalogiskt tänkande. Utgående från analysen kan man säga att de flesta nätkurslärarna har använt sig av alla eller nästan alla datalogiska begrepp som en person med ett utvecklat datalogiskt tänkande ska kunna använda sig av och förstå, men en lärare har endast använt sig av två olika begrepp i sitt Scratchprojekt. De datalogiska begrepp som oftast saknas i lärarnas projekt är *loop*, *parallellism* och *operator*. De övriga begreppen *sekvens*, *händelse*, *villkorssats* och *data* är kanske mera konkreta och därför har lärarna haft enklare att förstå och använda sig av dem. Det finns variation i hur många gånger samma begrepp har använts av samma lärare.

Knappt hälften av nätkurslärarna har spontant beskrivit två eller tre datalogiska handlingsstrategier i sina projektkommentarer, medan den andra hälften bara har beskrivit en datalogisk handlingsstrategi (se tabell 7). Klart flest kommentarer kan placeras in i handlingsstrategierna *stegvis och via upprepning* (¾) och *testa och felsöka* (½), vilket betyder att de allra flesta lärarna spontant har reflekterat över hur de har jobbat med sitt Scratchprojekt. I de mera abstrakta handlingsstrategierna *återanvända och remixa* samt *abstrahera och skapa modeller* kan mycket färre kommentarer placeras in. Det kan bero på att så få lärare valde att remixa ett annat projekt och på att lärarna inte tänkte på att de skapade modeller och abstraherade, fastän det var det som alla ändå lyckats göra på ett sätt eller annat.

Majoriteten av nätkurslärarna har beskrivit två eller tre olika datalogiska perspektiv i sina spontana projektkommentarer, men två lärare har inte beskrivit ett enda perspektiv (se tabell 7). Perspektivet *att förenas* var lättast för lärarna att spontant beskriva eftersom de har beskrivit olika sätt att fråga hjälp av andra programmerare när de har stött på problem i programmeringsprocessen. Perspektiven *att uttrycka sig* och *att ifrågasätta* var svårare för lärarna att spontant beskriva, vilket kan bero på att det är svårt att känna glädje och iver för något som man anser att är svårt och samtidigt är det svårt att vara kritisk mot något om man inte har tillräckligt med kunskap och erfarenhet av det.

Man kan säga att nätkurslärarnas datalogiska tänkande helt klart håller på att utvecklas och att alla har kommit en bit på vägen. Enligt det här datamaterialet och den här analysmetoden har ingen lärare ett fullt utvecklat datalogiskt tänkande ännu i samband med fortbildningskursens 5:e och 6:e kursuppgift, men alla har ändå kommit en bit på vägen och flera uttrycker iver över att få lära sig mera om programmering överlag och programmering i Scratch.

Tabell 7. *Sammanfattning av nätkurslärarnas avsaknade användning av datalogiskt tänkande*

Scratchprojekt →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Avsaknad användning av datalogiska begrepp:															
Sekvens														X	X
Loop					Nej			Nej					Nej	X	X
Händelse														X	X
Parallellism		Nej					Nej					Nej	Nej	X	X
Villkorssats													Nej	X	X
Operator	Nej												Nej	X	X
Data													Nej	X	X
Avsaknad användning av datalogiska handlingsstrategier:															
Stegvis och via upprepning					Nej		Nej				Nej	Nej			
Testa och felsöka			Nej	Nej			Nej	Nej	Nej		Nej	Nej		Nej	
Återanvända och remixa	Nej	Nej	Nej	Nej	Nej	Nej		Nej	Nej	Nej			Nej		
Abstrahera och skapa modeller	Nej	Nej	Nej	Nej	Nej		Nej	Nej	Nej	Nej	Nej	Nej	Nej	Nej	Nej
Avsaknad användning av datalogiska perspektiv:															
Att uttrycka sig		Nej			Nej	Nej	Nej				Nej	Nej		Nej	Nej
Att förenas											Nej	Nej	Nej		Nej
Att ifrågasätta			Nej	Nej	Nej		Nej	Nej				Nej	Nej		Nej

7 Diskussion

I det sjunde kapitlet diskuteras först avhandlingens resultat och dess koppling till tidigare forskning. Den första forskningsfrågan som handlar om lärares användning av datalogiska begrepp diskuteras först och därefter följer en diskussion om den andra forskningsfrågan som handlar om lärares användning av datalogiska handlingsstrategier och perspektiv. Resultatdiskussionen avslutas med en allmän diskussion om datalogiskt tänkande som begrepp, nätkursen som kurs och funderingar kring vad lärarna gör efter kursen. Därefter diskuteras avhandlingens val av metoder och dess för- och nackdelar. Avslutningsvis ges förslag till fortsatt forskning.

7.1 Resultatdiskussion

Datalogiska begrepp

Den första forskningsfrågan sökte svar på vilka datalogiska begrepp nätkurslärarna hade använt sig av i sina Scratchprojekt. Överlag använde lärarna flitigt alla sju olika datalogiska begrepp i sina Scratchkoder i den här studien. Det är omöjligt för mig att dra några slutsatser huruvida lärarna förstod begreppens innebörd utanför Scratchmiljön, men någon form av förståelse har de ändå fått eftersom de har lyckats använda blocken och begreppen korrekt i sina koder. Scratch tillåter inte att man kombinerar block som saknar betydelse, vilket kan ha hjälpt flera lärare att undgå misstag (Maloney m.fl., 2010).

Alla praktiker har sina specifika termer. Att nätkurslärarna har börjat använda just dessa specifika datalogiska begrepp kan ses som ett slags djupare deltagande i programmeringspraktiken (Wenger, 1998). Det saknades loopar i 3 projekt, parallellism i 4 projekt och operatorer i 2 projekt. Man kan tänka att de projekt som saknar vissa begrepp, har en upphovsman som inte har rört sig lika mycket från cirkelns periferi mot kärnan av cirkeln som de upphovsmäns projekt som har använt alla datalogiska begrepp. Personligen tycker jag att data och operatorer är de begrepp som är knepigast att förstå, eftersom de begreppen är mest abstrakta och de kan se

olika ut i olika koder beroende på vilken slags data man har och vilka matematiska operationer som ska utföras.

Norrena (2013) har kommit fram till att bl.a. lärares otillräckliga färdigheter och bristande attityder försämrar deras undervisning om framtidskompetenserna, som innefattar många samma kompetenser som det datalogiska tänkandet. Därför är det viktigt att lärare själv förstår och kan använda de datalogiska begreppen för att kunna undervisa om begreppen för eleverna. Även forskarna Barr och Stephenson (2011) poängterar att det är viktigt att lärare själva lär sig om vad datalogiskt tänkande är för att sedan kunna undervisa om det. Munson m.fl. (2011) beskriver även vikten av att lärare har goda attityder, kunskaper och värderingar gällande datavetenskap, eftersom de överförs till eleverna.

Datalogiska handlingsstrategier och perspektiv

Den andra forskningsfrågan sökte svar på vilka datalogiska handlingsstrategier och perspektiv som nätkurslärarna hade nämnt att de hade använt sig av i sina spontana kommentarer. Trots att det var stor variation på kommentarernas längd som kunde placeras in i dimensionen datalogiska handlingsstrategier, kunde jag ändå urskilja tydliga skillnader och exempel på hur lärarna hade gått till väga när de programmerade i Scratch. Som Wenger (1998) lyfter fram ska lärandet skapa innebörd och det har det verkligen gjort för lärarna i nätkursen. Lärarna lärde sig till exempel att använda en korrekt programmeringsvokabulär vilket gjorde att de i kommentarsfälten kunde hjälpa och diskutera med varandra om hur de hade löst en uppgift. Wenger (1998) ser även lärandet som ökat deltagande i en praktik och han understryker även att man endast lär sig när man gör och att lärandet är ett evigt samspel mellan tanke och handling. Flera lärare uttryckte glädje över att ha programmerat i Scratch. Kursuppgifterna var den del av nätkursen där lärarna själva fick vara aktiva, vilket gör dem viktiga för lärarnas eget lärande.

I underkategorin *stegvis och via upprepning* nämnde lärarna försök och misstag, att det först gick långsamt och att de mot slutet hela tiden kom på flera saker som kunde utvecklas. En lärare nämnde också att det var bra att man bara kan använda de enklaste funktionerna och avancera därefter. Ju mera lärarna lärde sig om programmering i Scratch, desto mera kan man säga att de rörde sig från periferin av

programmeringsgemenskapen mot gemenskapens centrum (se figur 1 på s. 25) som Wenger (1998) beskriver att varje praktikbaserad gemenskap har. Jag tror att det är viktigt i all programmeringsundervisning att läraren kan planera uppgifter där inte alla programmeringsbegrepp och -funktioner kommer på samma gång. I början kanske eleverna bara får använda en eller vissa typer av block i Scratch. Samma gäller andra programmeringsverktyg.

I underkategorin *testa och felsöka* nämnde lärarna att de vid motgångar i programmeringsprocessen tog hjälp av internet, pausade eller frågade någon annan om de kunde pröva på programmet. Tidigare forskning uppmanar till att elever och lärare ska pröva varandras programmeringar för att utvecklingsprocessen ska fortsätta (Baytak & Land, 2011). I klassrumsundervisningen kan man ha elever att pröva varandras spel, lösa varandras problem eller försöka hitta så många lösningar som möjligt på ett problem. Scratch marknadsför sig även med idén att den sociala kontexten är av stor vikt när man programmerar (Maloney m.fl., 2010). Det är också viktigt att lära ut strategier som eleverna kan använda sig av när de kör fast i programmeringsprocessen.

I underkategorin att *återanvända och remixa* kunde endast fem lärares kommentarer placeras in, fast kurshållarna uppmuntrade till att remixa andras projekt i instruktionerna för kursuppgift 5. En lärare skrev att hen funderade på att göra det men ändå beslutade sig för att skapa ett helt nytt projekt från början. Idén med att remixa andras projekt är att man kan klara av att göra mycket svårare saker när man gör dem tillsammans med andra, än man skulle klara av att göra helt själv. Att remixa andras projekt är något som Wenger (1998) skulle kunna uppmuntra till, eftersom man då lär sig tillsammans med andra, precis så som teorin om praktikbaserad gemenskap rekommenderar.

Att *abstrahera och skapa modeller* är egentligen hemligheten bakom alla avancerade konstruktioner. Det fanns egentligen bara en kommentar som kunde kopplas till den här underkategorin. Läraren konstaterade att en idé som verkade enkel egentligen var ganska svår när hen började bena i den. Alla lärare har ändå lyckats dela upp problem i mindre delar, eftersom de lyckats programmera fungerande projekt. Att just den här

underkategorin var svårast för lärarna att spontant kommentera kan ha att göra med att den både har känts självklar men samtidigt abstrakt.

Även i dimensionen datalogiska perspektiv kunde väldigt få kommentarer placeras in, men trots det finns det flera intressanta och beskrivande kommentarer som belyser olika datalogiska perspektiv som lärarna har utvecklat. I underkategorin *att uttrycka sig* nämnde lärarna att det var roligt att programmera i Scratch och att de var ivriga. I Wengers (1998) teori om lärande kan man se det som att lärarnas identitet gradvis förändrades ju mer de lärde sig och ju mer de deltog i programmeringspraktiken. Även andra studier har kommit fram till att programmering i Scratch är glädjeframkallande. En studie som undersökte årskurserna 5 och 6 programmeringar i Scratch visade att eleverna tyckte att programmering var roligt, eftersom de fick jobba med intressanta teman som berörde dem själva och för att arbetssättet var aktivt (Sáez-López m.fl., 2016). En annan studie visade även att de som hade gått en Scratchkurs oftare anmälde sig till andra kurser i datavetenskap, de hade högre motivation och bättre självförtroende gällande programmering (Armoni m.fl., 2015).

I underkategorin *att förenas* nämnde lärarna att de hade använt hjälpmanualer, diskussionsforum, remixat andras projekt och bett någon annan att testa deras projekt för att komma framåt med sina egna projekt. Forskning visar att interaktionen mellan lärare och elev, samt mellan elev och elev är viktig för att eleven ska komma vidare i sin programmeringsprocess (Baytak & Land, 2011). Att programmera är inte en linjär process, utan processen är mera formad som en cirkel eller en spiral, där programmeraren måste pröva sig framåt i olika omgångar. En programmerare ska kunna analysera problemet, dela upp det i mindre bitar, planera en lösning i form av en sekvens och sen koda lösningen (Mannila m.fl., 2014, s. 4). Scratch hemsida uppmuntrar därför till interaktion mellan Scratchanvändarna (Maloney m.fl., 2010). Därför skulle de vara intressant att veta om lärarnas projekt hade blivit annorlunda om de skulle ha programmerat i par, eller suttit i samma klassrum och programmerat eller om de hade haft en obligatorisk handledning någon gång under programmeringsprocessen.

Nätkurslärarna önskade en kurs där endast Scratch skulle behandlas så att de skulle få lära sig mera om endast ett visuellt programmeringsspråk. En lärare önskade även

mera samarbete mellan lärarna med till exempel lektionsplaner och programmeringsuppgifter. Ett ökat samarbete mellan datalärare är även något som tidigare forskning visar att är viktigt. Det har till exempel föreslagits att stödgrupper borde formas över skolgränserna så att datalärarna får känna att de också tillhör en grupp. (Sentance & Csizmadia, 2017; Yadav m.fl., 2016). Scratch har även år 2009 lanserat ett nätforum för lärare Scratch-Ed[8] där lärare får dela med sig av tips, erfarenheter och lektionsplaner (Resnick m.fl., 2009).

Tidigare forskning har även visat att datalärare som har blivit handledda i klassen har förbättrat sina didaktiska och datavetenskapliga kunskaper och att deras isoleringskänsla har minskat (Margolis m.fl., 2017). Jag tror att alla lärare skulle dra stor nytta av att bli handledda i all sin undervisning, men kanske speciellt i sin programmeringsundervisning eftersom det är nytt för de flesta klasslärarna och matematiklärarna. Nu har lärarna gått kursen, men vad händer sen? Jobbar de i klassen med Scratch? Det är svårt att säga. Det är också svårt att säga vilka lärare som har gått kursen. Är det ivriga tutorlärare eller är det ett helt kollegium?

Forskning visar också att IT-lärare som har gått en kurs med inslag av Scratch har fått ett bättre självförtroende gällande nästan alla programmeringsuppgifter och att deras negativa attityder till programmering har minskat (Yukselturk & Altiok, 2017). Det belyser igen relevansen av Wengers (1998) teorier om lärande där ökat deltagande i en praktikbaserad gemenskap förändrar deltagarnas kunskaper, identitet och förståelse för i det här fallet programmering. Det är omöjligt att uttala sig om nätkurslärarnas självförtroende och attityder eftersom inga för- och eftertester har gjorts med specifika frågor som berör detta. Det är därför också omöjligt att spekulera i vad som hände med de lärarna som inte slutförde kursen och huruvida deras attityder till och självförtroende för programmering förändrades under kursens gång.

I underkategorin *att ifrågasätta* nämnde lärarna att stora programfiler blev tröga att jobba med i Scratch. Scratch bildredigering och val av typsnitt och storlek kunde även

[8] Scratch ED http://scratched.gse.harvard.edu/

utvecklas enligt lärarna. Att få matematiken att bli en betydande del i programmeringsundervisningen är också en sak som lärarna ställde sig kritiska till.

En del lärare kritiserade hur åk 6 skulle kunna använda Scratch och de tyckte att Scratch skulle passa bättre i åk 7–9. Tidigare forskning visar ändå att grundskolelever lärde sig datalogiska begrepp när de fick skapa spel i Scratch (Baytak & Land, 2011). I Scratch är det omöjligt att sammanlänka två block som inte passar ihop. Därför lämpar sig visuella programmeringsspråk bättre för barn eftersom de inte kräver någon kunskap om programmeringssyntax (Lye & Koh, 2014). Enligt EDU (2017) kan man redan börja programmera i Scratch i åk 3–6. Hur bra barnen klarar av att programmera i Scratch beror mera på när de har börjat bekantat sig med programmering och hur duktig läraren är på att handleda eleverna. Det som ännu är osäkert är hur man ska utvärdera barns och ungas utveckling av datalogiskt tänkande och programmeringsfärdigheter (t.ex. Brennan & Resnick, 2012; Heintz m.fl., 2016; Román-González m.fl., 2017).

Allmänt

Datalogiskt tänkande har ingen entydig definition. Till exempel har Barr och Stephenson (2011, s. 52) gjort en egen lista på några begrepp och färdigheter som också kunde innebära datalogiskt tänkande. Deras lista ser ut så här: datainsamling, dataanalys, datarepresentation, uppdelning av problem i mindre delar, abstraktion, algoritmer, automation, parallellisering och simulering. Vissa av dessa begrepp och färdigheter kan jämföras med Brennans och Resnicks (2012) modell av datalogiskt tänkande. Algoritmer kan jämföras med sekvenser, abstraktion och automation kan jämföras med villkorssats och loop och parallellisering kan jämföras med parallellism. Färdigheterna att dela upp problem i mindre delar, insamling, analys och representation av data samt simulering kan jämföras med flera av Brennans och Resnicks (2012) handlingsstrategier. Med den här utläggningen vill jag klargöra att datalogiskt tänkande inte är en enkel term med en entydig definition, men i min studie har jag valt att utgå från Brennans och Resnicks (2012) definition av datalogiskt tänkande i min analysprocess. Jag har analyserat lärarnas Scratchkoder och kommentarer utgående från de datalogiska begrepp som finns i Brennans och Resnicks (2012) modell av datalogiskt tänkande.

Utgående från kommentarerna och de andra inläggen på kursbloggen verkar nätkurslärarana vara mycket nöjda med kursen. Lärarnas ökade deltagande i den praktikbaserade gemenskapen gav resultat i form av mera kunskap och färdighet att programmera speciellt i Scratch, men även i andra programmeringsverktyg som de bekantade sig med under nätkursen (Wenger, 1998). Att alla lärare inte slutförde kursen kan bero på tidsbrist, att kursuppgift 5 var för svår eller att man bara ville komma åt kursmaterialet. Det går inte att generalisera den här studiens resultatet. De lärare som redan kan programmera har inte gått den här grundkursen i programmering. Jag vet inte heller om lärarna gjorde sitt yttersta när de gjorde kursuppgift 5. Vissa lärare kanske bara var nöjda med att de hade åstadkommit något medan andra hade satt mycket mera tid på sina projekt. Skulle lärarna ha gjort bättre om de då i stunden vetat att deras projekt skulle analyseras? Jag tror att lärarna fick goda grundläggande programmeringskunskaper i och med kursen *Programmering i grundskolan – vad, hur och varför?*. Flera av nätkurslärarna önskade ändå att få gå en specifik kurs i bara Scratch.

Varje år ordnas det flera fortbildningskurser om programmering – men vad händer efter kurserna? Vågar lärarna pröva på att använda det som de har lärt sig i klassen? Vem som sköter programmeringsundervisningen i kommunerna och skolorna varierar. Alla klasslärare kanske inte behöver undervisa i programmering. Det kan komma någon utifrån som håller programmeringsdelen eller sen kanske det finns en programmeringsexpert i varje skola.

7.2 Metoddiskussion

Syftet med avhandlingen var att studera finlandssvenska grundskolelärares datalogiska tänkande inom ramen för nätkursen *Programmering i grundskolan – vad, hur och varför?,* som gavs som fortbildningskurs för lärare. Jag tycker att avhandlingen uppfyller sitt syfte i den mån som datamaterialet ger möjlighet till. Ett gedignare material skulle ge ännu djupare kunskap om finlandssvenska grundskolelärares datalogiska tänkande.

Jag valde att bädda in min studie i Wengers (1998) teori om *praktikbaserad gemenskap*. Wenger anser att lärande sker när man deltar i en social gemenskap och aktivt gör saker tillsammans. Jag anser att det fungerade bra att ha Wenger som teoretisk bakgrund för att analysera lärares lärande. Det kanske inte är en regelrätt forskningsansats, men det ger ändå en teoretisk grund för min studie. I den här studien är lärarna deltagare i nätkursens programmeringsgemenskap där de passivt får ta emot information men också aktivt testa på programmeringsverktyg, reflektera över sig själv och fundera på sin egen skolas framtid rörande programmering. Lärarna tog del av samma kursmaterial och utförde samma kursuppgifter och delade sina kursuppgifter med varandra. Nätkursens hemsida, kursblogg och nätforum skapade tillhörighet och gemenskap lärarna emellan samt mellan kurshållarna och lärarna. Samtidigt utgör dessa kursens kommunikationskanaler. Goda kommunikationskanaler är viktiga enligt Wenger (1998) därför att de håller samman den praktikbaserade gemenskapen.

I och med kursen rör sig lärarna hela tiden från periferin i programmeringsgemenskapens cirkel inåt mot kärnan av gemenskapen. Det fanns flera positiva kommentarer rörande kursen, vilket betyder att lärarna har känt att det var meningsfullt att lära sig mera om programmering. I takt med att man lär sig nya saker omformas även ens identitet (Wenger, 1998). Efter kursen kände sig lärarna förhoppningsvis lite säkrare i rollen som programmeringsundervisare i grundskolan.

Kvalitativa fallstudier används då forskaren vill få djupgående kunskap om en viss situation (Merriam, 1994). Det föll sig naturligt att den här studien blev en kvalitativ fallstudie eftersom datamaterialet bestod av lärares Scratchprojekt och tillhörande kommentarer. Datamaterialet var en deluppgift i en nätkurs som alla lärare hade gått samtidigt år 2016. Eftersom det inte finns några direkta sätt att utvärdera människors datalogiska tänkande skulle det ha varit svårt att göra en kvantitativ studie och utvärdera ett stort sampel av lärare och deras datalogiska tänkande.

En fallstudie går inte att generalisera (Denscombe, 2016; Patel & Davidson, 2011). Ingen annan grupp kommer att göra den här uppgiften lika, så resultatet går inte att generalisera på alla lärare. Möjligtvis skulle det gå att jämföra den här gruppen av lärare med den grupp av lärare som har gått samma nätkurs senare. Mitt resultat visar ändå ett exempel på hur lärares datalogiska tänkande kan se ut efter att de har fått

bekanta sig med Scratch via en nätkurs. Deras kommentarer i samband med Scratchprojekten belyser tankar som även andra lärare kan ha om programmering i grundskolan.

Denscombe (2016), men även Patel och Davidson (2011) och Merriam (1994) skriver att fallstudien tillåter och uppmuntrar till att forskaren använder flera olika slags data, för att på det viset ge en så exakt bild som möjligt av företeelsen. Jag hade tyvärr bara tillgång till lärarnas Scratchprojekt och tillhörande kommentarer, men det skulle ha varit intressant att veta vad som rörde sig i nätkurslärarnas huvuden medan de programmerade, vilket mål de hade med sitt Scratchprojekt och vad de tänkte när de var klara. Att intervjua dem i samband med kursuppgift 5 och 6 skulle ha varit intressant. Ett annat alternativ kunde ha varit att låta alla deltagare besvara en enkät eftersom det var synd att vissa hade så få kommentarer. En skärminspelning av deras skärmar och ljudupptagning medan de programmerade skulle även ha varit intressant för att se hur deras programmeringsprocess framskred.

Analysverktyget som jag använde har jag utvecklat ur Brennans och Resnicks (2012) definition av datalogiskt tänkande. Analysverktyget fungerade bra för mitt syfte. Det som var svårast med analysarbetet var att förstå vilka blocktyper som hörde ihop med vilka datalogiska begrepp. Ännu bättre skulle analysverktyget ha fungerat om lärarnas kommentarer skulle ha varit längre. Det var svårare att uttala sig om nätkurslärarnas begreppskunskap än om deras handlingsstrategier och perspektiv. Begreppsanalysen kunde ännu förbättras på något sätt. Eventuellt kunde man räkna hur många sekvenser, loopar etc. som fanns i varje Scratchprojekt. Att analysera meningen med hela Scratchprojektet och hur bra projektet sist och slutligen fungerar skulle också kunna ge en djupare förståelse för lärares datalogiska tänkande.

Med tillförlitlighet menas huruvida en studie kan upprepas med samma resultat (Denscombe, 2016). Det är svårt att upprepa kvalitativa fallstudier eftersom den sociala kontexten aldrig går att återskapa helt. Jag tror ändå att en nätkurskontext kan vara relativt likadan om instruktionerna är samma. Varje kursdeltagare har olika mycket erfarenhet i bagaget och alla människor tolkar samma instruktioner på olika sätt, men alla ska ändå slutföra samma uppgifter. Mitt datamaterial består av dokument, vilket gör att materialet finns bevarat för andra att läsa om man frågar lov.

Jag har inte varit med när datamaterialet samlades in, vilket gör att jag inte har kunnat påverka datamaterialet alls. Dokument är ett icke-reaktivt material och det förhöjer trovärdigheten hos data (Bryman, 2011, s. 488–4994, 499–500). Dessutom är mitt analysverktyg väldigt tydligt avgränsat, vilket gör att oberoende vem som analyserar samma material som jag har analyserat kommer hen att få ungefär samma resultat. Jag säger ungefär samma resultat eftersom jag har varit ensam i min analysprocess och någon annan kan tolka lärarnas kommentarer på ett annat sätt än jag har gjort.

Med trovärdighet menas huruvida en studie mäter det som den anser sig mäta Denscombe (2016). Det är omöjligt att få helt objektiva och träffsäkra data i kvalitativa studier, vilket gör det problematiskt att generalisera kvalitativa fallstudier. Jag anser inte att den här studiens resultat kan generaliseras på alla finlandssvenska grundskolelärares datalogiska tänkande, men jag tror att vissa handlingsstrategier och perspektiv som de här nätkurslärarna har gett uttryck för även kan kännas bekanta för andra lärare. Genom hela min avhandling har jag varit väldigt transparent. Min metod och mina val beskrivs utförligt och jag har delat med mig av exempel från mitt datamaterial i min resultatdel. Hela den här transparensen gör att trovärdigheten i min avhandling höjs.

Det skulle ha varit bättre om man skulle ha frågat nätkurslärarna om deras samtyckte att delta i den här studien redan i samband med att kursen hölls. Kurshållaren hjälpte mig att skicka e-post till deltagarna i januari 2018, men ett e-postmeddelande kan lätt gå förbi eller så kan någon ha bytt e-postadress. I resultatdelen nämns ingen vid namn utan jag refererar till Scratchprojekten som projekt 1, 2, 3 o.s.v.

7.3 Förslag till fortsatt forskning

Det skulle vara intressant att se vad som hade hänt med lärarnas Scratchprojekt om de hade fått träffas och diskuterat sina projekt med varandra och med kurshållarna. Lite diskussion uppstod i blogginläggens kommentarsfält, men om de skulle ha suttit i samma rum och programmerat ensamma/parvis kanske de skulle ha lyckats göra ännu mera avancerade projekt.

En annan intressant aspekt att studera skulle vara på vilket sätt nätkurslärarna arbetade med sina Scratchprojekt genom att filma deras skärmar och be dem tänka högt och spela in ljud. I vilka skeden fastnade de i sin programmeringsprocess och vad gjorde de då? Man skulle även kunna styra slutkommentarerna mera genom att t.ex. be lärarna att besvara en enkät. Det skulle också vara intressant att undersöka vad lärarna gör efter kursen. Fortsätter lärarna att delta i programmeringsgemenskapen på eget initiativ? Hur inför lärarna det som de har lärt sig om programmering i sina klassrum? Ger nätkursen sådana redskap som lärarna kan och väljer att använda sig av i klassen efteråt eller blir kursen bara något som de har gjort och sedan glömmer bort?

Litteratur

Armoni, M., Meerbaum-Salant, O. & Ben-Ari, M. (2015). From Scratch to "Real" Programming. *ACM Transactions on Computing Education, 14*(4), 25:1–25:15. DOI: 10.1145/2677087

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K−12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–55. DOI: 10.1145/1929887.1929905

Baytak, A. & Land, S. M. (2011). An investigation of the artifacts and process of constructing computers games about environmental science in a fifth-grade classroom. *Educational Technology Research and Development, 59*(6), 765–782. DOI: 10.1007/s11423-010-9184-z

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016). *Developing computational thinking in compulsory education – Implications for policy and practice*. Luxemburg: Publications Office of the European Union. DOI: 10.2791/792158

Bower, M., Wood, L. N., Lai, J. W., Howe, C., Lister, R., Mason, R., Highfield, K., & Veal, J. (2017). Improving the Computational Thinking Pedagogical Capabilities of School Teachers. *Australian Journal of Teacher Education, 42*(3), 51–72. DOI: 10.14221/ajte.2017v42n3.4

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking, s. 1-25. American Educational Research Association meeting. Vancouver, Canada. Hämtad 6 november 2017, från https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Bryman, A. (2011). *Samhällsvetenskapliga metoder*. Stockholm: Liber.

CAS. (2015). *Computational Thinking. A guide for teachers*. Hämtad 2 oktober 2017, från https://community.computingatschool.org.uk/resources/2324

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014). Introducing teachers to computational thinking using unplugged storytelling. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WIPSCE 2014)*, 89–92. New York: ACM.

Denning, P. J. (2009). The Profession of IT: Beyond Computational Thinking. *Communications of the ACM, 52*(6), 28–30. DOI: 10.1145/1516046.1516054

Denning, P. J. (2017). Remaining Trouble Spots with Computational Thinking. *Communications of the ACM, 60*(6), 33–39. DOI: 10.1145/2998438

Denscombe, M. (2016). *Forskningshandboken – för småskaliga forskningsprojekt inom samhällsvetenskaperna*. Lund: Studentlitteratur.

EDU. (2016). *Programmering*. Hämtad 1 november 2017, från http://www.edu.fi/planera/grundlaggande_utbildning/matematik/lp2016_-_stodmaterial_i_matematik/programmering

EDU A. (2017). *Lärandeprogression*. Hämtad 1 november 2017, från http://www.edu.fi/hitta_material/it_i_skolan/programmering/larandeprogression

Eklöf, A. (2005). *Diskussionsforum i en lärplattform – ett socialt och kunskapsmässigt stöd för distansstudenter* (Rapport från Centrum för lärande och undervisning, 2005:6). Borås: Högskolan i Borås.

European Schoolnet. (Uppdaterad 2015*). Computing our future*. Hämtad 3 oktober 2017, från http://fcl.eun.org/documents/10180/14689/Computing+our+future_final.pdf/746e36b1-e1a6-4bf1-8105-ea27c0d2bbe0

FSL. Senast uppdaterad 23.8.2017. Fortbildning. Hämtad 25 september 2017, från https://www.fsl.fi/315-rad-och-stoed/arbetstid/3492-fortbildning

Grover, S. & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38–43. DOI: 10.3102/0013189X12463051

Heintz, F. & Mannila, L. (u.å.). *Datalogiskt tänkande för grundskolan – vad, hur och varför?* [Digital presentation] Linköpings universitet. Hämtad 13 november 2017, från https://www.ida.liu.se/~frehe08/FredrikHeintzLindaMannila-skolverket-20150302.pdf

Heintz, F., Mannila, L. & Färnqvist, T. (2016). A Review of Models for Introducing Computational Thinking, Computer Science and Computing in K–12 Education. Frontiers in Education Conference (FIE), 2016 IEEE. DOI: 10.1109/FIE.2016.7757410

Hemmendinger, D. (2010). A Plea for Modesty. *ACM Inroads, 1(*2), 4–7. DOI 10.1145/1805724.1805725

Henriksson, S. [u.å.]. Programmering. I *Nationalencyklopedin*. Hämtad 27 september 2017, från http://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/programmering

Henriksson, S. & Nordström, B. [u.å.]. Programspråk. I *Nationalencyklopedin*. Hämtad 27 september 2017, från http://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/programspr%C3%A5k

Imsen, G. (2006). *Elevens värld. Introduktion till pedagogisk psykologi*. Lund: Studentlitteratur.

ISTE. (2016). *ISTE Standards for students.* Hämtad 2 oktober 2017, från http://www.iste.org/standards/for-students

ISTE & CSTA. (2011). *Concepts of Compuational Thinking*. Hämtad 2 oktober 2017, från https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/CompThinkingFlyer.pdf

Jaipal-Jamani, K. & Angeli, C. (2017). Effect of Robotics on Elementary Preservice Teachers' Self-Efficacy, Science Learning, and Computational Thinking. *Journal of Science Education and Technology, 26*(2), 175–192. DOI: 10.1007/s10956-016-9663-z

Kafai, Y. B. & Burke, Q. (2013). Computer Programming Goes Back to School. *Phi Delta Kappan, 95*(1), 61–65. DOI: 10.1177/003172171309500111

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., … Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37. DOI: 10.1145/1929887.1929902

Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*(2014), 51–61. DOI: 10.1016/j.chb.2014.09.012

Maloney, J., Resnick, M., Rusk, N., Silverman, B. & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 1–2. DOI: 10.1145/1868358.1868363

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L. & Settle, A. (2014). Computational Thinking in K−9 Education. *ITiCSE '14 Proceedings of the 2014 Conference on Innovation & technology in Computer Science Education*. 1–29. DOI: 10.1145/2713609.2713610

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T. & Mendes, A. J. (2018). Learning Computational Thinking and Scratch at distance. *Computers in Human Behaviors, 80*(2018), 470–477. DOI: 10.1016/j.chb.2017.09.025

Margolis, J., Ryoo, J. & Goode, J. (2017). Seeing myself through someone else's eyes: The value of in-classroom coaching for computer science teaching and learning. *ACM Transactions on Computing Education, 17*(2), 6:1–6:18. DOI: 10.1145/2967616

Menekse, M. (2015). Computer science teacher professional development in the United States: a review of studies published between 2004 and 2014. *Computer Science Education, 25*(4), 325–350. DOI: 10.1080/08993408.2015.1111645

Merriam, S. B. (1994). *Fallstudien som forskningsmetod*. Lund: Studentlitteratur.

Munson, A., Moskal, B., Harriger, A., Lauriski-Karriker, T. & Heersink, D. (2011).Computing at the high school level: Changing what teachers and students know and believe. *Computers & Education, 57*(2), 1836–1849. DOI: 10.1016/j.compedu.2011.03.005

Norrena, J. (2013). *Opettaja tulevaisuuden taitojen edistäjänä: "jos haluat opettaa noita taitoja, sinun on ensin hallittava ne itse"*. Doktorsavhandling. Jyväskylän yliopisto.

Patel, R. & Davidson, B. (2011). *Forskningsmetodikens grunder - Att planera, genomföra och rapportera en undersökning*. Lund: Studentlitteratur.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K. & Silverman, B. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67. DOI: 10.1145/1592761.1592779

Román-González, M., Pérez-González, J-C. & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior 72(2017)*, 678–691. DOI: 10.1016/j.chb.2016.08.047

Sáez-López, J-M., Roman-González, M. & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*(2016), 129–141. DOI: 10.1016/j.compedu.2016.03.003

Scaffidi, C. & Chambers, C. (2012). Skill Progression Demonstrated by Users in the Scratch Animation Environment. *International Journal of Human Computer Interaction, 28*(6), 383–398. DOI: 10.1080/10447318.2011.595621

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

Smith, M. (30 januari 2016). *Computer Science For All*. [Blogginlägg]. Hämtad 12 oktober 2017, från https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

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

Voogt, J. & Roblin, N. P. (2012). A comparative analysis of international frameworks for 21st century competences: Implications for national curriculum policies. *Journal of Curriculum Studies, 44(3)*, 299–321. DOI: 10.1080/00220272.2012.668938

Voogt, J., Fisser, P., Good, J., Mishra, P. & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715–728. DOI: 10.1007/s10639-015-9412-6

Wenger, E. (1998). *Communities of Practice. Learning, Meaning, and Identity.* Cambridge: Cambridge University Press.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–35. DOI: 10.1145/1118178.1118215

Yadav, A., Gretter, S., Hambrusch, S. & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education, 26*(4), 235–254. DOI: 10.1080/08993408.2016.1257418

Yukselturk, E. & Altiok, S. (2017). An investigation of the effects of programming with Scratch on the preservice IT teachers' self-efficacy perceptions and attitudes towards computer programming. *British Journal of Educational Technology, 48*(3), 789–801. DOI: 10.1111/bjet.12453

Bilaga 1

Brennans och Resnicks (2012) definition av datalogiskt tänkande: En ingående förklaring på de tre dimensionerna och deras underkategorier

Brennan och Resnick (2012) delar in det datalogiska tänkandet i tre dimensioner. De tre dimensionerna är dimensionen datalogiska begrepp, dimensionen datalogiska handlingsstrategier och dimensionen datalogiska perspektiv. Här under förklaras dimensionerna och deras underkategorier mera ingående.

Dimension 1: Datalogiska begrepp

Med dimensionen *datalogiska begrepp* menas de begrepp som en programmerare kommer att stöta på och lära sig använda när hen programmerar. Brennan och Resnick (2012, s. 3–6) har identifierat sju olika begrepp som beskriver vad man lär sig när man programmerar.

Att skriva en *sekvens* (eng. *sequence*) innebär att man beskriver en aktivitet eller en uppgift som en serie instruktioner som kan utföras av en dator.

Att använda sig av en *loop* (eng. *loop*) innebär att man använder sig av en mekanism som upprepar en sekvens flera gånger, istället för att skriva samma sekvens flera gånger i rad.

Att en specifik *händelse* (eng. *event*) startar eller bestämmer nästa händelse i sekvensen en väsentlig del av programmering. T.ex. är det viktigt att programmera startknappen till ett spel så att spelet börjar när man trycker på start.

Att få saker att hända *parallellt* (eng. *parallelism*) innebär i programmering att flera olika sekvenser sker samtidigt. I Scratch kan man t.ex. programmera så att man samtidigt hör bakgrundsmusik, ser en katt dansa och hör katten säga att den älskar att dansa.

Att använda sig av *villkorssatser* (eng. *conditionasl*) betyder att man ger instruktioner med olika alternativ, där olika villkor måste uppfyllas beroende på alternativet och utgående från vilket villkor som uppfylls utförs sedan följande instruktion. Man brukar

även kalla det för if-satser eller om-satser. T.ex. Om katten rör den gula kuben blir katten osynlig, annars förblir katten synlig.

Att använda sig av olika o*peratorer* (eng. *operators*) gör det möjligt att uttrycka bl.a. matematiska operationer i ett program som addition, subtraktion, multiplikation och division. En operator är en symbol eller ett nyckelord som utför en viss beräkning eller jämför olika tal och värden.

Det sista begreppet *data* (eng. *data*) innebär olika sätt att lagra, hämta och uppdatera värden. I Scratch lagras data som variabler (innehåller ett nummer eller en sträng) och listor (innehåller en samling av numror eller strängar). T.ex. kan man använda sig av en variabel när man räknar poäng i ett spel som man har programmerat.

Dimension 2: Datalogiska handlingsstrategier

Med dimensionen *datalogiska handlingsstrategier* menas de handlingsstrategier som en programmerare utvecklar över tid när hen blir en duktigare programmerare. Brennan och Resnick (2012, s. 7–9) har identifierat fyra datalogiska handlingsstrategier som utvecklas med tiden.

För det första lär man sig att programmera *stegvis och via upprepning* (eng. *being incremental and iterative*). Att planera ett spel är inte en rak, sekventiell process som börjar med att man väljer ett tema, utvecklar en plan för designen och sedan kodar. Processen är mera cyklisk; man utvecklar något, testar det och utvecklar det vidare på basis av nya erfarenheter och nya idéer.

För det andra lär man sig att programmera genom att *testa och felsöka* (eng. *testing and debugging*). Saker fungerar väldigt sällan som man tänkt sig och därför är det bra att ha strategier för hur man kan göra när man stöter på problem. Bra strategier är t.ex. att försöka identifiera källan till problemet, läsa igenom koden igen, experimentera med koden, skriva in koden igen, hitta exempel på koder som fungerar, fråga eller berätta för någon annan om problemet eller att ta en paus.

För det tredje lär man sig att programmera genom att *återanvända och remixa* (eng. *reusing and remixing*) andras program. När man återanvänder och remixar delar av

andras program kan man få idéer och lyckas programmera program som är mycket svårare än om man själv skulle tvingats göra allting. Återanvändning och remixing lär en också att läsa kod kritiskt och det väcker frågor om vem som har rätten att äga och författa (vad är rimligt att låna från andra, hur kan man hedra originalförfattaren och hur kan man bedöma slutprodukten?).

För det fjärde lär man sig att programmera genom att *abstrahera och skapa modeller* (eng. *abstracting and modularizing*). Det betyder att man först måste lära sig att bygga små modeller och kombinera dem för att kunna skapa ett stort spel. Färdigheten att dela upp problem i mindre delar är väldigt viktig i all planering och problemlösning. I Scratch kan man t.ex. separera karaktärers beteende, utseende och aktiviteter från varandra för att lättare kunna testa och felsöka koden, samtidigt som det blir lättare för andra att läsa koden.

Dimension 3: Datalogiska perspektiv

Med dimensionen *datalogiska perspektiv* menas de nya perspektiv som en programmerare utvecklar i och med att hens programmeringsfärdigheter utvecklas. Brennan och Resnick (2012, s. 10–11) har identifierat tre olika datalogiska perspektiv som beskriver hur programmerares självförståelse, relation till andra och relation till den teknologiska världen utvecklas.

Det första perspektivet handlar om att få *uttrycka sig* (eng. *expressing*). Människor omges idag av interaktiva medier som konsumeras via klickningar, skrollande och chatt. Dessa aktiviteter lär oss att använda teknologi, men det utvecklar inte det datalogiska tänkandet. En person som tänker datalogiskt ser dataanvändningen som något mera än bara konsumtion – de ser dataanvändningen som ett medium där de själva kan planera och skapa idéer. Det ser dataanvändningen som ett sätt att förverkliga sina fantasier.

Det andra perspektivet handlar om att *förenas* (eng. *connecting*) med andra. Kreativitet och lärande är färdigheter som utvecklas i sociala sammanhang. Därför är det naturligt att Scratch-användare har stor nytta av att jobba tillsammans (ansikte mot ansikte eller online). Man brukar tala om att jobba med andra och att jobba för andra. *Att jobba med*

andra gör att man klarar av att göra mera än man skulle kunna göra själv, eftersom man kan få hjälp med att fixa fel, man kan studera eller remixa andras koder eller hitta en samarbetspartner. Man utvecklas också när man jobbar för andra för då har man en autentisk publik. *Att jobba för andra* kan betyda att man underhåller andra, engagerar sig i andras projekt, utrustar andra eller utbildar andra.

Det tredje perspektivet handlar om att *ifrågasätta* (eng. *questioning*). För att utveckla det datalogiska tänkandet måste man känna sig berättigad att ställa frågor om och med teknologi. Människor ska känna sig sammankopplade med teknologin som omger dem och känna att de har möjlighet att förhandla om den tekniska världens realiteter.

Bilaga 2

Analysverktyget

Nätkurslärarnas datalogiska tänkande

	PROJEKT	1	2
Datalogiska begrepp	**sekvens**	x	x
	loop	upprepa för alltid	repetera tills
	händelse	startknapp, spelet börjar	startknapp, spelet börjar
	parallellism	när grön flagga klickas på x5	nej
	villkorssats	om, annars	om svar, annars
	operator	nej	antal kvar<frågor per rond
	data	räknar poäng	räknar poäng
		-1	-1
	PROJEKT	1	2
Datalogiska handlings-strategier	**Att gå stegvis och upprepa**	I något skede ska jag nog få scriptet att fungera så att jag får flera nivåer med mera fart på bollen och eventuellt flera bollar att fånga (det testade jag också lite på redan...)	Testade med två dimensionell tabell, fildelning, men gjorde sedan två skila tabeller
	Att testa och felsöka	Jag kunde lägga till en ny nivå (level) med en ny bakgrund då bollen rörde sig snabbare men sedan bara slutade spelet vid en viss poäng. Så jag gav upp för tillfället	Det naturliga skulle ha varit att ha frågorna och svaren i var sin kolumn i en tabell, men nu verkar det inte gå att hantera tvådimensionella tabeller i Scratch. Jag löste problemet med att skapa två tabeller. Frågorna finns i den ena och svaren på motsvarande rader i den andra.
	Att återanvända och remixa	nej	nej
	Att abstrahera och skapa modeller	nej	nej
Datalogiska perspektiv	**Att få uttrycka sig**	Sedan blev jag riktigt ivrig och ville göra ett spel som jag som liten spelade på vår första datamaskin på 80-talet. "Tennispelet" där man skulle fånga bollen med brädan och spela mot datorn.	nej
	Att förenas	Jag kom ganska långt. Jag fick mycket tips från Scratch men sen buggade något till sig.	Det skulle vara mera optimalt att läsa in frågorna och svaren från en fil, men jag hade inte kunskaper i hur filhantering fungerar i Scratch.
	Att ifrågasätta	Scratch kan nog vara utmanande att använda i åk 1-6. Jag kan tänka mig att vissa elever med ett brinnande intresse nog klarar av Scratch. Men att få alla med på det här kanske inte går.	När programfilen blev rätt stor fich jag ganska stora problem att hantera den i editorn. Allting gick mycket trögt när jag släpade och släppte blocken.
		-2	-3